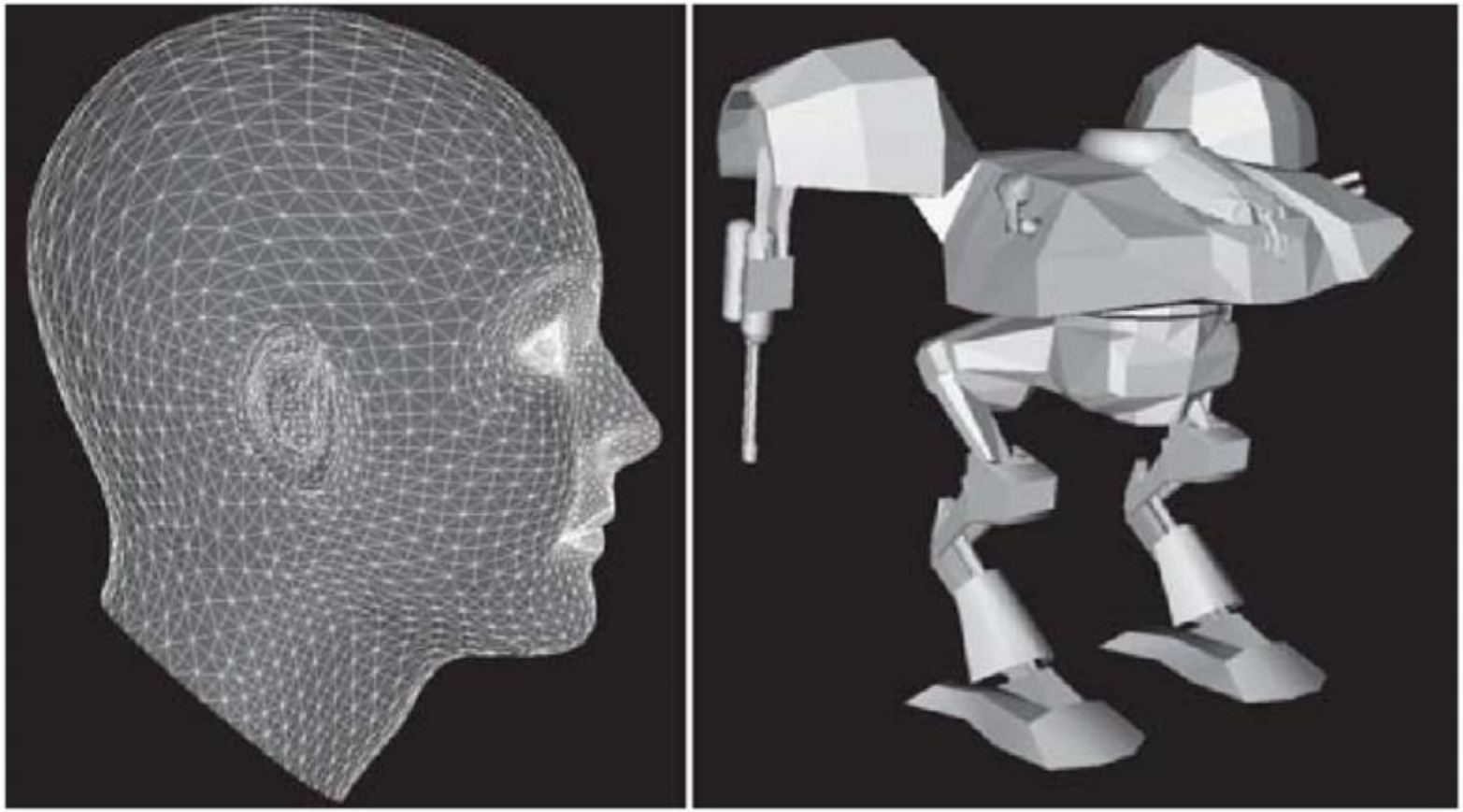
Computación Gráfica

Ing. Gabriel Ávila, MSc.

Modelado de superficies y sólidos

Estrategias para construcción de sólidos

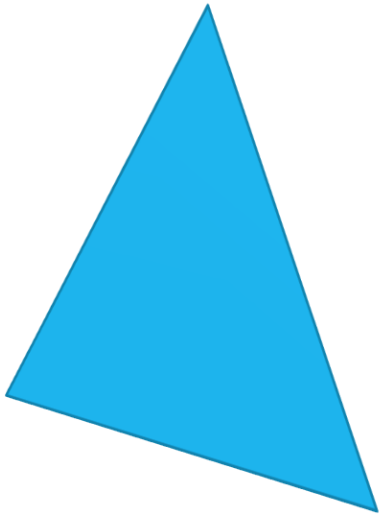
- Modelado mediante mallas de polígonos.
- Geometría Sólida Constructiva (CSG).
- Modelado con curvas o superficies.
- Modelado por grillas de vóxeles.
- …



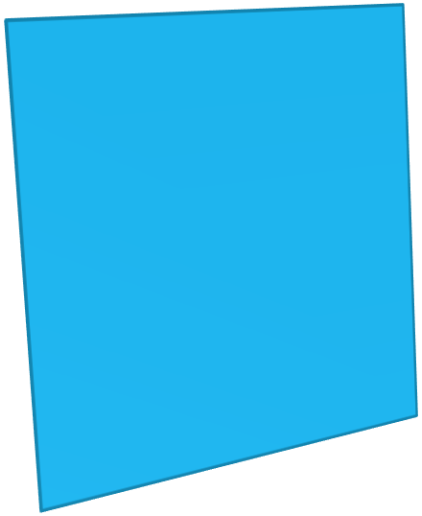
MODELADO MEDIANTE MALLAS DE POLÍGONOS

Mallas de polígonos

En su forma más sencilla, ***una malla de polígonos está conformada por una lista de polígonos***. Permiten representar objetos complejos.



TRIÁNGULO



CUADRILÁTERO

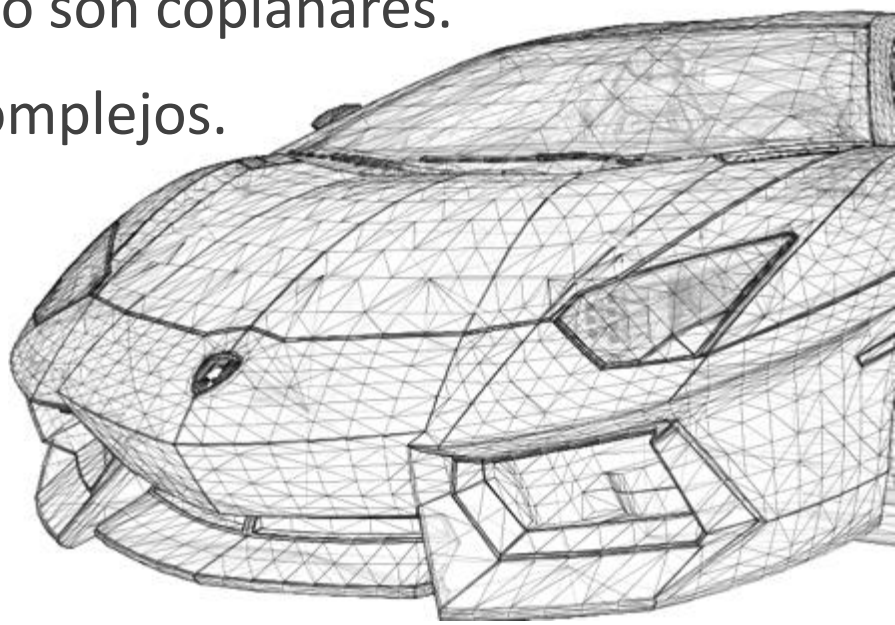
Mallas de triángulos

Varios triángulos unidos mediante sus aristas, formando una superficie.

Ventajas:

Todos los vértices de un triángulo son coplanares.

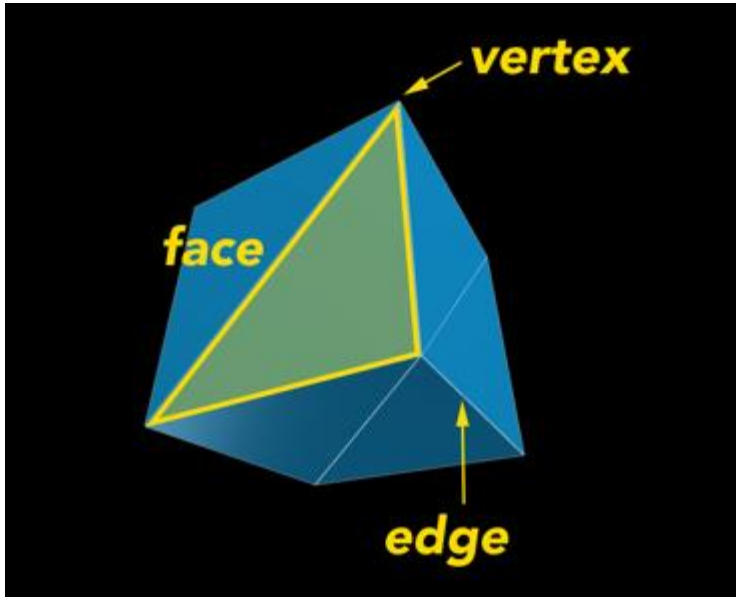
Permiten representar objetos complejos.



Objetos 3D

- Un objeto 3D es un volumen formado por parches bidimensionales, llamados caras (***faces***).
- Cada una de estas caras está bordeada por elementos unidimensionales llamados bordes, curvas o aristas (***edges***).
- Estos bordes están definidos por puntos en el espacio (***vertex***).

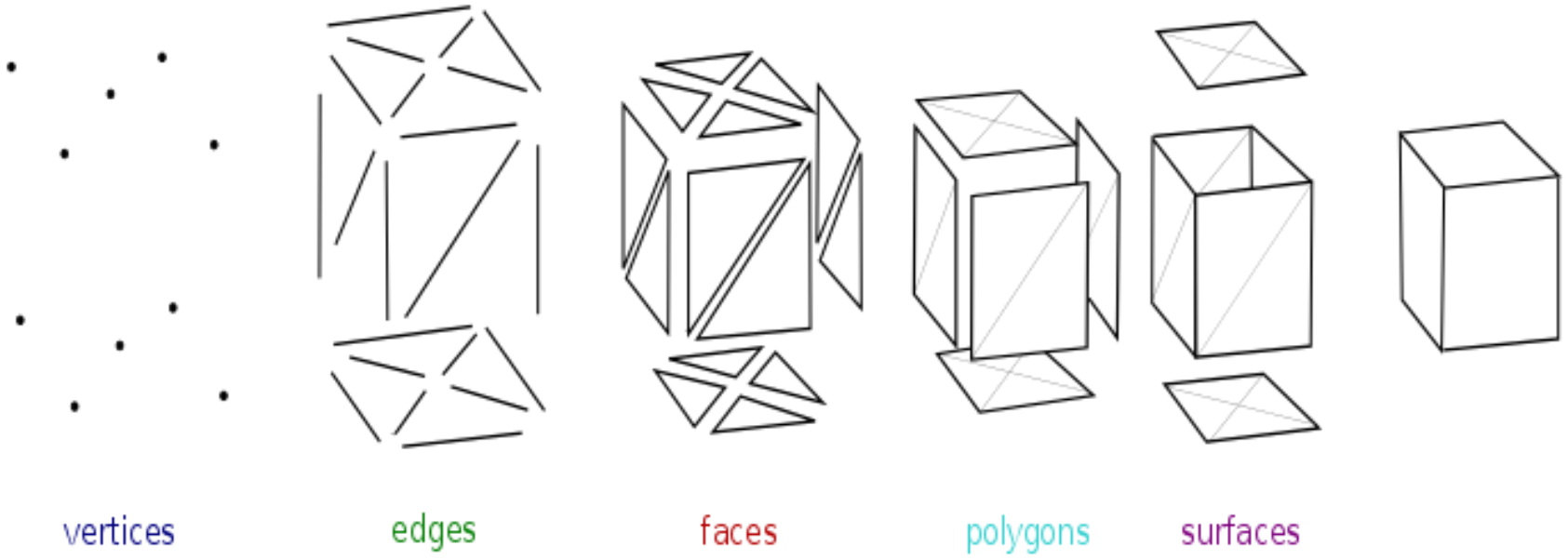
Objetos 3D



Vértices. 3 por cada triángulo. Pueden ser compartidos entre triángulos.
Aristas. Conectan dos vértices. Cada triángulo tiene 3 aristas.
Caras. La superficie del triángulo, se puede definir por 3 aristas o 3 vértices.

Tomado de: https://codepen.io/rachsmith/post/beginning-with-3d-webgl-pt-2-geometry

Polygon mesh

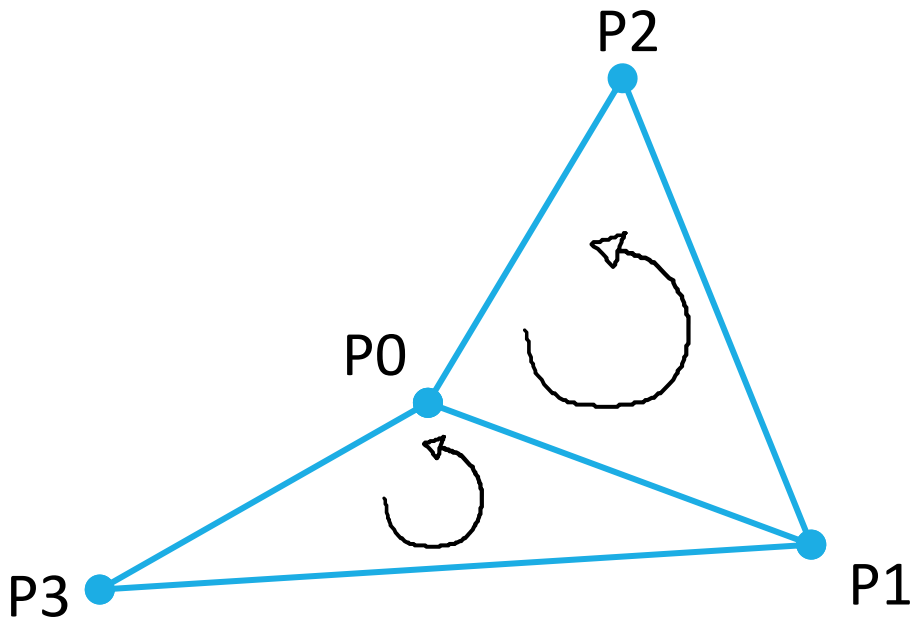


Tomado de: https://en.wikipedia.org/wiki/Polygon_mesh

Malla de vértices indexada

Vertex List	
index	Vertex Position
0	X_0, Y_0, Z_0
1	X_1, Y_1, Z_1
2	X_2, Y_2, Z_2
3	X_3, Y_3, Z_3

Triangle List	
index	Vertex Indices
	0, 1, 2
1	X_1, Y_1, Z_1
…	…



THREE.js

```
//GEOMETRIA
var geoTriangulo = new THREE.Geometry();
//VERTICES
var v1 = new THREE.Vector3(  2.0,  0.5,  0.0 );
var v2 = new THREE.Vector3(  0.0, -2.0,  0.0 );
var v3 = new THREE.Vector3( -0.5, -1.5,  0.0 );
geoTriangulo.vertices.push( v1 );
geoTriangulo.vertices.push( v2 );
geoTriangulo.vertices.push( v3 );
//CARAS
geoTriangulo.faces.push( new THREE.Face3( 0, 1, 2 ) );
geoTriangulo.computeFaceNormals();
//MATERIAL Y CREACION DEL OBJETO
var matPlano = new THREE.MeshBasicMaterial();
var triangulo = new THREE.Mesh( geoTriangulo, matPlano );
scene.add( triangulo );
```

Otras representaciones: *Face-Vertex Mesh*

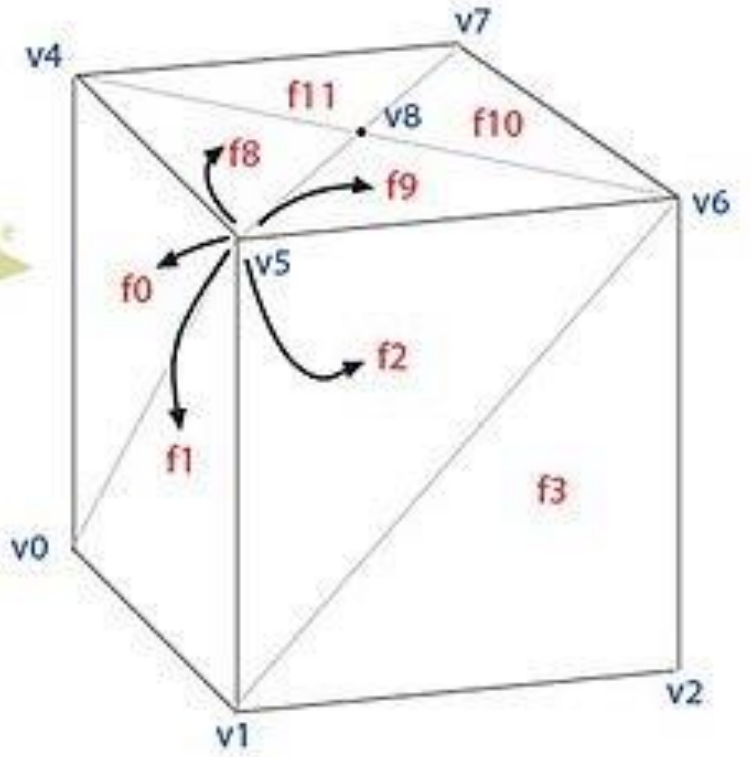
Face List

f0	v0 v4 v5
f1	v0 v5 v1
f2	v1 v5 v6
f3	v1 v6 v2
f4	v2 v6 v7
f5	v2 v7 v3
f6	v3 v7 v4
f7	v3 v4 v0
f8	v8 v5 v4
f9	v8 v6 v5
f10	v8 v7 v6
f11	v8 v4 v7
f12	v9 v5 v4
f13	v9 v6 v5
f14	v9 v7 v6
f15	v9 v4 v7

Vertex List

v0	0,0,0	f0 f1 f12 f15 f7
v1	1,0,0	f2 f3 f13 f12 f1
v2	1,1,0	f4 f5 f14 f13 f3
v3	0,1,0	f6 f7 f15 f14 f5
v4	0,0,1	f6 f7 f0 f8 f11
v5	1,0,1	f0 f1 f2 f9 f8
v6	1,1,1	f2 f3 f4 f10 f9
v7	0,1,1	f4 f5 f6 f11 f10
v8	.5,.5,0	f8 f9 f10 f11
v9	.5,.5,1	f12 13 14 15

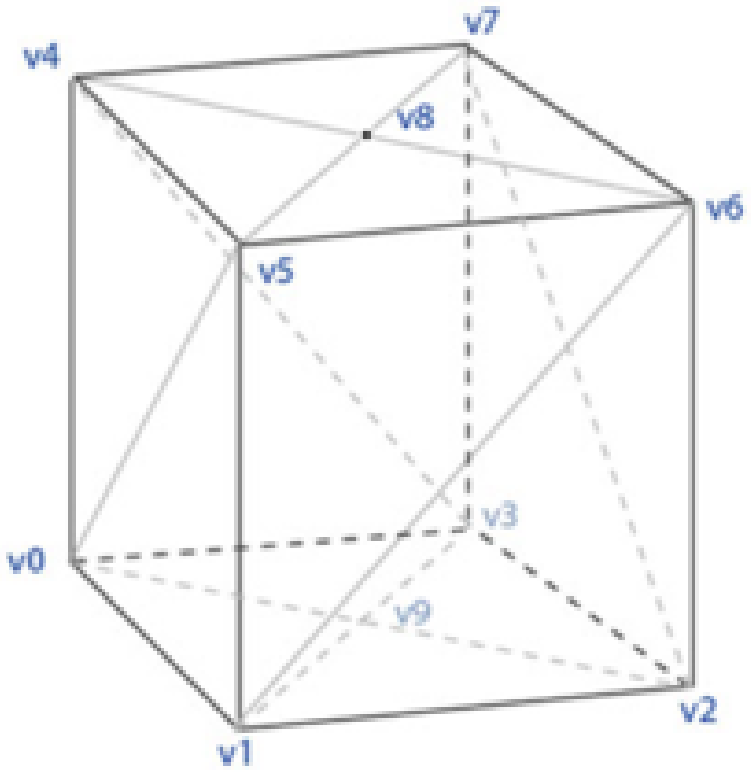
example



Otras representaciones: *Vertex-Vertex Meshes*

Vertex List

v0	0,0,0	v1 v5 v4 v3 v9
v1	1,0,0	v2 v6 v5 v0 v9
v2	1,1,0	v3 v7 v6 v1 v9
v3	0,1,0	v2 v6 v7 v4 v9
v4	0,0,1	v5 v0 v3 v7 v8
v5	1,0,1	v6 v1 v0 v4 v8
v6	1,1,1	v7 v2 v1 v5 v8
v7	0,1,1	v4 v3 v2 v6 v8
v8	.5,.5,1	v4 v5 v6 v7
v9	.5,.5,0	v0 v1 v2 v3



Otras representaciones: *Winged-Edges Mesh*

Winged-Edge Meshes

Face List

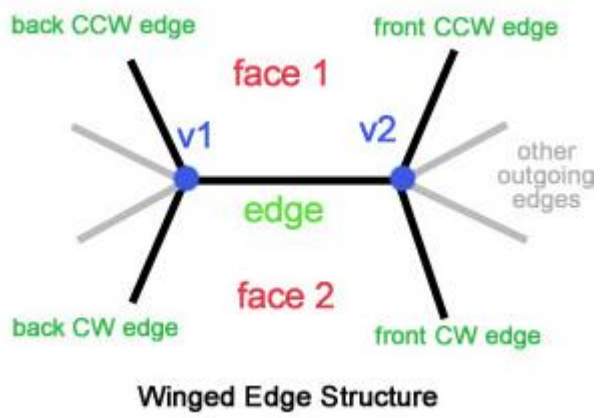
f0	4 8 9
f1	0 10 9
f2	5 10 11
f3	1 12 11
f4	6 12 13
f5	2 14 13
f6	7 14 15
f7	3 8 15
f8	4 16 19
f9	5 17 16
f10	6 18 17
f11	7 19 18
f12	0 23 20
f13	1 20 21
f14	2 21 22
f15	3 22 23

Edge List

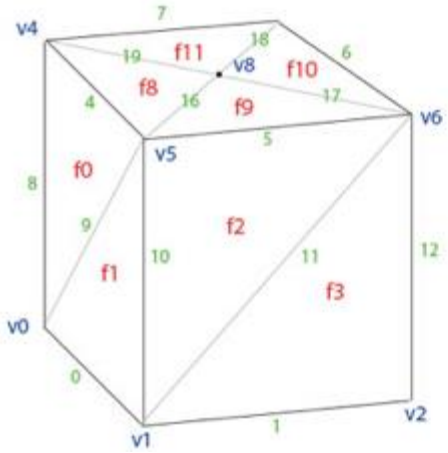
e0	v0 v1	f1 f12	9 23 10 20
e1	v1 v2	f3 f13	11 20 12 21
e2	v2 v3	f5 f14	13 21 14 22
e3	v3 v0	f7 f15	15 22 8 23
e4	v4 v5	f0 f8	19 8 16 9
e5	v5 v6	f2 f9	16 10 17 11
e6	v6 v7	f4 f10	17 12 18 13
e7	v7 v4	f6 f11	18 14 19 15
e8	v0 v4	f7 f0	3 9 7 4
e9	v0 v5	f0 f1	8 0 4 10
e10	v1 v5	f1 f2	0 11 9 5
e11	v1 v6	f2 f3	10 1 5 12
e12	v2 v6	f3 f4	1 13 11 6
e13	v2 v7	f4 f5	12 2 6 14
e14	v3 v7	f5 f6	2 15 13 7
e15	v3 v4	f6 f7	14 3 7 15
e16	v5 v8	f8 f9	4 5 19 17
e17	v6 v8	f9 f10	5 6 16 18
e18	v7 v8	f10 f11	6 7 17 19
e19	v4 v8	f11 f8	7 4 18 16
e20	v1 v9	f12 f13	0 1 23 21
e21	v2 v9	f13 f14	1 2 20 22
e22	v3 v9	f14 f15	2 3 21 23
e23	v0 v9	f15 f12	3 0 22 20

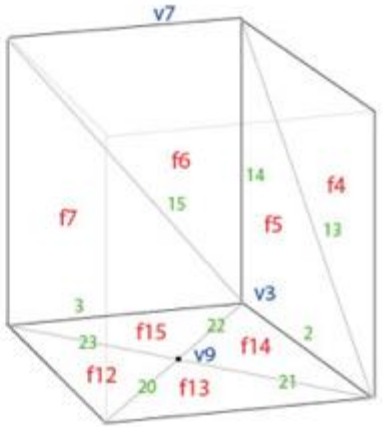
Vertex List

v0	0,0,0	8 9 0 23 3
v1	1,0,0	10 11 1 20 0
v2	1,1,0	12 13 2 21 1
v3	0,1,0	14 15 3 22 2
v4	0,0,1	8 15 7 19 4
v5	1,0,1	10 9 4 16 5
v6	1,1,1	12 11 5 17 6
v7	0,1,1	14 13 6 18 7
v8	.5,.5,0	16 17 18 19
v9	.5,.5,1	20 21 22 23

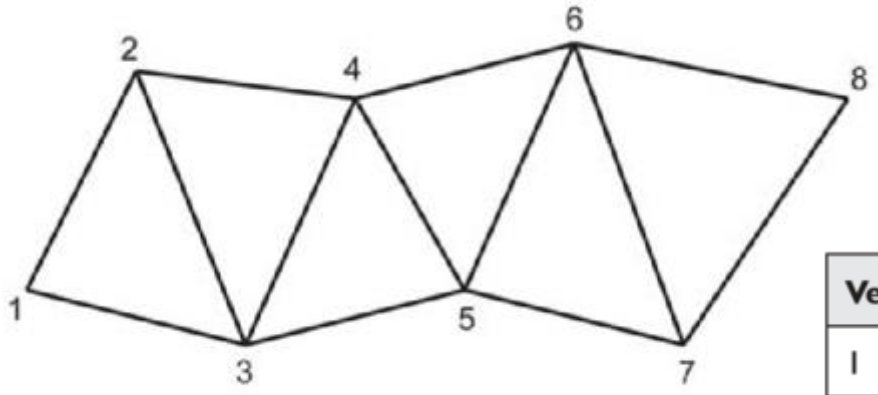


Winged Edge Structure





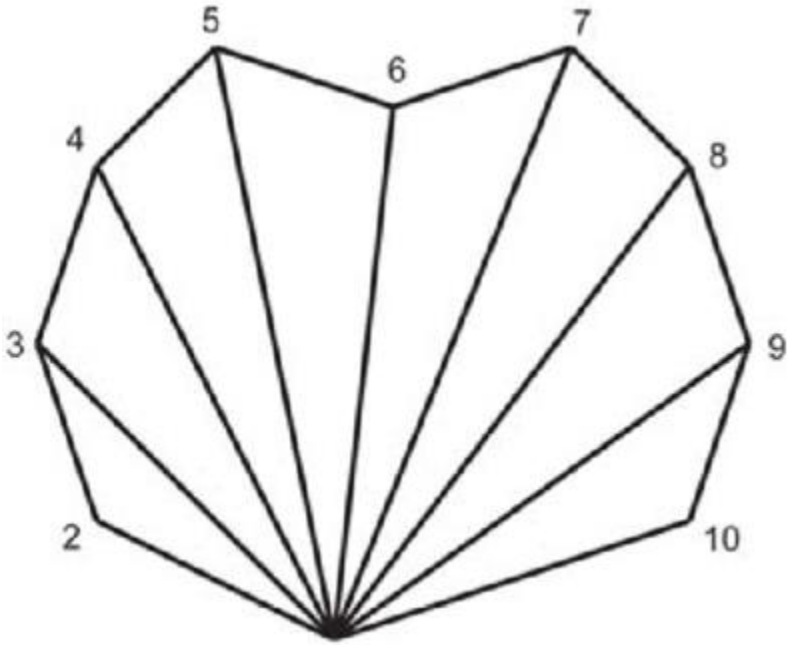
Otras representaciones: *Triangle strip*



Vertex order: 1, 2, 3, 4, 5, 6, 7, 8

Vertex	Triangle	Vertex Ordering
1	(none)	
2	(none)	
3	1,2,3	Clockwise
4	2,3,4	Counterclockwise
5	3,4,5	Clockwise
6	4,5,6	Counterclockwise
7	5,6,7	Clockwise
8	6,7,8	Counterclockwise

Otras representaciones: *Triangle fan*



Operaciones: Subdivisión y Simplificación

Subdivisión

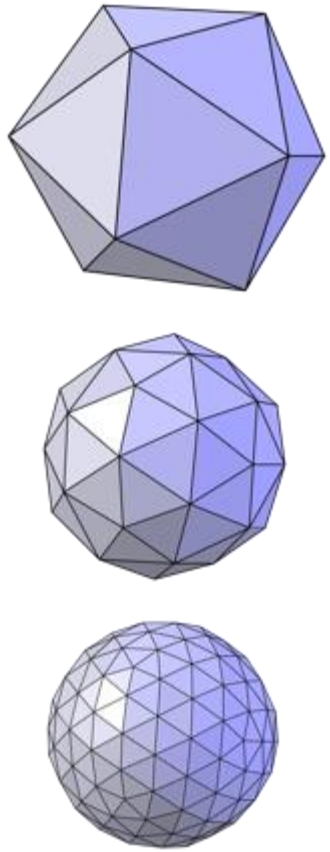
Un triángulo es reemplazado por otros más pequeños. Puede ser coplanar. Usada para suavizar un objeto 3D.

El número de triángulos aumenta, haciendo que la malla sea más compleja.

Simplificación

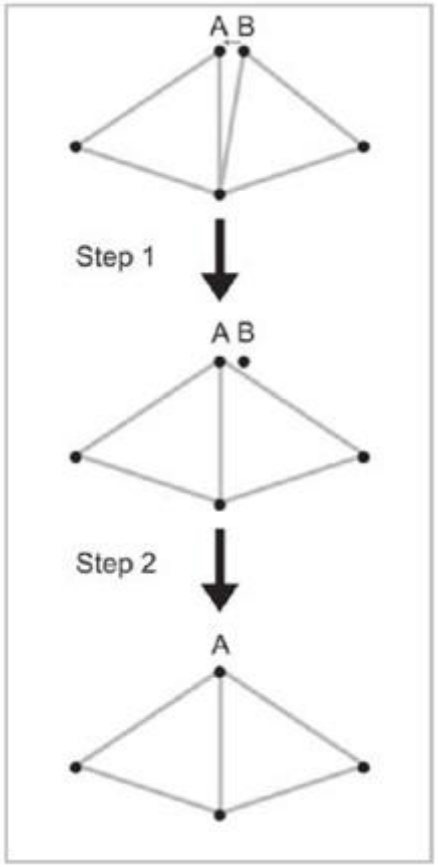
Una malla compleja es reemplazada por una mas sencilla, similar topológicamente.

Si se simplifica demasiado, se pierde la forma original.

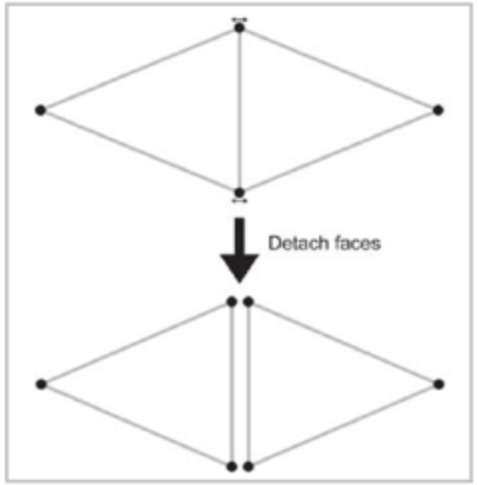


Operaciones:

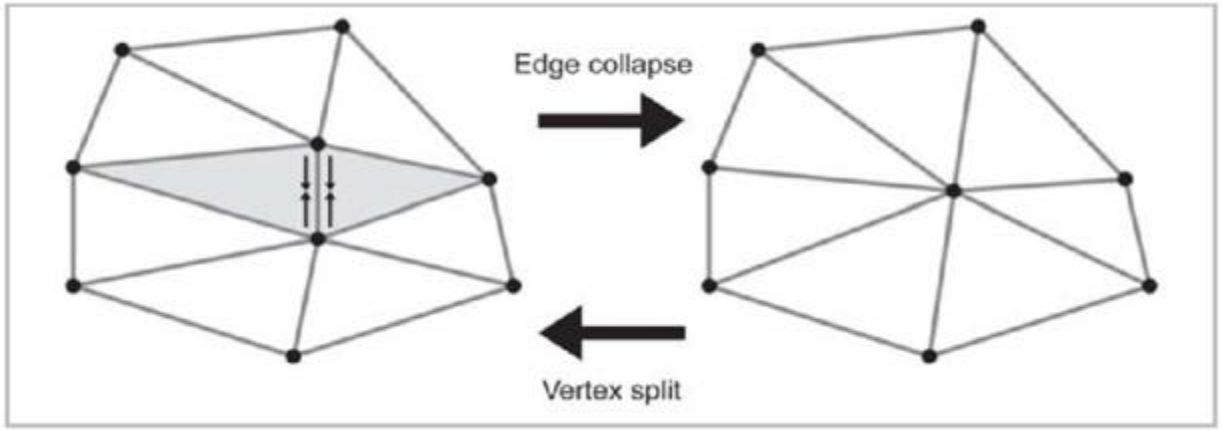
Soldar vértices:

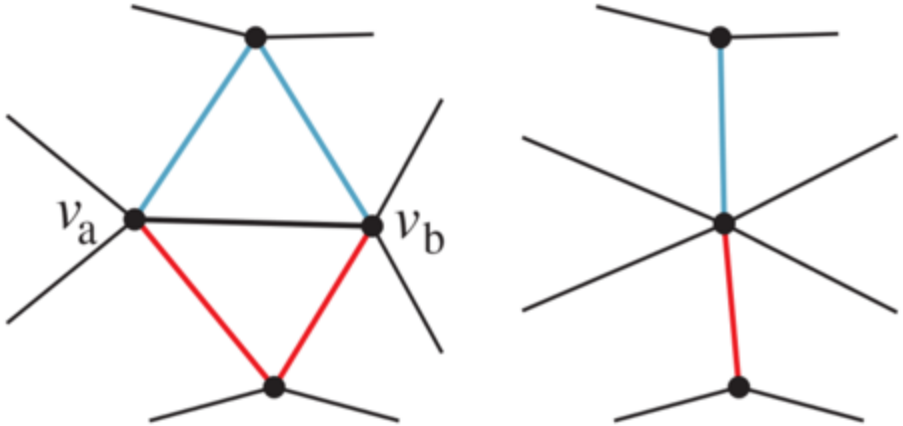


Separar caras :

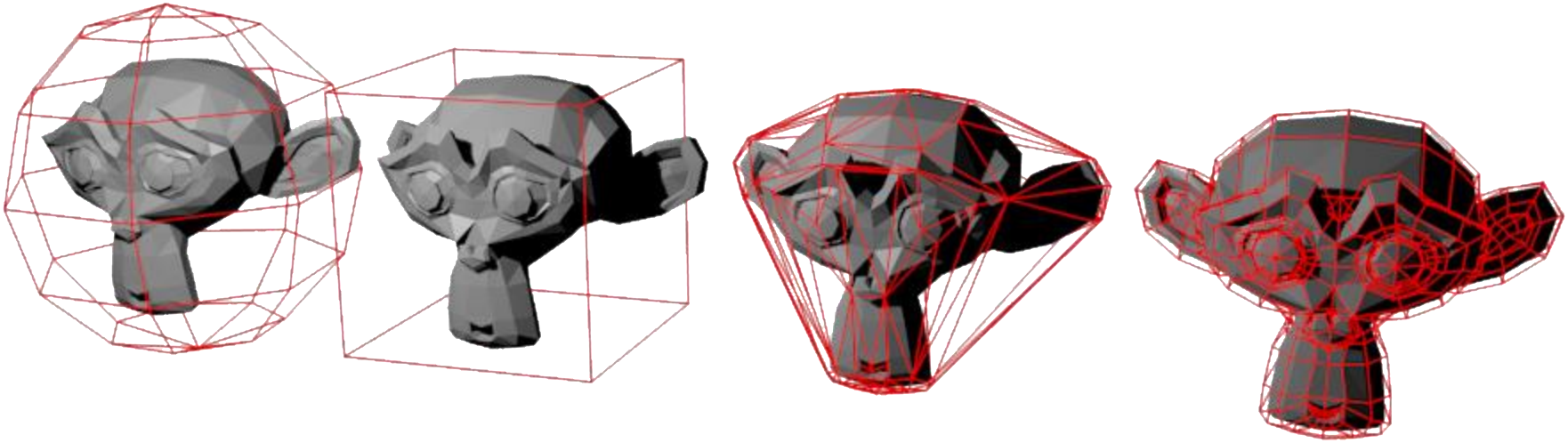


Operaciones: colapsar aristas

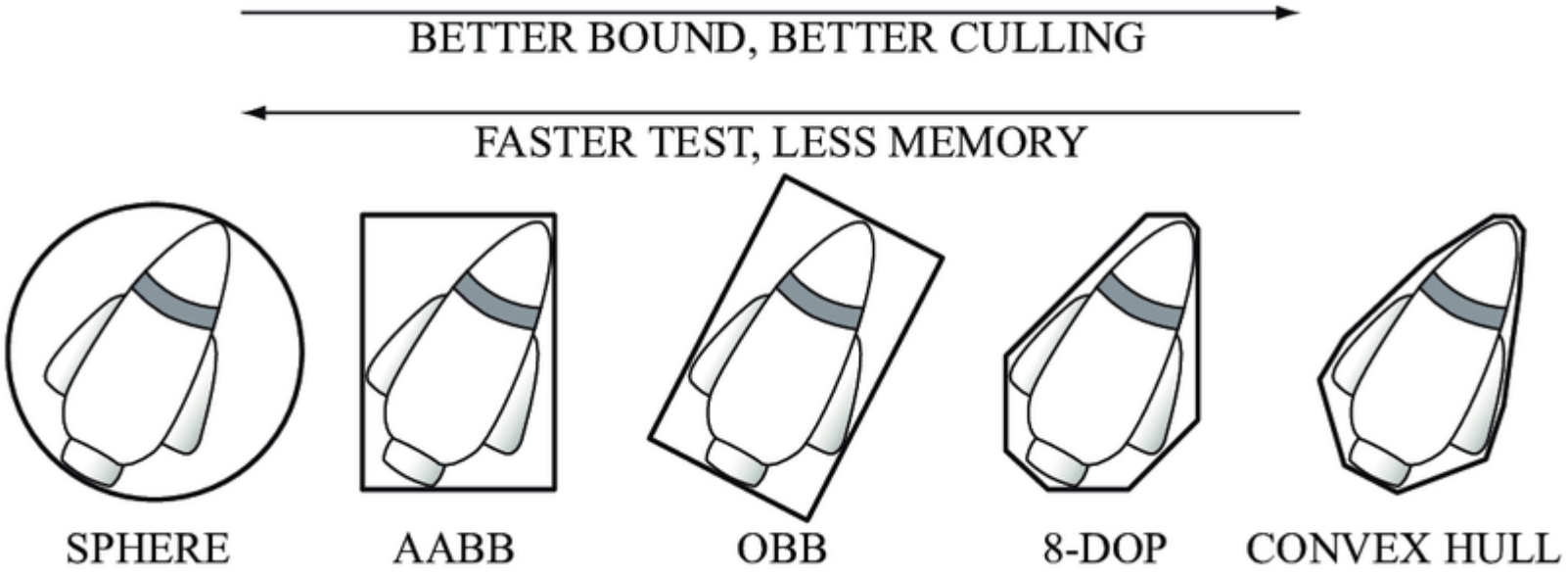




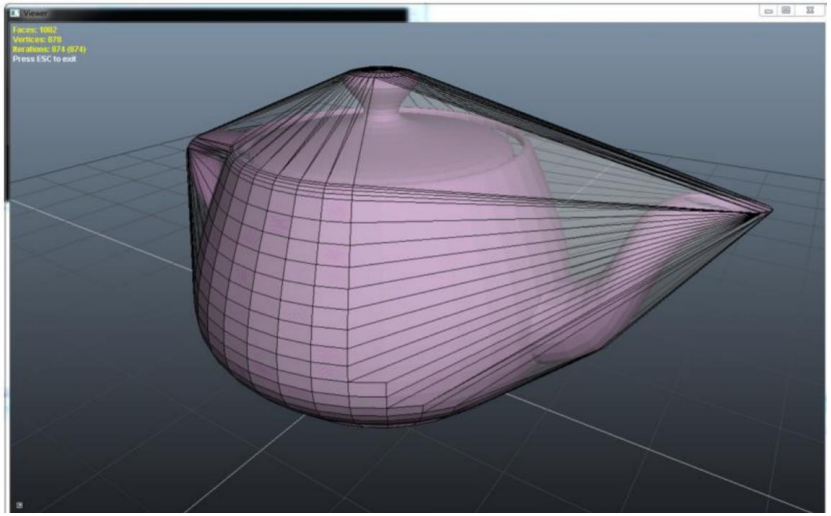
Bounding volumes



Bounding volumes



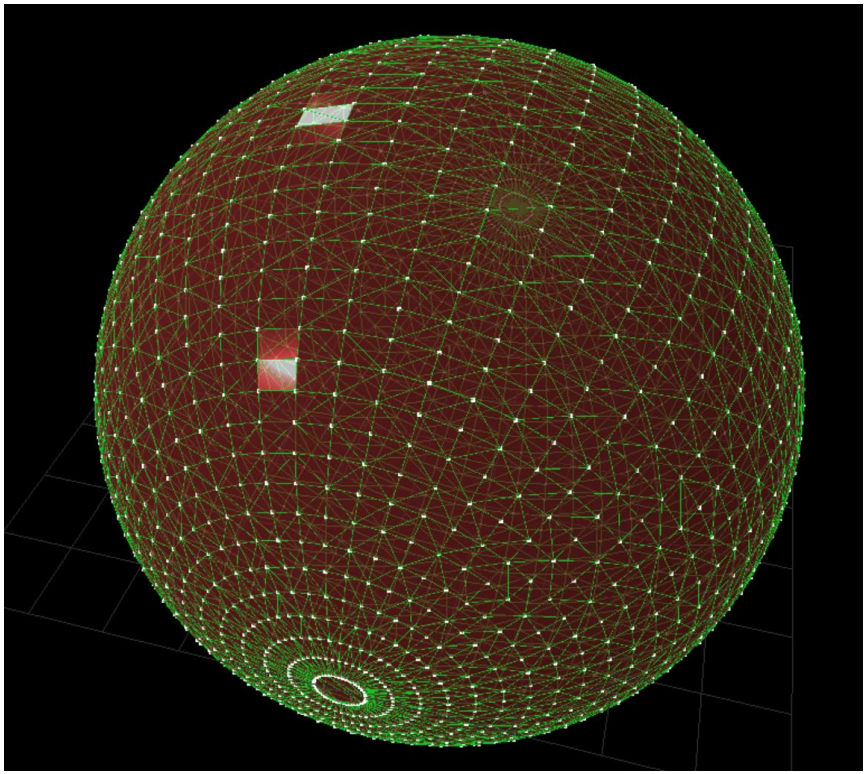
Casco convexo



Dados N puntos, el casco convexo es el conjunto mínimo de polígonos, que contienen todos los puntos en su interior.

Se trata de un poliedro que envuelve de la manera más óptima un conjunto de puntos.

Imagen tomada de: link



Casco convexo

QuickHull: Algoritmo que permite generar, a partir de una nube de puntos, las diferentes caras que conforman su casco convexo.

Permite generar sólidos, siempre y cuando la nube de puntos siga una forma convexa.

QuickHull en Three.js

El algoritmo está implementado en la librería y puede utilizarse para generar nuevos sólidos, o para analizar sólidos existentes. Es posible utilizarlo rápidamente mediante ConvexGeometry (ejemplo), su uso es:

Agregar:

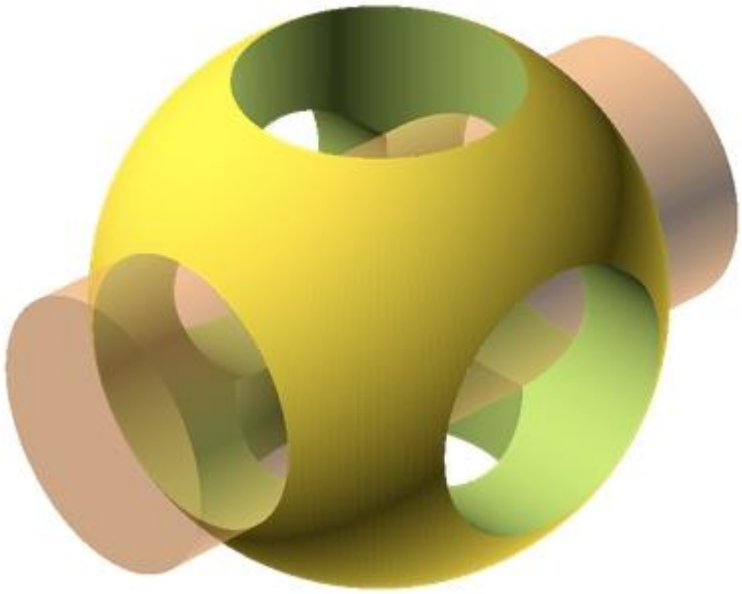
```
<script src="/examples/js/Math/QuickHull.js"></script>
<script src="/examples/js/geometries/ConvexGeometry.js"></script>
```

Dada una geometría, compuesta por puntos (geoEsfera):

```
var esfera = new THREE.Points( geoEsfera, matPunto );
var geometry = new THREE.ConvexBufferGeometry( points );
var matMalla = new THREE.MeshStandardMaterial({color: 0xAA3333)};
var mesh = new THREE.Mesh( geometry, matMalla );
```

Ejercicio

Dados los puntos calculados para la máscara creada previamente, obtener su casco convexo, utilizando ConvexGeometry.



GEOMETRÍA CONSTRUCTIVA DE SÓLIDOS (CSG)

Geometría Constructiva de Sólidos

Es posible crear objetos complejos, a partir de la combinación de sólidos más simples, utilizando operaciones booleanas.

Las diferentes operaciones se deben hacer de forma jerárquica y organizada.

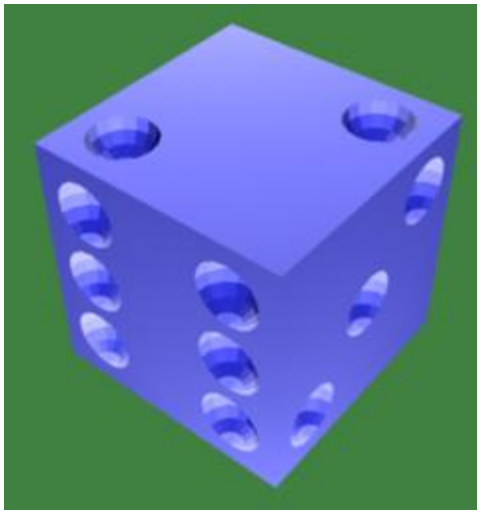
Geometría Constructiva de Sólidos

Formada por:

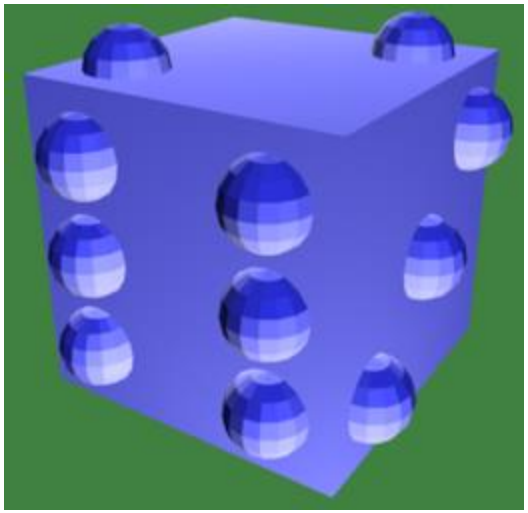
- Sólidos simples (primitivas):
  - Cubos, esferas, toroides, entre otros, disponibles en la librería.
- Operaciones booleanas:
  - Unión.
  - Intersección.
  - Diferencia.

Operaciones booleanas

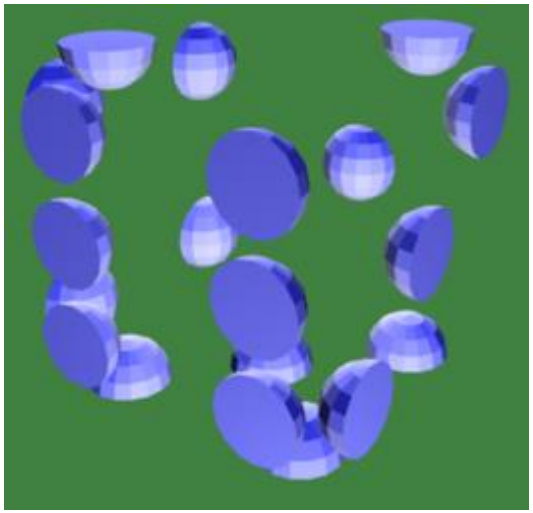
Diferencia



Unión



Intersección



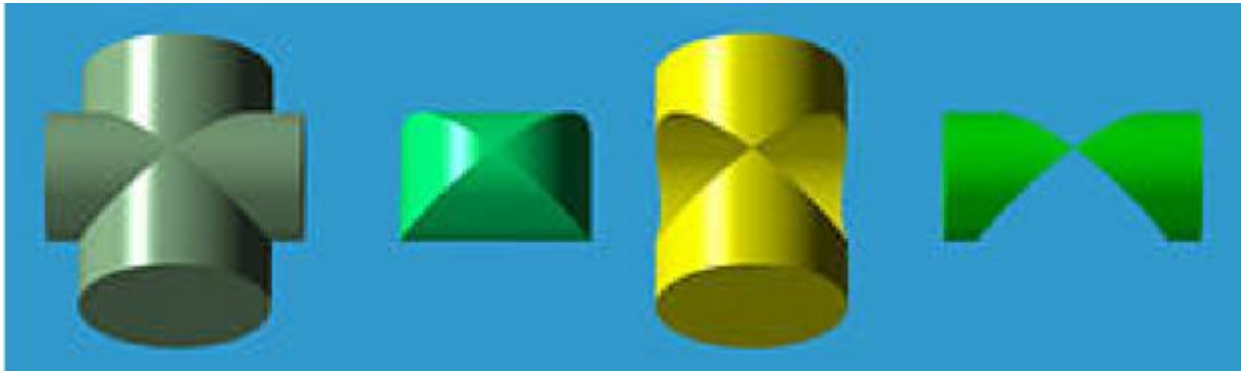
Formalmente

Dados dos conjuntos A y B:

Unión: Todos los puntos de A y B.

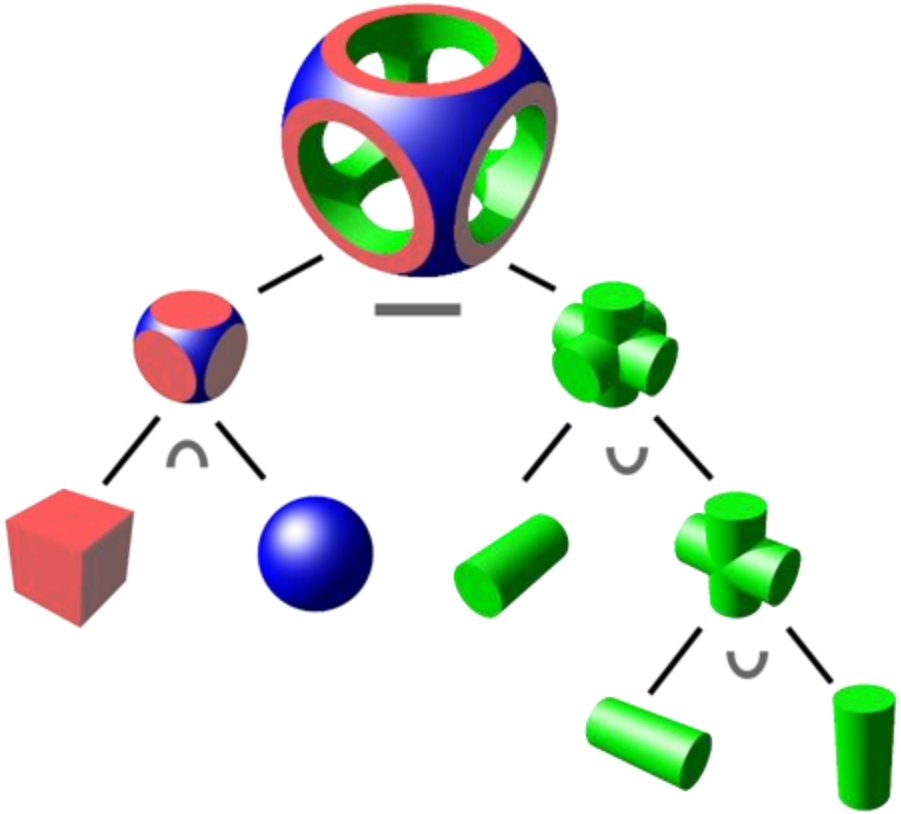
Intersección: Todos los puntos que están en A y en B simultáneamente.

Diferencia (A-B): Todos los puntos de en A, pero no en B.



CSG

Se representa internamente como un árbol jerárquico de operaciones:



csg.js y THREE.CSG.js

csg.js es una librería que implementa operaciones de CSG en mallas, usando árboles BSP.

Para poder implementar los algoritmos de CSG de forma rápida y práctica, mediante las operaciones booleanas, se utiliza THREE.CSG.js, que funciona como un puente entre las librerías csg.js y THREE.js

Es necesario agregar las librerías:

<script src="csg.js"**></script>**

<script src="THREE.CSG.js"**></script>**

Uso con THREE.CSG.js

1. Crear las mallas:

```
var cube = new THREE.Mesh( boxGeometry, material1 );
var sphere = new THREE.Mesh( sphereGeometry, material2 );
```

2. Transformarlas a la posición deseada:

```
sphere.translateX( .5 );
sphere.translateY( .5 );
```

3. Convertir a CSG:

```
var boxCSG = THREE.CSG.fromMesh( cube );
var sphereCSG = THREE.CSG.fromMesh( sphere );
```

4. Realizar las operaciones booleanas:

- `var result1 = boxCSG.union( sphereCSG );`
- `var result2 = boxCSG.subtract( sphereCSG );`
- `var result3 = boxCSG.intersect( sphereCSG );`

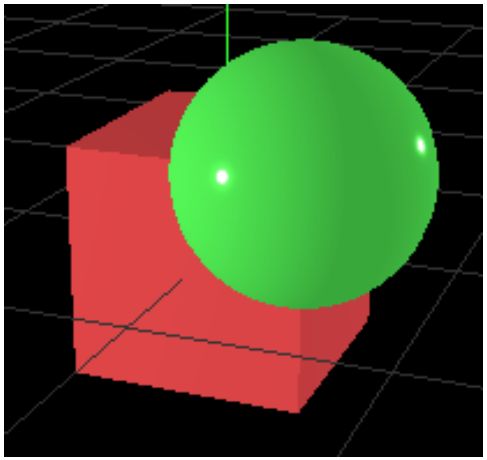
5. Volver a convertir a THREE Mesh:

```
cube = THREE.CSG.toMesh( result1 );
```

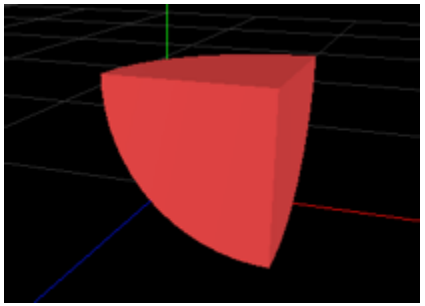
Ejemplo

Ejemplo

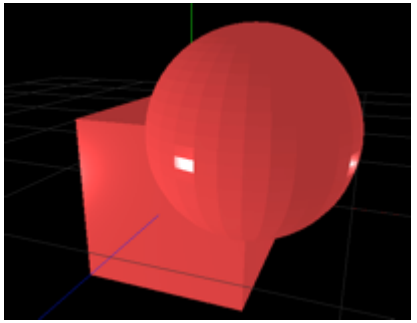
ORIGINAL



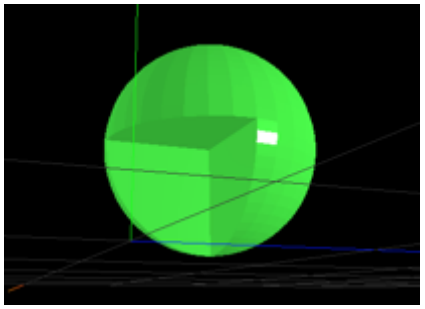
Intersección



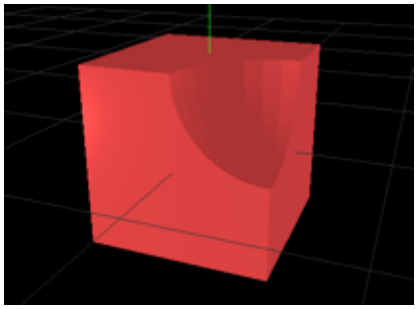
Unión



Esfera menos caja



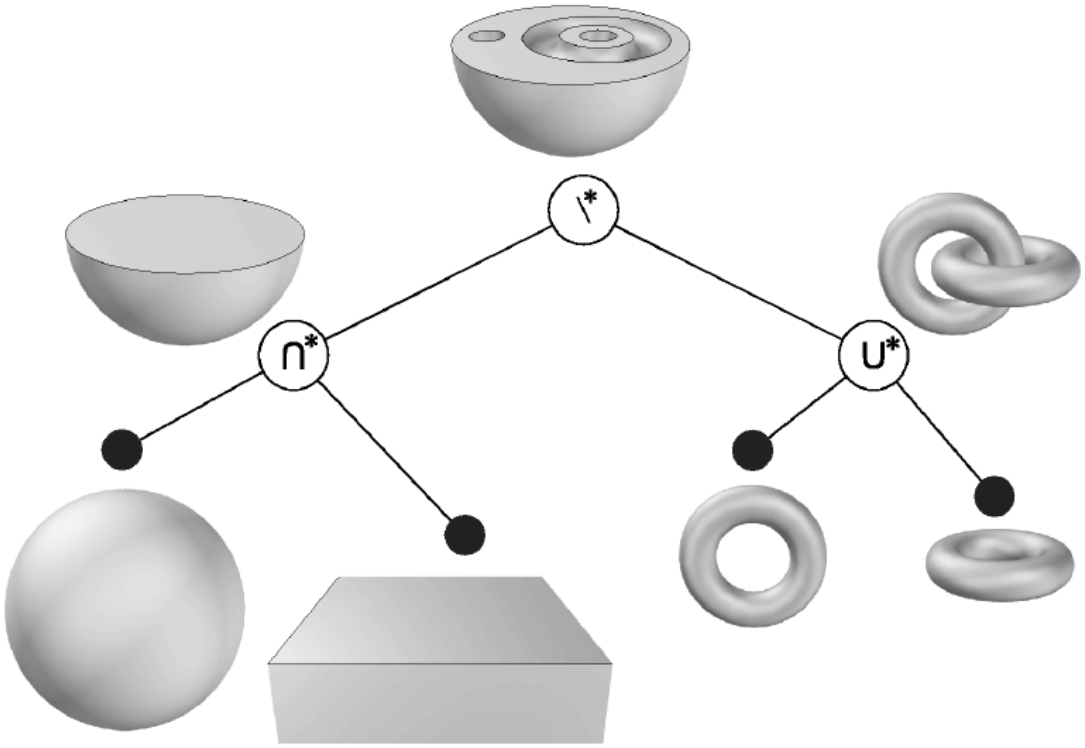
Caja menos esfera



Código disponible en el aula virtual.

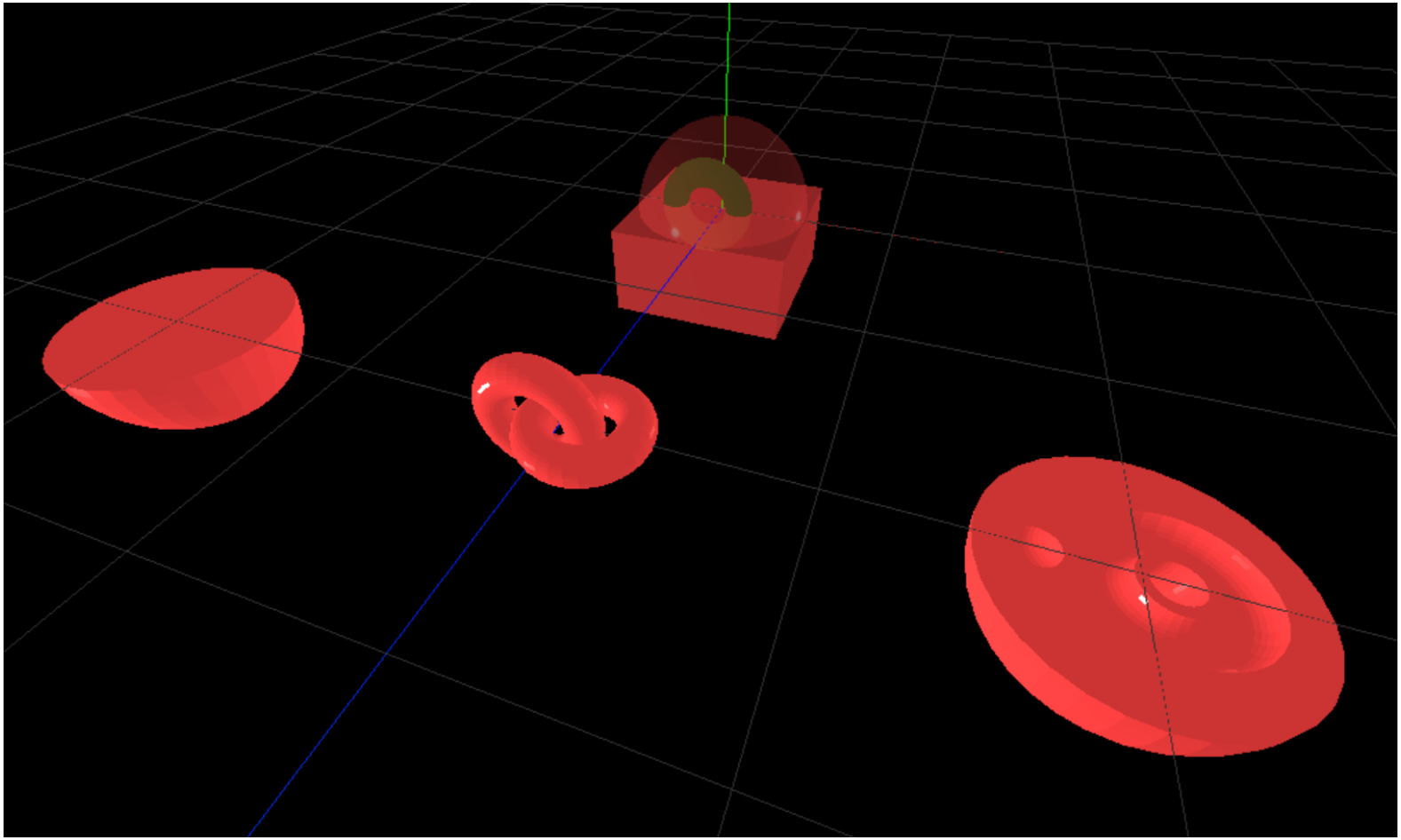
Ejercicio

1. Hacer la figura, usando las librerías csg.js, THREE.CSG.js y three.js



Tomado de: https://www.researchgate.net/figure/Example-of-a-CSG-tree_fig1_226357801

Resultado



Bibliografía

Dunn, F. y Parberry, I. (2002). Chapter 14 - Triangle Meshes en: ***3D Math Primer for Graphics and Game Development***. Wordware Publishing, Inc.

THREE.CSG. Constructive Solid Geometry with csg.js. Recuperado de: http://learningthreejs.com/blog/2011/12/10/constructive-solid-geometry-with-csg-js/ el 21 de marzo de 2018.

Hughes, J et al. (2014). Chapter 8 – A simple way to describe shape in 2D and 3D en: ***Computer Graphics: Principles and Practice***. 3rd Ed. Addison-Wesley.

Rueda, A. (2015). ***Construcción de sólidos en 3D***. Pontificia Universidad Javeriana.

Bibliografía

Gregorius, Dick. Implementing QuickHull. Valve Software.

Marschner, Steve (2014). ***Surfaces and solids***. CS4620 Lecture 19. Cornell University.

Kocabas, H. Computer Aided Design. Tomado de: https://slideplayer.com/slide/10394214/