Introducción al diseño 3D con OpenSCAD

Tabla de contenidos

Introducción

Este material ha sido elaborado para el grupo de trabajo **Aplicaciones al aula de la impresión 3D** del IES Celia Viñas (Almería). Se trata de un material que intenta dar a concer al alumnado el software de modelado OpenSCAD [https://www.openscad.org] y sus primitivas geométricas, para que puedan realizar algunos modelos 3D sencillos que luego pueden ser impresos en la impresora 3D del instituto.

¿Qué es OpenSCAD?

OpenSCAD [https://www.openscad.org] es una aplicación de software libre para se utiliza para crear objetos sólidos de **CAD** (Computer-Aided Design - Diseño Asistido por Computador). No es un editor interactivo sino un compilador 3D basado en un lenguaje de descripción textual. Un documento de OpenSCAD especifica **primitivas geométricas** (esfera, cubo, cilindro, etc.) y define como pueden ser modificadas y manipuladas para reproducir un modelo 3D. OpenSCAD está disponible para Windows, Linux y OS X.

Fuente: Wikipedia [https://es.wikipedia.org/wiki/OpenSCAD]

Primitivas geométricas

cube

```
cube(size = [x,y,z], center = true/false);
cube(size =  x ,     center = true/false);
```

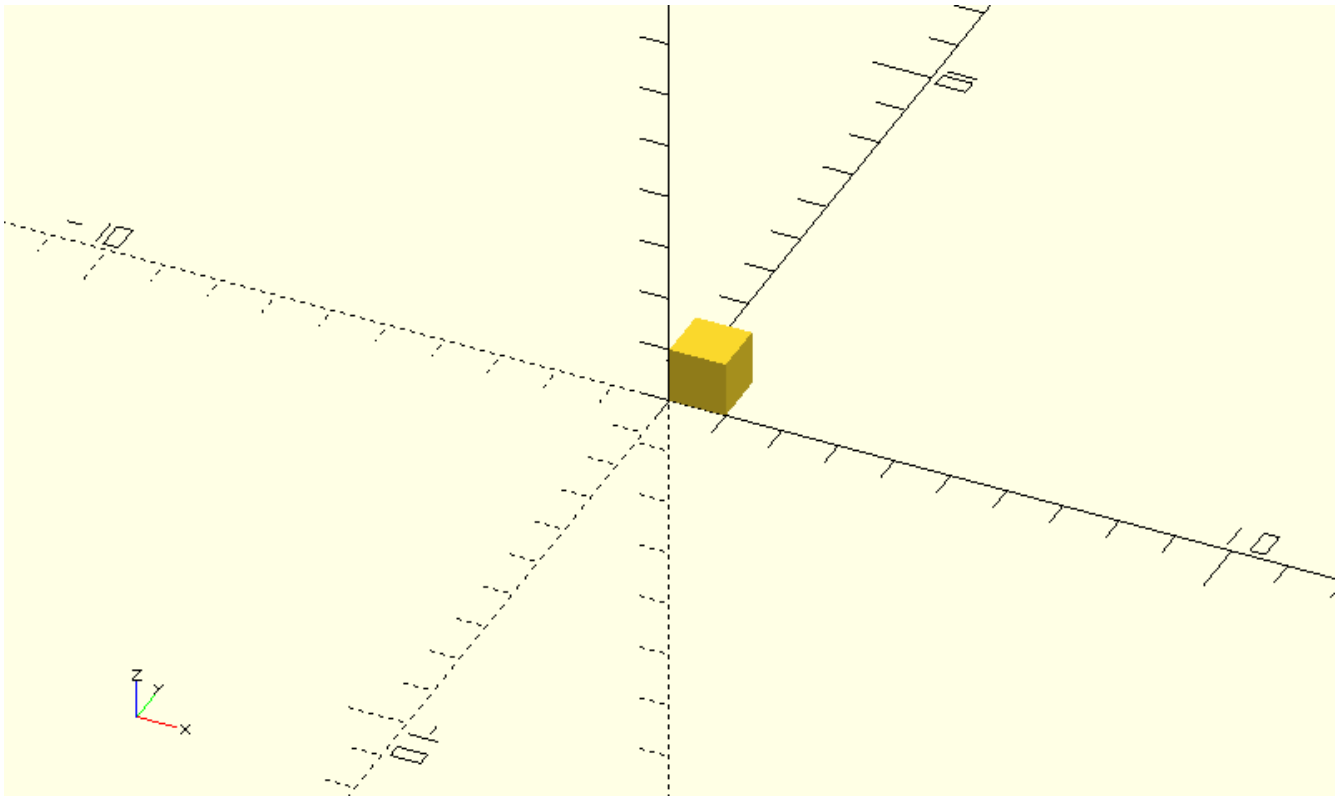
Esta primitiva nos permite crear un cubo y podemos indicarle dos parámetros de entrada:

- `size`: nos permite indicar las dimensiones de la arista.
- `center`: nos permite centrar el cubo en el origen de coordenadas.

Cuando utilizamos `cube` sin pasarle ningún parámetro de entrada, crea un cubo con una arista de 1mm.

Ejemplo:

```
cube();
```

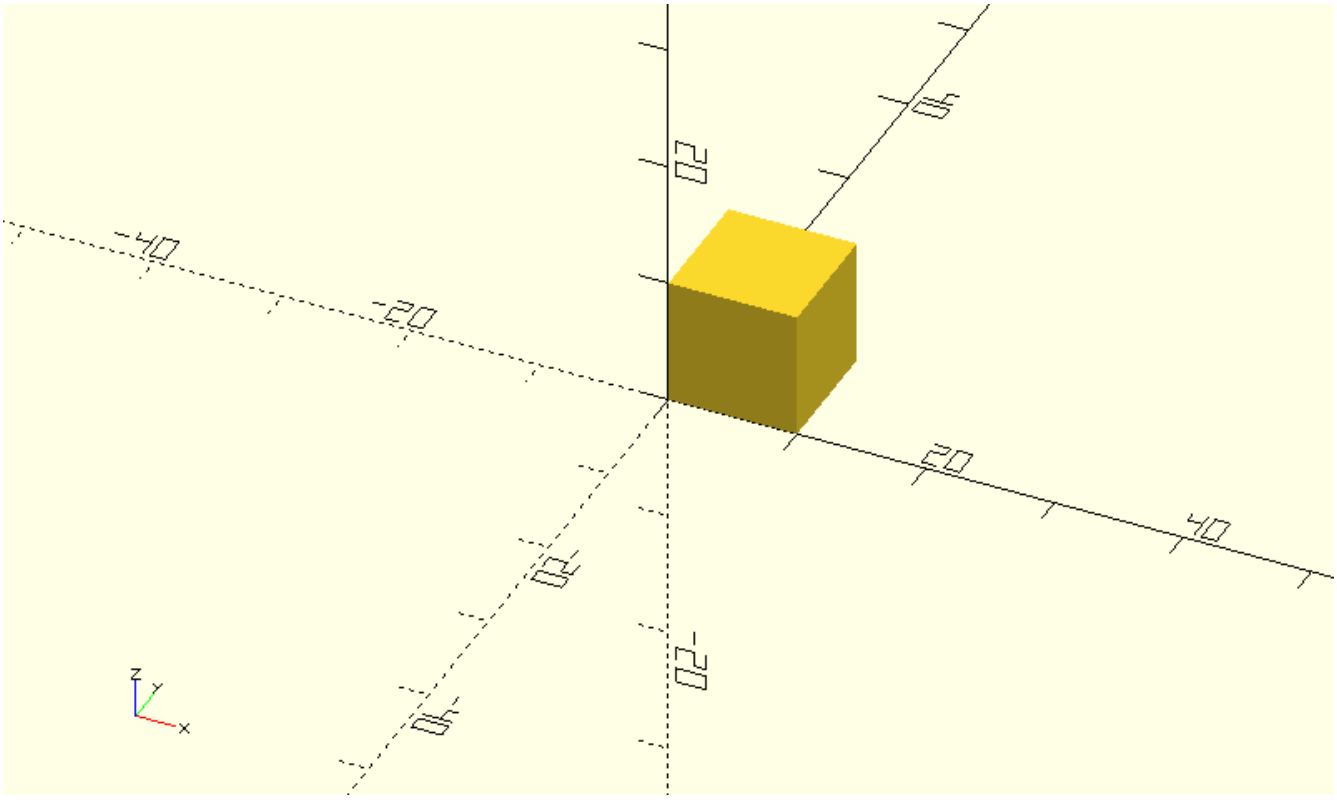


Ejemplo:

```
cube(10);
```

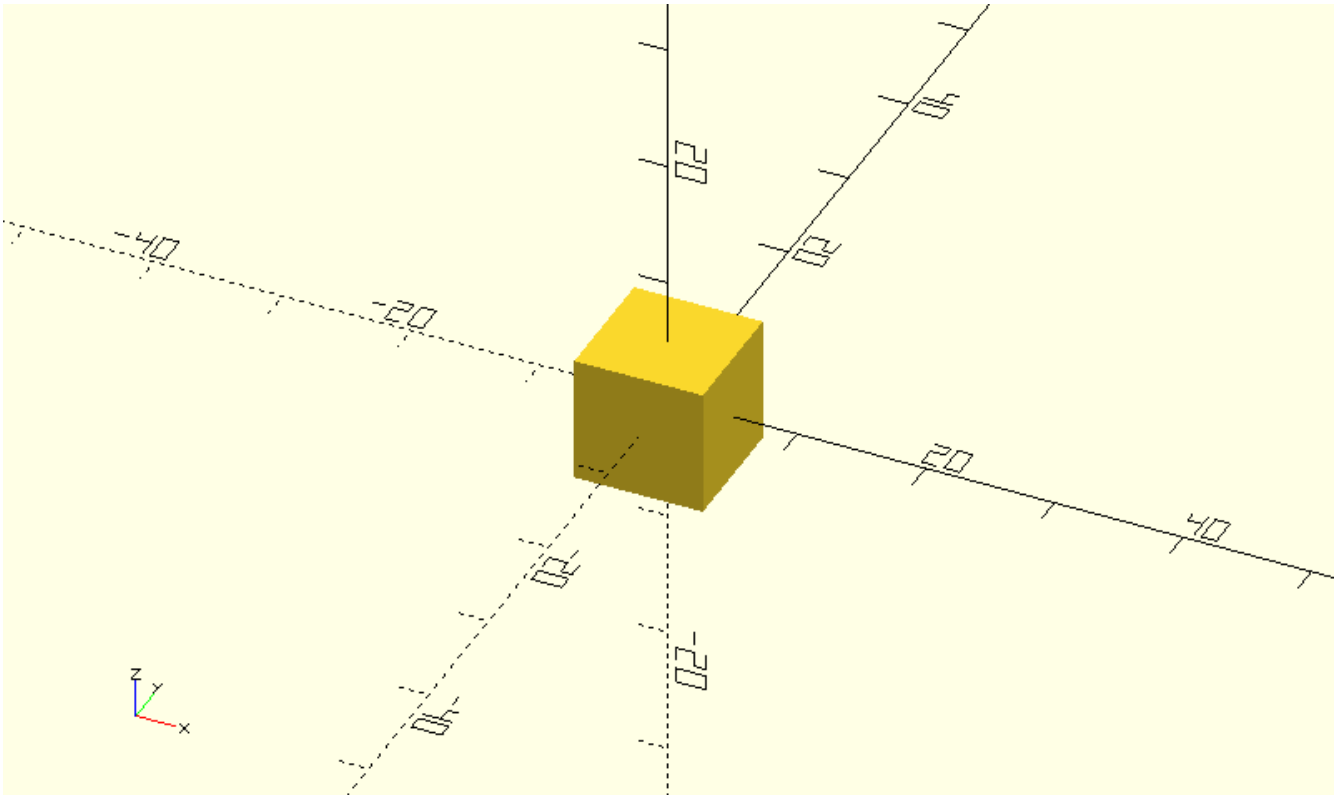
El ejemplo anterior creará un cubo de 10mm de arista. Como sólo hemos indicado un valor, este será el valor de la arista en `x`, `y` y `z`. Por lo tanto, `cube(10)`, es equivalente a:

```
cube([10, 10, 10]);
```



Ahora vamos a ver lo que pasa si pasamos el valor `true` como segundo parámetro.

```
cube([10, 10, 10], center = true);
```



sphere

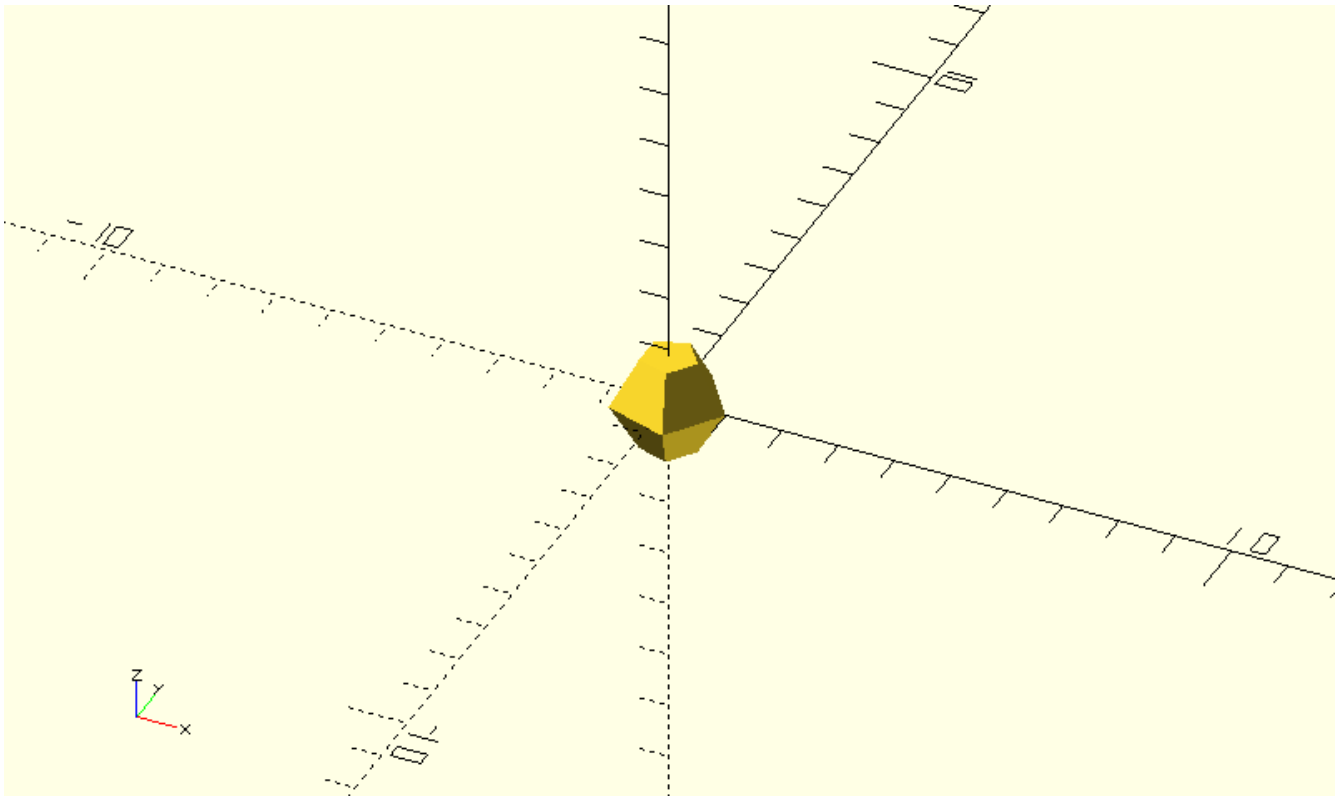
```
sphere(radius | d=diameter)
```

Crea una esfera centrada en el origen de coordenadas. Podemos indicar el radio de la esfera con el parámetro del `r` o el diámetro con el parámetro `d`.

Cuando utilizamos `sphere` sin pasarle ningún parámetro de entrada, crea una esfera con un radio de 1mm.

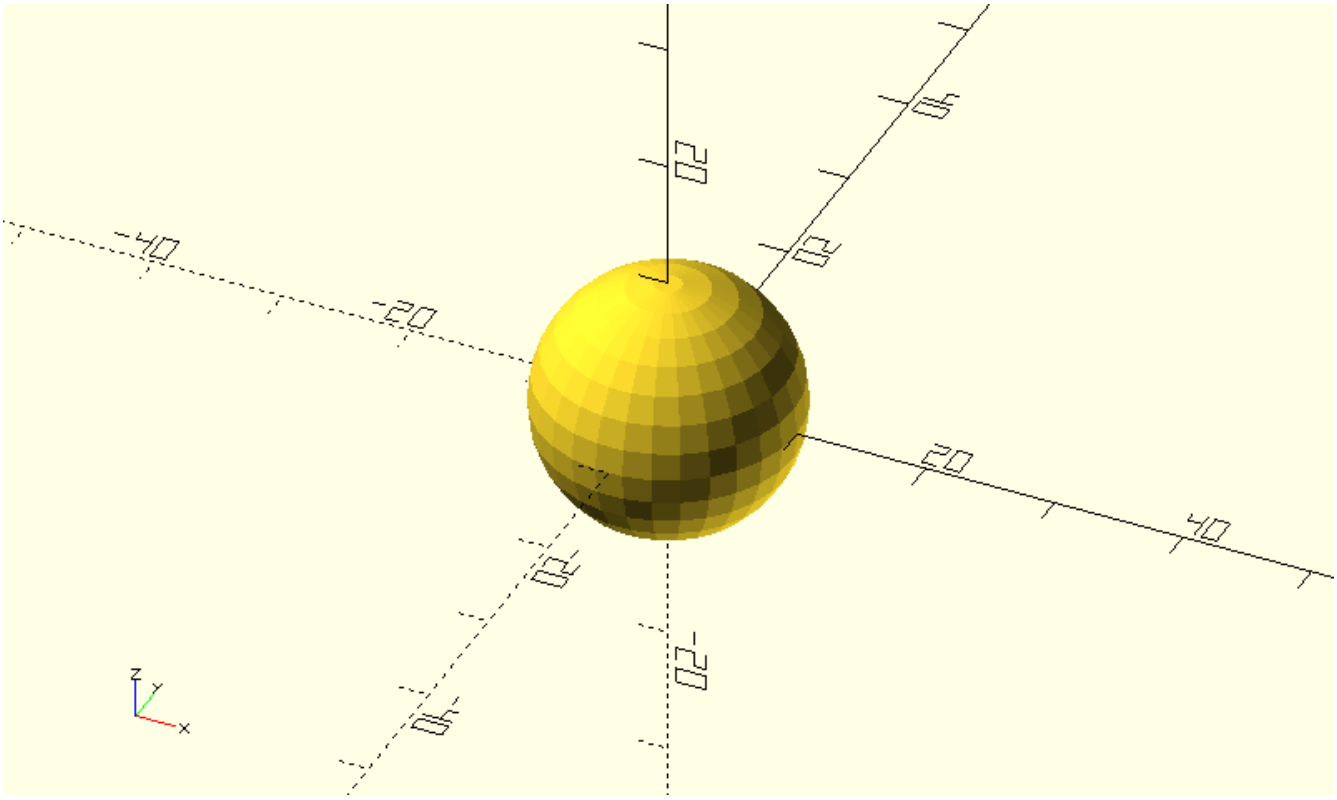
Ejemplo:

```
sphere();
```



Ejemplo:

```
sphere(10);
```

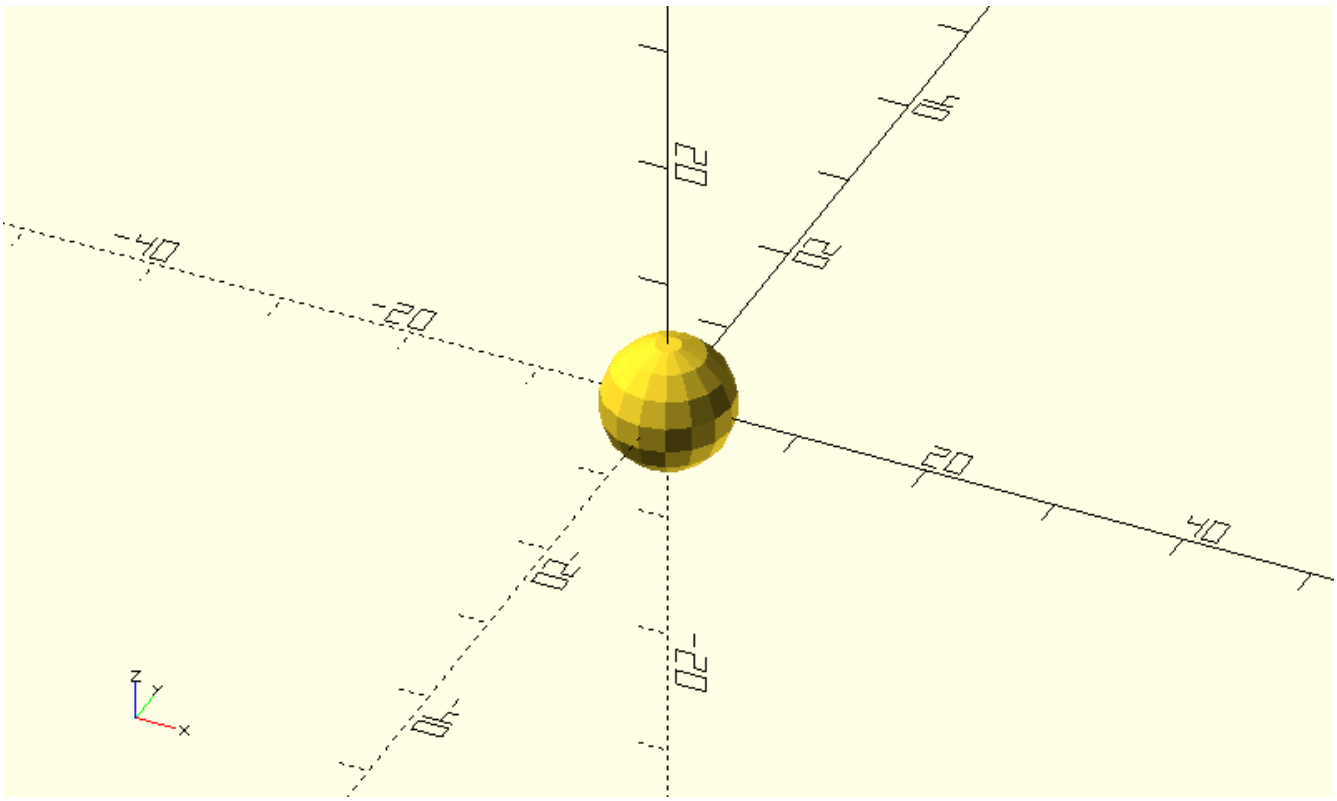


En este ejemplo hemos creado una esfera con un **radio** de 10mm. Es equivalente a:

```
sphere(r=10);
```

Ejemplo:

```
sphere(d=10);
```



En este ejemplo hemos creado una esfera con un **diámetro** de 10mm.

Transformaciones geométricas

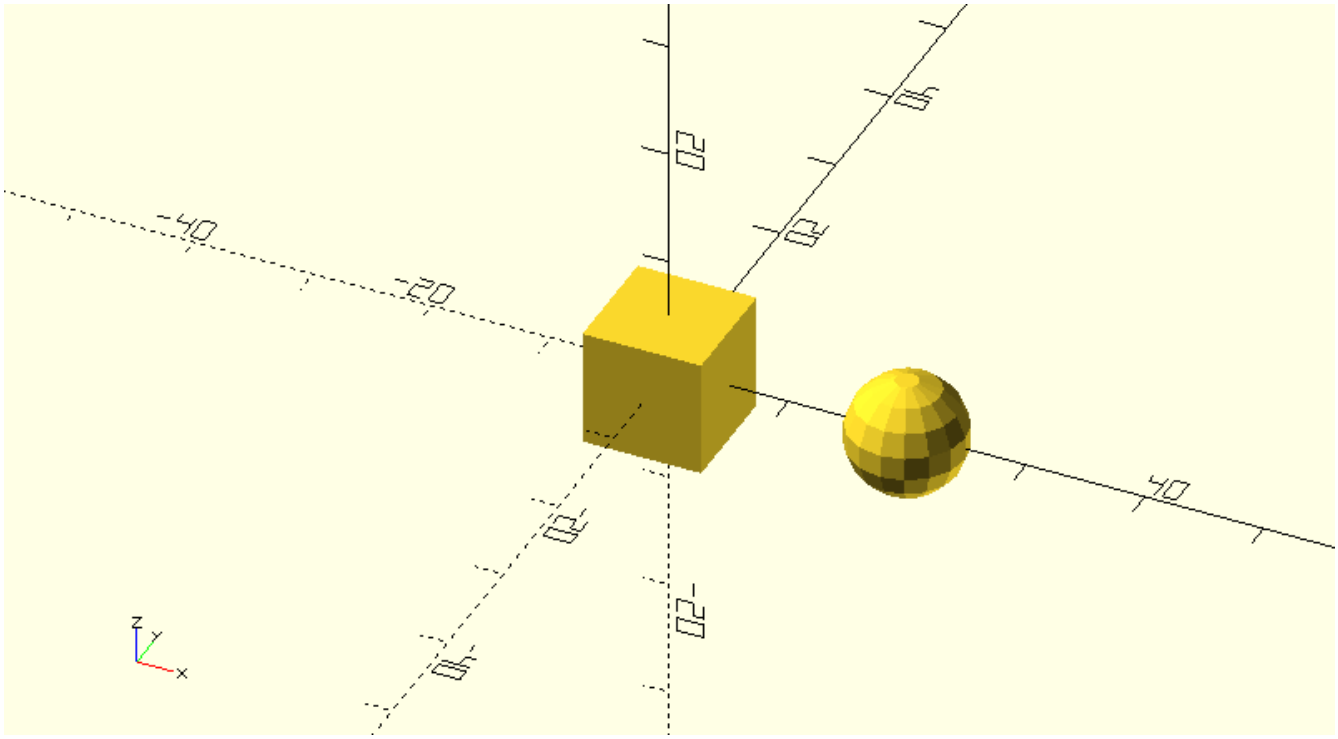
translate

```
translate(v = [x, y, z]) { ... }
```

Esta función nos permite trasladar (mover) el elemento que aparece a continuación.

Ejemplo:

```
cube(10, center = true);
translate([20,0,0])
    sphere(5, center = true);
```



En este ejemplo podemos ver como la esfera se ha desplazado 20mm en el eje de la x.

Si quisiéramos trasladar más de un elemento deberíamos utilizar las llaves ({, }) para encerrar el grupo de elementos que queremos trasladar. Por ejemplo, el siguiente código sólo realizaría la traslación de la esfera, porque es el elemento que aparece a continuación de la función `translate`.

```
translate([20,0,0])
    sphere(5, center = true);
    cube(10, center = true);
```

Si quisiéramos trasladar la esfera y el cubo deberíamos hacer:

```
translate([20,0,0]) {
    sphere(5, center = true);
    cube(10, center = true);
}
```

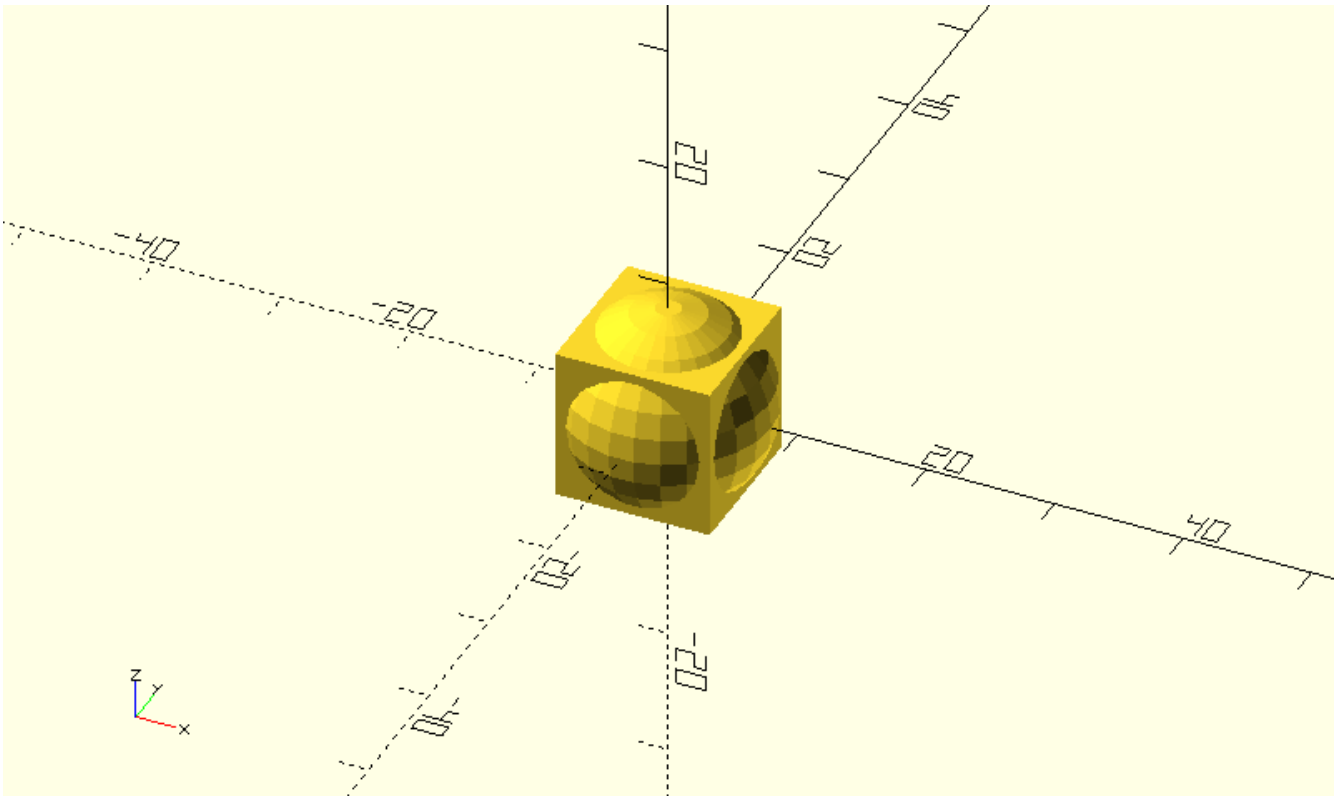
Operaciones booleanas

union

Esta operación realiza la unión de dos elementos. Es equivalente a realizar la operación booleana `or`.

Ejemplo:

```
union() {
    cube(12, center = true);
    sphere(8);
}
```



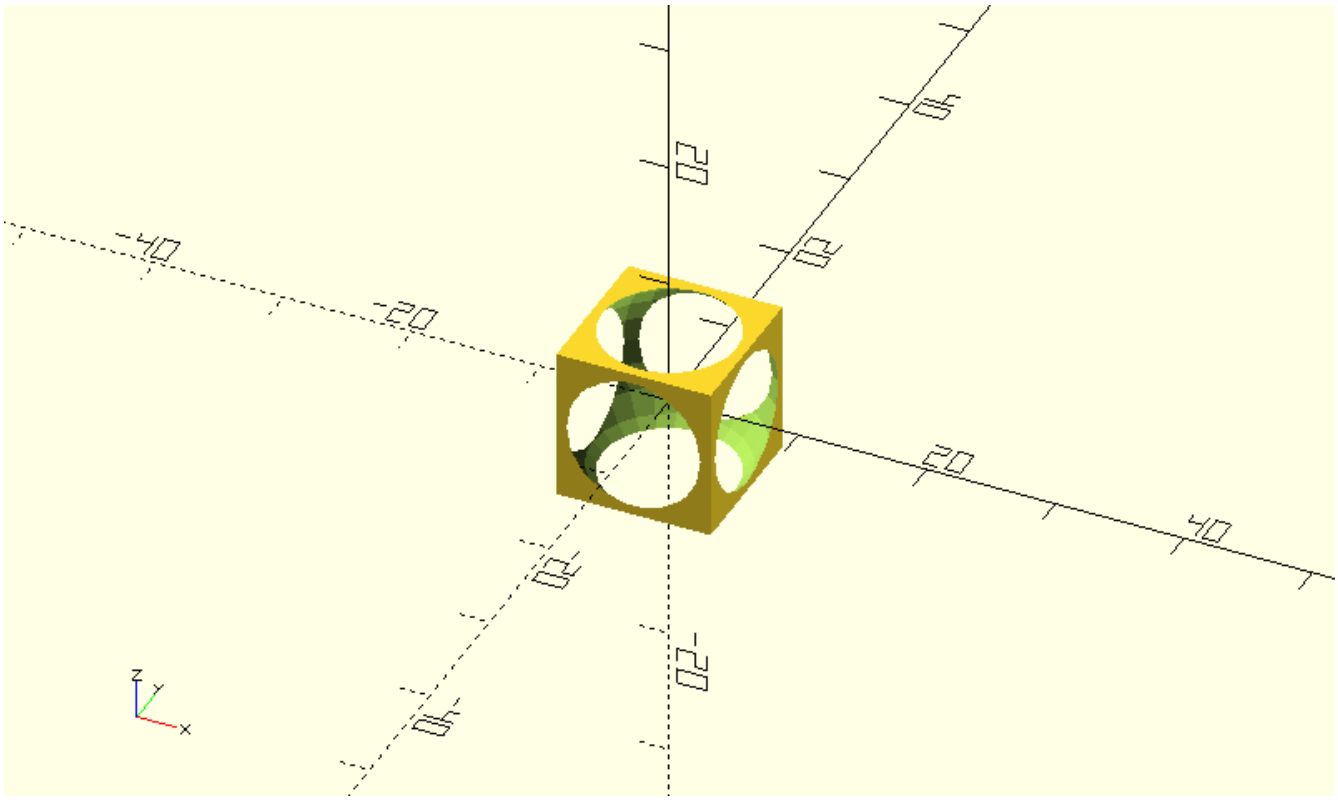
difference

Esta operación realiza la diferencia entre dos elementos. Es equivalente a realizar la operación booleana `and not`.

Ejemplo:

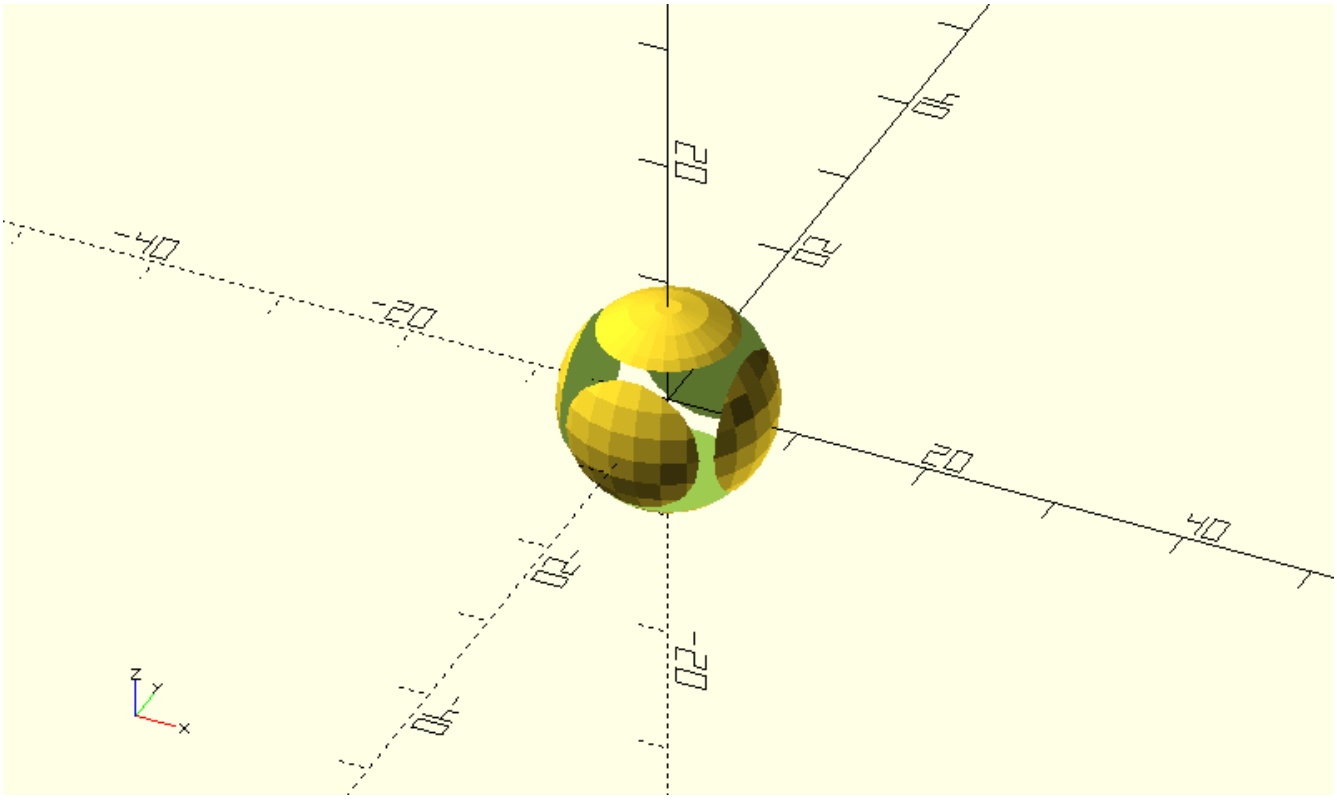
En este ejemplo, al cubo le estamos restando la esfera, es decir, estamos realizando la operación booleana *cube* `and not` *sphere*.

```
difference() {
    cube(12, center = true);
    sphere(8);
}
```



Sin embargo en este ejemplo, a la esfera le estamos restando el cubo, es decir, estamos realizando la operación booleana *sphere* `and not` *cube*.

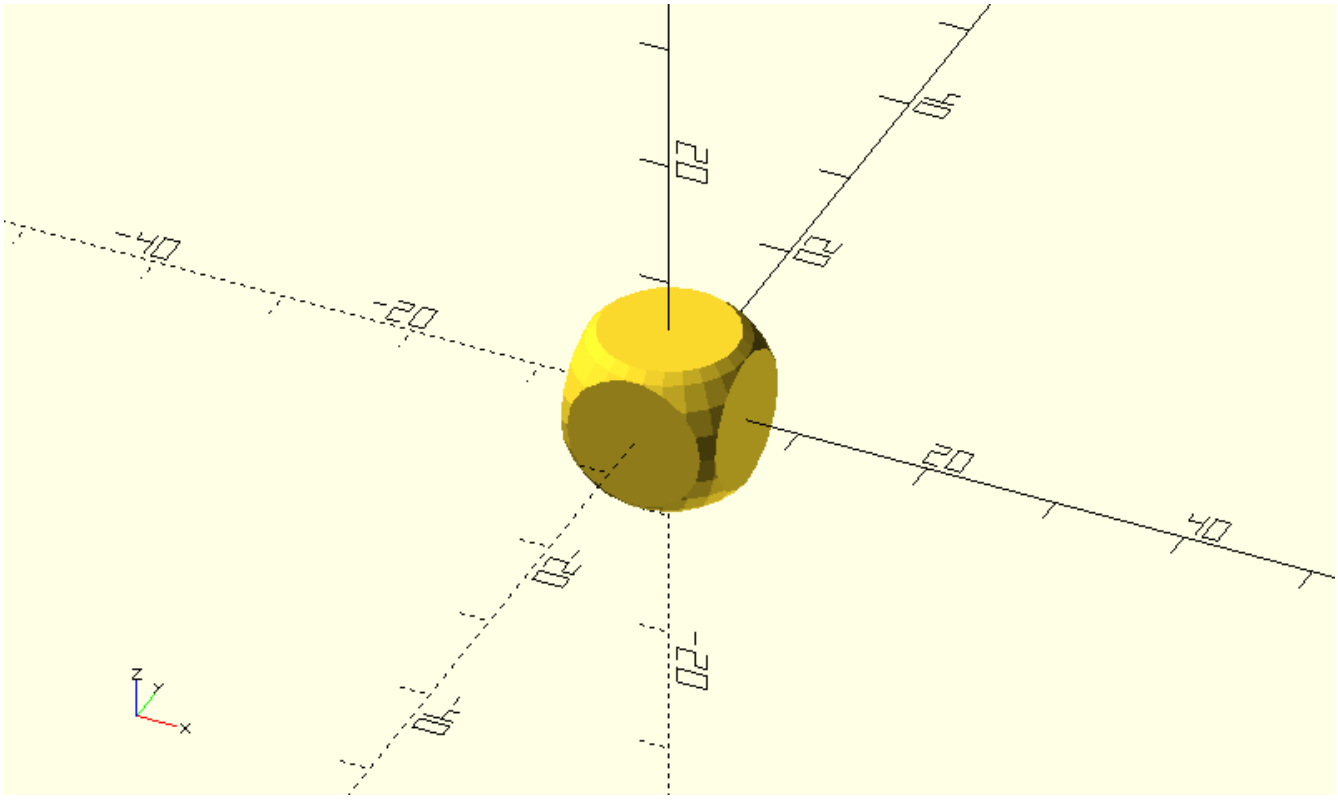
```
difference() {
    sphere(8);
    cube(12, center = true);
}
```



intersection

Esta operación realiza la intersección entre dos elementos. Es equivalente a realizar la operación booleana `and`.

```
intersection() {
    cube(12, center = true);
    sphere(8);
}
```



Módulos

Creación de módulos con `module`

OpenSCAD nos permite crear nuestros propios módulos, donde podemos agrupar diferentes elementos.

La sintaxis para definir un módulo es la siguiente:

```
module name ( parameters )
{
    actions
}
```

Ejemplo:

En este ejemplo hemos creado un módulo llamado `interseccion_cubo_esfera` donde hemos incluido el código que realiza la intersección entre un cubo y una esfera.

```
module interseccion_cubo_esfera() {
    intersection() {
        cube(12, center = true);
        sphere(8);
    }
}
```

Una vez que hemos creado el módulo `interseccion_cubo_esfera` podemos usarlo todas las veces que queramos, sólo tenemos que hacer lo siguiente:

```
interseccion_cubo_esfera();
```

Parámetros de entrada

Podemos definir parámetros de entrada en nuestros módulos para hacerlos mucho más versátiles. Por ejemplo, en el módulo anterior podríamos tener la **arista del cubo** y el **radio de la esfera** como **parámetros de entrada**, de modo que nos permita crear intersecciones de estas dos figuras de diferentes tamaños.

El módulo quedarías así con los parámetros de entrada:

```
module interseccion_cubo_esfera(arista, radio) {
    intersection() {
        cube(arista, center = true);
        sphere(radio);
    }
}
```

Y para utilizarlo sólo tendríamos que indicar los valores de entrada de la arista y del radio:

```
interseccion_cubo_esfera(12, 8);
```

Ejercicio

Crea un módulo que tenga parámetros de entrada para cada una de las siete piezas del Tetris [https://es.wikipedia.org/wiki/Tetris].

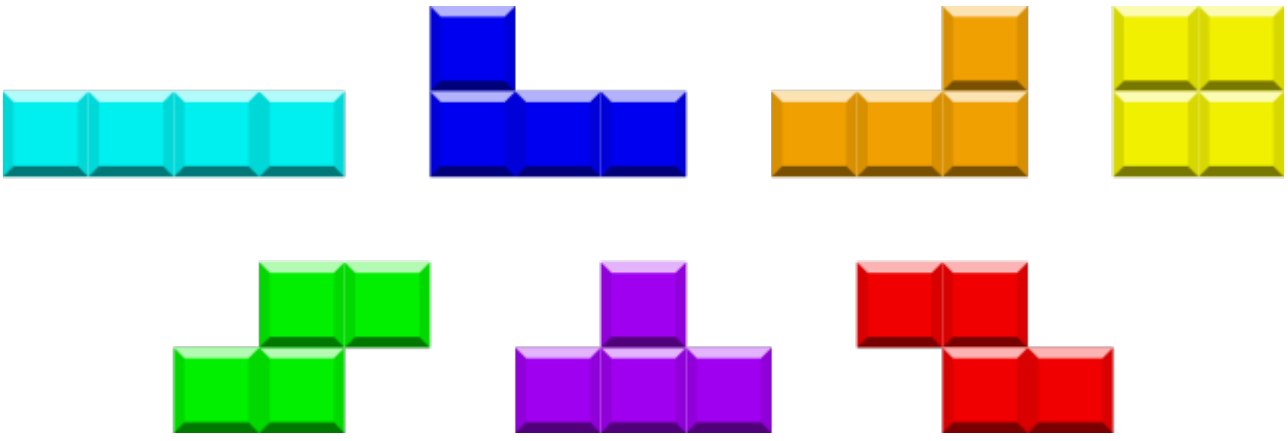


Imagen de Damian Yerrick.

Los módulos tienen que tener los siguientes nombres:

- pieza_I
- pieza_J
- pieza_L
- pieza_O
- pieza_S
- pieza_T
- pieza_Z

Créditos

Este material ha sido creado por José Juan Sánchez Hernández [http://josejuansanchez.org] para el grupo de trabajo **Aplicaciones al aula de la impresión 3D** del IES Celia Viñas (Almería).

Licencia

Referencias

- Página oficial de OpenSCAD [https://www.openscad.org].
- Curso de iniciación al diseño 3D con OpenSCAD (por ObiJuan) [http://diwo.bq.com/course/curso-de-iniciacion-al-diseno-3d-con-openscad-por-obijuan/].
- OpenSCAD CheatSheet [https://www.openscad.org/cheatsheet/index.html].