MateFun 0.16

Cambios y nuevas funcionalidades

Nuevas funciones básicas

Estas funciones son aquellas que con pasarle un parámetro, ya grafica una figura. Nuevas funciones:

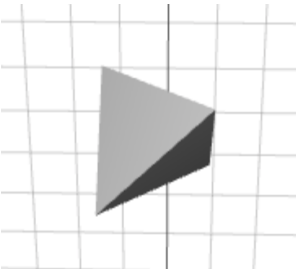
- **tetraedro**
- **prisma** → Antes era cubo, pero se renombró a prisma
- **octaedro**
- **dodecaedro**
- **icosaedro**

A continuación se muestra su descripción y ejemplos para saber cómo se usan.

tetraedro

```
tetraedro :: R -> Fig3D
```

Recibe un número real, que es el medida del radio y devuelve un tetraedro.



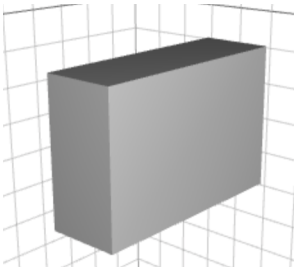
Ejemplos.

```
tetraedro(4.5)
tetraedro(3)
```

prisma

```
prisma :: (R X R X R) -> Fig3D
```

Antes era el cubo, pero se renombró a prisma. Recibe tres números reales, que son el ancho, alto y largo de la figura.



Nota: todos los trabajos que tuvieran la primitiva cubo ahora la tienen que cambiar por un prisma.

Ejemplos.

```
prisma(1, 3, 5)
prisma(2.3, 1, 8)
```

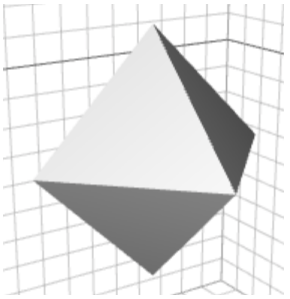
octaedro

```
octaedro :: R -> Fig3D
```

Recibe un número real, que es el medida del radio y devuelve un octaedro.

Ejemplos.

```
octaedro(4.5)
octaedro(3)
```



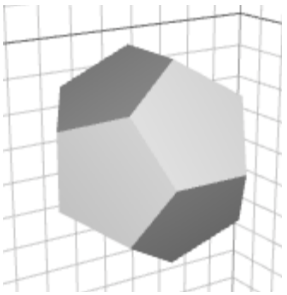
dodecaedro

```
dodecaedro :: R -> Fig3D
```

Recibe un número real, que es el medida del radio y devuelve un dodecaedro.

Ejemplos.

```
dodecaedro(4.5)
dodecaedro(3)
```



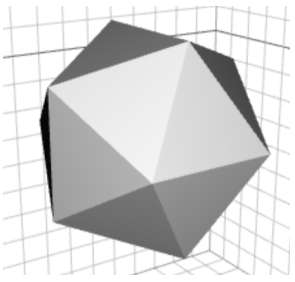
icosaedro

```
icosaedro :: R -> Fig3D
```

Recibe un número real, que es el medida del radio y devuelve un icosaedro.

Ejemplos.

```
icosaedro(4.5)
icosaedro(3)
```



Nuevas funciones para componer con otras funciones

Estas nuevas funciones precisan de una composición con otras primitivas para su correcto uso. Nuevas funciones:

- **transparencia3D**
- **juntarFigEn3D**

A continuación se muestra su descripción y ejemplos para saber cómo se usan.

transparencia3D

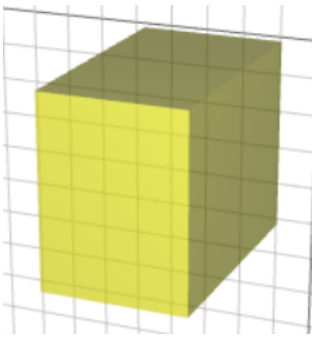
```
transparencia3D :: (Fig3D X R) -> Fig3D
```

Recibe una figura 3D y un número real con valor entre 0 y 1, donde 0 significa 0% de transparencia y 1 significa 100% de transparencia. Devuelve la figura 3D que recibió con su valor de transparencia ajustado.

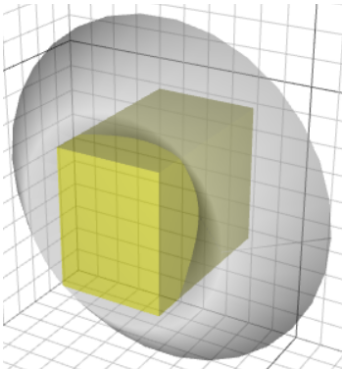
Nota: Si se aplica varias veces transparencia3D sobre una figura, la figura va a tomar el último valor de transparencia que fue aplicado. Por ejemplo, en este caso la esfera toma valor **0.5** (50%): `transparencia3D(transparencia3D(esfera(3), 0.25), 0.5)`

Ejemplos.

```
color3D(transparencia3D(prisma(3, 4, 5), 0.35), Amarillo)
```



```
juntar3D(color3D(transparencia3D(prisma(3, 4, 5), 0.35), Amarillo),
transparencia3D(anillo(2, 4, 3), 0.5))
```



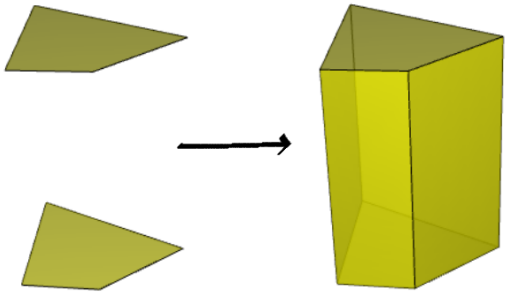
juntarFigEn3D

```
juntarFigEn3D :: (Fig X Fig X R) -> Fig3D
```

Dadas dos figuras 2D y una altura que las separa, construye una nueva figura 3D uniendo las figuras 2D mediante caras laterales. Las figuras 2D pasan a ser las bases de la nueva figura 3D, donde la primera figura es la base superior y la segunda figura es la base inferior.

Consideraciones: Para usar esta función, ambas figuras 2D tienen que ser círculos, rectángulos o polígonos. Si se usan polígonos, ambos tienen que tener la misma cantidad de vértices.

Visualmente, usando dos polígonos, esto hace la función juntarFigEn3D:



Nota: Debido a que se agrega esta primitiva, la función cilindro fue borrada, entonces todos los trabajos que tuvieran la primitiva cilindro ahora tienen que usar alguna de las siguientes opciones:

1. Usar la primitiva `juntarFigEn3D` en sustitución de la primitiva `cilindro`.
   Si antes se tenía
   ```
   cilindro(3, 5, 6)
   ```
   Ahora se usaría esto
   ```
   juntarFigEn3D(circ(3), circ(5), 6)
   ```

2. Definir las funciones `conoTrunc` y `cilindro` en un archivo, usando `juntarFigEn3D`:
   ```
   conoTrunc :: (R X R X R) -> Fig3D
   conoTrunc (r1, r2, h) = juntarFigEn3D(circ(r1), circ(r2), h)

   cilindro :: (R X R) -> Fig3D
   cilindro (radio, h) = juntarFigEn3D(circ(radio), circ(radio), h)
   ```

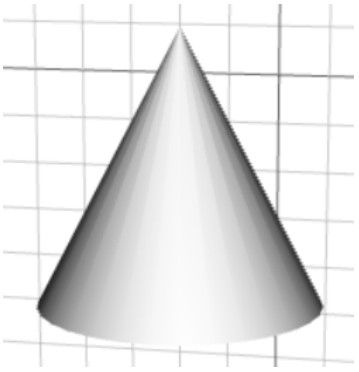
3. Solo definir la función `cilindro` en un archivo, usando `juntarFigEn3D`:
   ```
   cilindro :: (R X R X R) -> Fig3D
   cilindro (r1, r2, h) = juntarFigEn3D(circ(r1), circ(r2), h)
   ```

Para evitar confusiones, lo mejor puede que sea la **alternativa 2**, ya que realmente lo que dibujaba antes la función cilindro, era un cono truncado. Con esto, se diferencia lo que es un cono truncado de un cilindro.

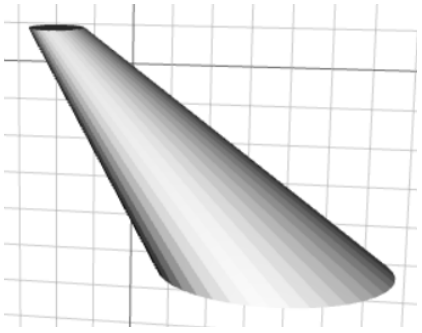
Ejemplos.

```
juntarFigEn3D(circ(1), circ(2), 4)
```

```
juntarFigEn3D(circ(0), circ(2), 4)
```

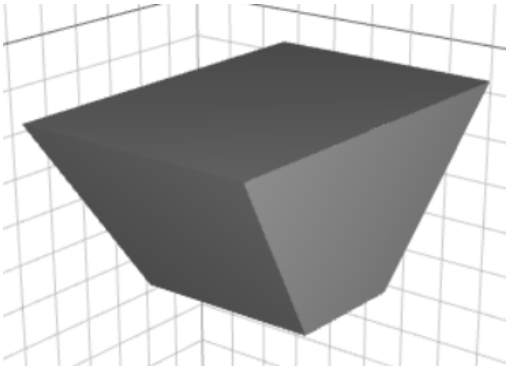


```
juntarFigEn3D(circ(0.5),
mover(circ(2), (4,-1)), 4)
```

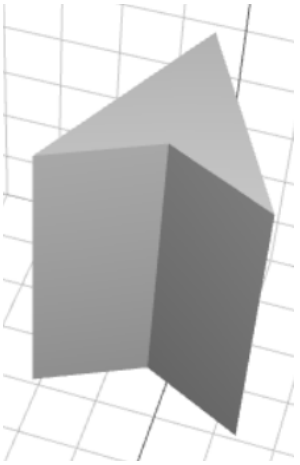


Cilindro oblicuo

```
juntarFigEn3D(rect(4,6), rect(3,2), 3)
```



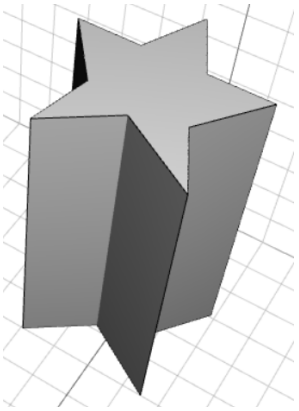
```
juntarFigEn3D(poli([(0,0), (1,2), (2,0), (1,0.5)]), poli([(0,0), (1,2), (2,0),
(1,0.5)]), 3)
```



Primero ejecutar en la línea de comandos:

```
estrella = poli([(0,2), (1,2), (1.5, 3), (1.5,3), (2,2), (3,2), (2.25, 1), (2.75,0), (1.5, 0.75), (0.5, 0), (0.75,1)])
```

Luego, ejecutar: `juntarFigEn3D(estrella, estrella, 5)`



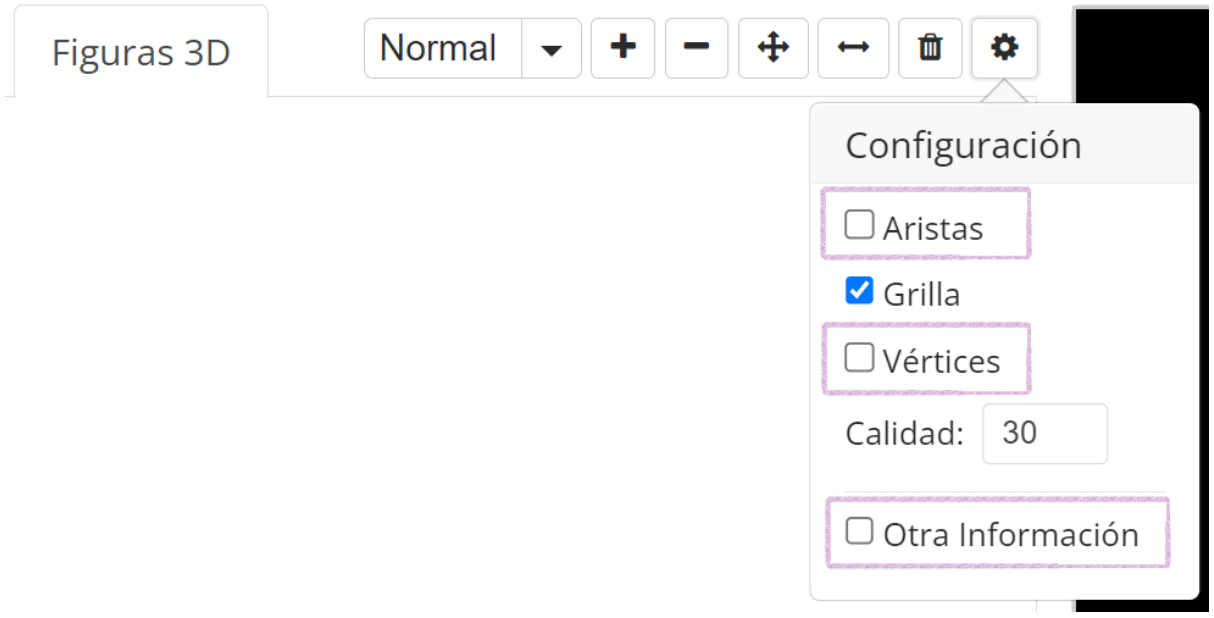
Nuevas opciones de configuración de las figuras 3D en el menú desplegable

En el menú desplegable encontramos tres nuevas opciones de configuración para las figuras 3D:

- **Aristas**: Al activarla, muestra las aristas de las figuras.
- **Vértices**: Al activarla, muestra los vértices de las figuras.
- **Otra Información**: Al activarla, muestra con líneas punteadas información complementaria de las figuras.

  Figuras en las cuales aplica esta opción:

  - **Esfera**: Muestra el radio de la esfera.
  - **Cilindro**: Muestra los radios de las bases y muestra la altura. Si las bases tienen el mismo radio, solo se muestra una línea punteada para la base inferior.
  - **Cilindro oblicuo**: Muestra los radios de las bases y muestra la altura. Si las bases tienen el mismo radio, solo se muestra una línea punteada para la base inferior.

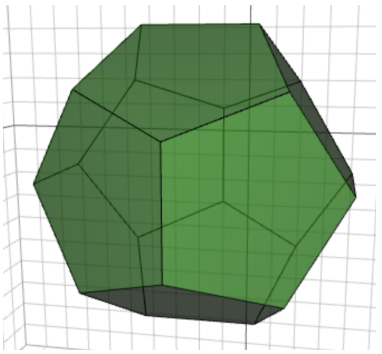


Algunos ejemplos de las nuevas opciones de configuración

Aristas

Marcar la opción **Aristas** y usar la siguiente función:

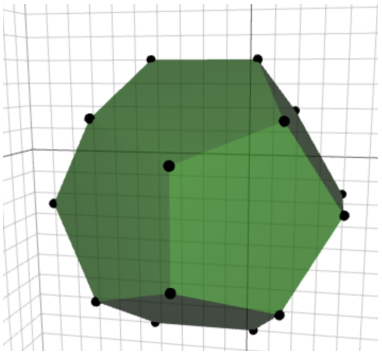
```
color3D(transparencia3D(dodecaedro(4), 0.25), Verde)
```



Vértices

Marcar la opción **Vértices** y usar la siguiente función:

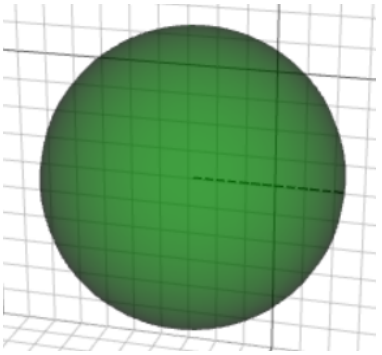
```
color3D(transparencia3D(dodecaedro(4), 0.25), Verde)
```



Otra información

Marcar la opción **Otra información** y usar la siguiente función:

```
transparencia3D(color3D(esfera(4), Verde), 0.25)
```



Ejemplo avanzado de un auto animado y poliedros regulares

Cargando el siguiente archivo se obtiene un ejemplo de un auto compuesto por la mayoría de nuevas figuras que se pueden hacer en MateFun. Además, también se obtiene un ejemplo para ver todos los poliedros regulares.

Una vez cargado el archivo, se puede ejecutar en la consola los siguientes comandos:

- `auto()`
- `poliedros()`
- `animarAuto()`
- `animarTodo()`

Archivo:

```
rueda :: () -> Fig3D
rueda () = anillo(1,4,1.2)

cilindro :: (R X R) -> Fig3D
cilindro (radio, altura) = juntarFigEn3D(circ(radio), circ(radio), altura)

ruedasTraseras :: () -> Fig3D
ruedasTraseras () = juntar3D(mover3D(rueda (), (0,-5,0)), mover3D(rueda (),
(5,-5,0)))

ruedasDelanteras:: () -> Fig3D
ruedasDelanteras () = juntar3D(mover3D(rotar3D(cilindro(2,1.8), (90,0,-90)),
(0,2,0)), mover3D(rotar3D(cilindro(2,1.8), (90,0,-90)), (5,2,0)))

ruedas :: () -> Fig3D
ruedas () = transparencia3D(color3D(juntar3D(ruedasTraseras(),
ruedasDelanteras()), Negro), 0.5)

carcasa :: () -> Fig3D
carcasa () = mover3D(rotar3D(escalar3D(juntarFigEn3D(poli([(0,0), (1,1), (3,1),
(4,0)]), poli([(0,0), (1,1), (3,1), (4,0)]), 2), 3), (0,90,0)), (2.5,-7,2))

luz :: () -> Fig3D
luz () = color3D(icosaedro(0.25), Amarillo)

luces :: () -> Fig3D
luces () = mover3D(juntar3D(luz(), mover3D(luz(), (5,0,0))), (0,4,3))

antena :: () -> Fig3D
antena () = mover3D(color3D(juntarFigEn3D(mover(circ(0.125), (-1,0)), circ(0.5),
1.5), Azul), (2.3,0,5.75))
```

```
aleron :: () -> Fig3D
aleron () = escalar3D(mover3D(color3D(juntar3D(prisma(0.75,0.75,7),
mover3D(rotar3D(tetraedro(0.65), (45,0,90)), (0,0,0.35))), Verde),
(2.5,-5,4.5)), (0.8))

matricula :: () -> Fig3D
matricula () = color3D(mover3D(octaedro(0.75), (2.5,5,2)), rgb(255, 80, 50))

retrovisores :: () -> Fig3D
retrovisores () = color3D(juntar3D(mover3D(dodecaedro(0.5), (5.75,1,3.5)),
mover3D(dodecaedro(0.5), (-0.75,1,3.5))), rgb(230, 50, 80))

auto :: () -> Fig3D
auto () = juntar3D(juntar3D(juntar3D(juntar3D(ruedas(), ruedas()),
juntar3D(carcasa(), luces())), juntar3D(antena(), aleron())),
juntar3D(matricula(), retrovisores()))

autoRotado :: () -> Fig3D
autoRotado () = rotar3D(juntar3D(juntar3D(juntar3D(juntar3D(ruedas(), ruedas()),
juntar3D(mover3D(carcasa(), (0,11.75,0.1)), luces())),
juntar3D(rotar3D(antena(), (90,0,90)), aleron())), juntar3D(matricula(),
retrovisores())), (0,0,-90))

animarAuto :: () -> Fig3D*
animarAuto () = [
  auto(),
  mover3D(auto(), (0,1,0)),
  mover3D(autoRotado(), (0,0,0))
]

poliedros :: () -> Fig3D
poliedros () =
rotar3D(transparencia3D(juntar3D(juntar3D(juntar3D(color3D(transparencia3D(mover
3D(tetraedro(3), (5,-5,-5)), 0.25), Rojo), mover3D(color3D(prisma(3, 5, 6),
Amarillo), (5,5,2))),juntar3D(mover3D(color3D(transparencia3D(octaedro(4),
0.25), Verde), (-5,-5,-2)), mover3D(color3D(dodecaedro(5), Azul), (-5,7,2)))),
transparencia3D(icosaedro(6), 0.25)), 0.3), (45,45,20))

animarTodo :: () -> Fig3D*
animarTodo () = [
  auto(),
  mover3D(auto(), (0,1,0)),
  mover3D(autoRotado(), (0,0,0)),
  poliedros()
]
```