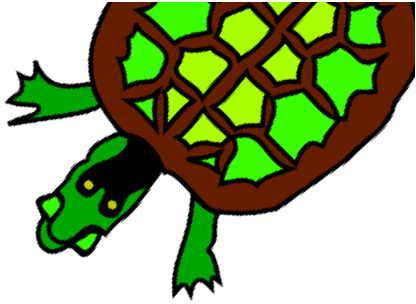
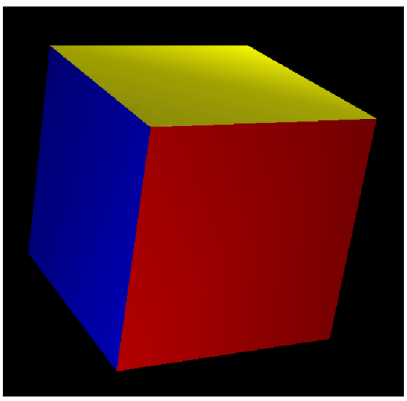


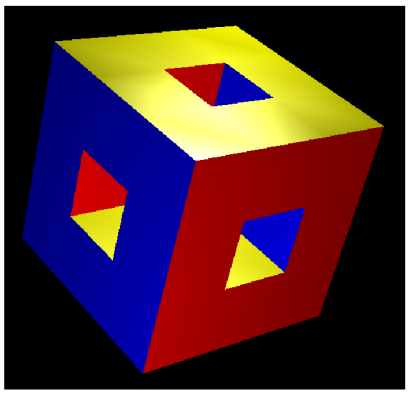
Apéndice D

La esponja de Menger

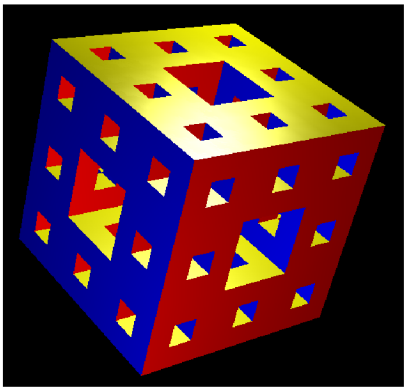
En este capítulo vamos a construir un sólido fractal llamado **la esponja de Menger**, cuyas primeras iteraciones son:



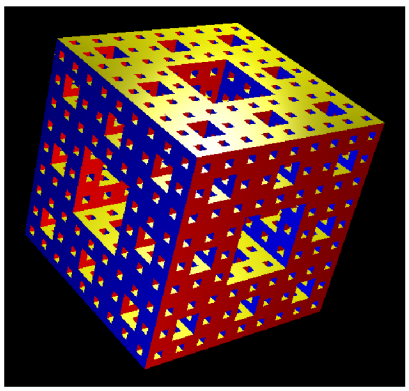
Estado inicial (paso 0)



Paso 1



Paso 2



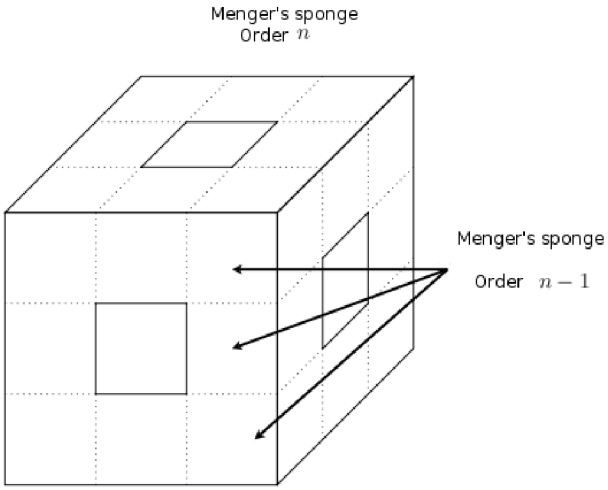
Paso 3

Este tema se divide en dos partes:

- Iniciciación, donde mostramos cómo crear el sólido usando la recursividad.
- Desarrollo y optimización, donde dibujaremos una esponja de Menger de orden 4.

D.1. Utilizando recursividad

Consideremos una esponja de Menger de orden n cuyo lado mide L.



El dibujo muestra que, realmente, esta *esponja* está formada por 20 esponjas de Menger de orden $n-1$, cada una de lado $\frac{L}{3}$. Hemos encontrado la estructura recursiva de la esponja.

El programa

```
para cubo :l
  si :contador=10000 [vistapoligono]
  # color de las caras laterales
  hazlocal "colores [amarillo magenta cyan azul]
# caras laterales
  repite 4
    [ poncolorlapiz ejecuta elemento contador :colores
      cuadrado :l giraderecha 90 avanza :l
      giraizquierda 90 balanceaderecha 90]
  poncolorlapiz rojo
  cabeceaabajo 90 cuadrado :l cabeceaarriba 90
  avanza :l cabeceaabajo 90
  poncolorlapiz verde
  cuadrado :l cabeceaarriba 90 retrocede :l
fin

para cuadrado :c
  haz "contador :contador+1
  empiezapoligono
    repite 4 [avanza :c giraderecha 90]
  finpoligono
fin
```

```
# Esponja de Menger
# p: nivel de recursividad
# l: arista del cubo final (grande)
para menger :l :p
  si :p=0 [cubo :l]
   [ hazlocal "p :p-1
     hazlocal "l :l/3
     repite 3 [ menger :l :p avanza :l ]
     retrocede 3*:l
     giraderecha 90 avanza :l giraizquierda 90
     menger :l :p avanza 2*:l menger :l :p retrocede 2*:l
     giraderecha 90 avanza :l giraizquierda 90
     repite 3 [menger :l :p avanza :l] retrocede 3*:l
  # lado derecho
    cabeceaabajo 90 avanza :l cabeceaarriba 90
     menger :l :p  avanza 2*:l menger :l :p retrocede 2*:l
     cabeceaabajo 90 avanza :l cabeceaarriba 90
     repite 3 [menger :l :p avanza :l]
     retrocede 3*:l giraizquierda 90 avanza :l giraderecha 90
     menger :l :p  avanza 2*:l menger :l :p retrocede 2*:l
     giraizquierda 90 avanza :l giraderecha 90
     repite 3 [menger :l :p avanza :l]
     retrocede 3*:l cabeceaabajo 90 retrocede :l cabeceaarriba 90
     menger :l :p  avanza 2*:l menger :l :p retrocede 2*:l
     cabeceaabajo 90 retrocede :l cabeceaarriba 90
   ]
fin

para esponja :p
  borrapantalla ocultatortuga
  haz "contador 0
  perspectiva poncolorpapel 0
  menger 800 :p
  mecanografia [Numero de poligonos: ] escribe :contador
  vistapoligono
fin
```

Este programa consta de cuatro procedimientos:

- `cuadrado :c`

  Este procedimiento dibuja un cuadrado de lado `:c`, almacenándolo para dibujarlo en 3-D. La variable `contador` guardará el número de polígonos dibujados.

- `cubo :l`

  Este procedimiento dibuja un cubo de arista `:l`, llamando al procedimiento `cuadrado`.

- `menger :l :p`

  Llegamos al más importante del programa; dibuja el *motivo* de Menger de orden p y lado l. Este *motivo* se crea de modo recursivo, como acabamos de explicar.

- `esponja :p`

  Crea una esponja de Menger de orden p y lado 800 y la dibuja en el visor 3D.

La mayor dificultad de este programa proviene de su consumo de memoria. La cantidad de memoria asignada por defecto para xLogo (ver pág. 17) no permite dibujar una esponja de orden 3 (`esponja 3`), ya que requiere dibujar **48 000** polígonos. Debemos aumentar la memoria asignada a xLogo hasta un mínimo de 256 Mb ... u optimizar el código.

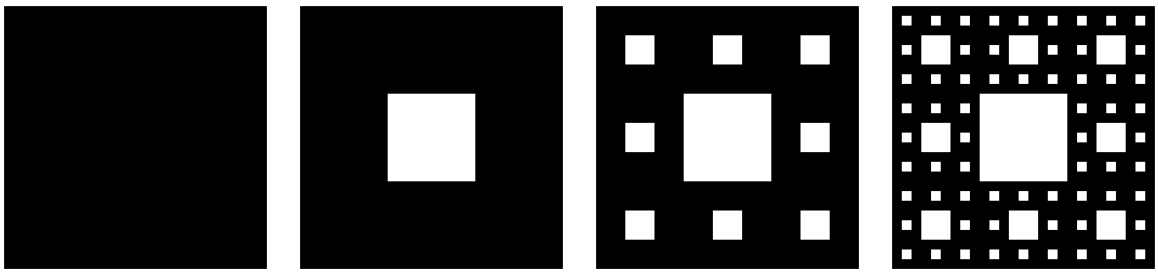
D.2. Segunda aproximación. Sólido de orden 4

La principal característica del programa anterior es su utilización de la estructura recursiva del sólido fractal. podemos observar que es similar al método utilizado para dibujar el copo de Koch en la página 99. La principal ventaja de utilizar recursividad es que el código del program es el más pequeño posible. A cambio, la principal desventaja es el número de polígonos que se deben crear que, como dijimos, alcanza los 48.000. Esto hace inviable plantearse una esponja de orden superior, pues los requisitos de memoria crecerán muy rápido.

Si queremos dibujar una esponja de Menger de orden 4, debemos replanearnos el programa y olvidar la recursividad.

D.2.1. La alfombra de Sierpinski

La esponja de Menger es la generalización en 3 dimensiones de una figura plana llamada **la alfombra de Sierpinski**, cuyas primeras iteraciones son:



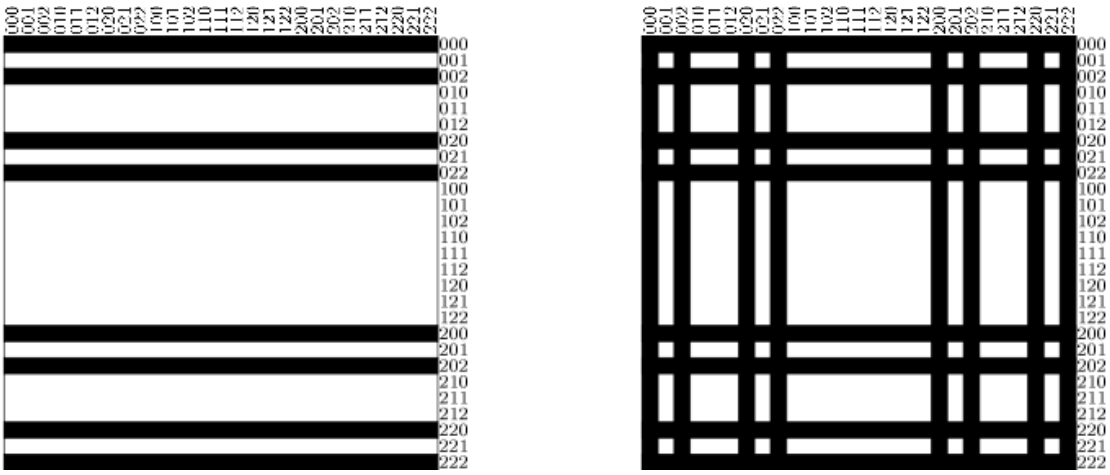
Paso 0 **Paso 1** **Paso 2** **Paso 3**

Observamos que cada cara de la esponja de Menger es una alfombra de Sierpinski.

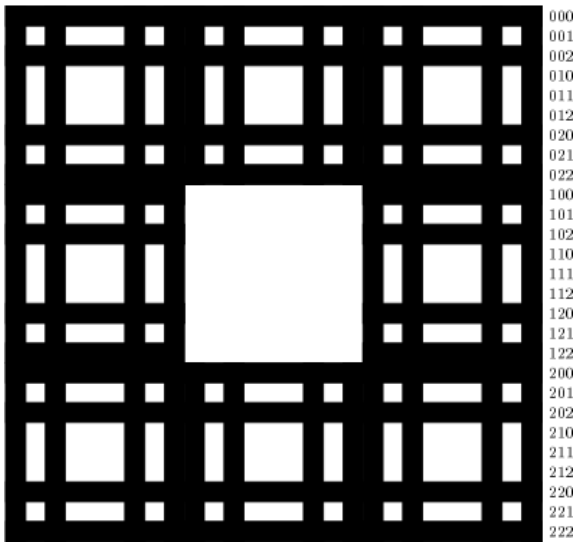
D.2.2. Dibujando una alfombra de Sierpinski de orden p

El objetivo es reducir al mímino el número de polígonos necesarios para dibujar una alfombra de Sierpinski. El ejemplo explica cómo dibujar una alfombra de Sierpinski de orden 3. Aquí, el cuadrado inicial consta de $3^3 = 27$ filas y 27 columnas. Escribimos en base 3 el número de cada fila y cada columna.:

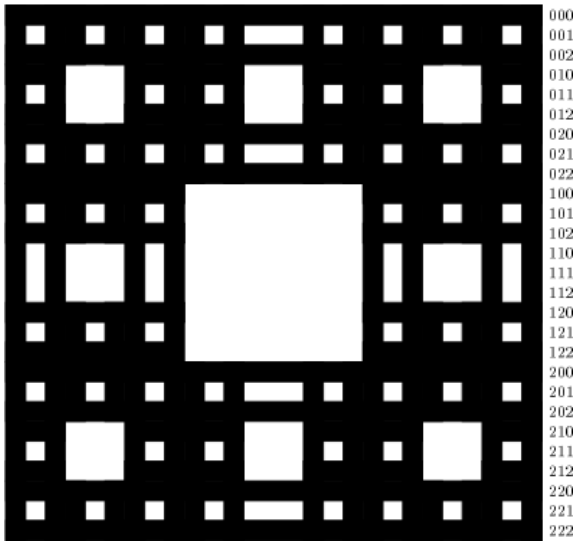
Primera etapa: Para cada fila cuyo número no contiene ningún 1, trazamos una línea de 27 unidades. Por simetría, realizamos la misma operación con las columnas.



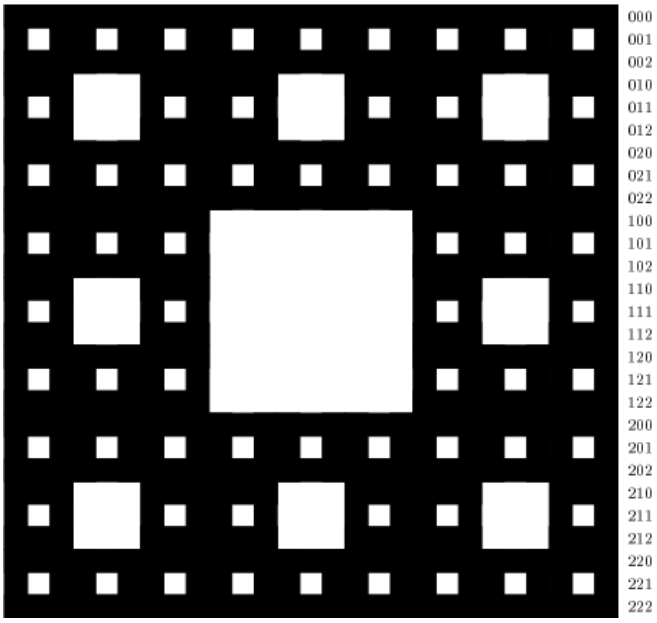
Segunda etapa: Ahora buscamos las filas que sólo tienen un 1 en la primera posición. Dibujaremos alternativamente rectángulos de longitud 9 unidades. Como antes, la simetría nos obliga a repetir esta operación con las columnas.



Tercera etapa: Buscamos las filas que sólo tienen un 1 en la segunda posición. Dibujaremos los rectángulos de acuerdo a la serie [3 3 6 3 6 3 3]. (dibujamos 3, 3 en blanco, dibujamos 6, etc). Repetimos con las columnas.



Última etapa: Miramos ahora las líneas que contienen un doble 1 en las dos primeras posiciones. La serie de rectángulos se dibuja de acuerdo al esquema [3 3 3 9 3 3 3], y repetimos en las columnas.



La construcción de la alfombra de Sierpinski de orden 3 está completa. Para dibujarlo, sólo necesitamos $16 + 16 + 32 + 16 = 80$ polígonos.

D.2.3. Otros esquemas posibles usando columnas

Para recapitular la construcción anterior, estos son los diferentes tipos de esquemas por columnas, de acuerdo al número de líneas (* representa 0 ó 2):

Número de línea	Esquema a aplicar
***	27
1**	9 9 9
1	3 3 6 3 6 3 3
11*	3 3 3 9 3 3 3

Del mismo modo, para construir una alfombra de orden 4, necesitamos un cuadrado de $3^4 = 81$ unidades. Los números de línea y de columna tendrán 4 cifras al descomponerlos a base 3. Para cada tipo de número de línea, el esquema a aplicar será (* representa 0 ó 2):

Número de línea	Esquema a aplicar
****	81
1***	27 27 27
*1**	9 9 18 9 18 9 9
**1*	3 3 6 3 6 3 6 3 6 3 6 3 6 3 6 3 6 3 3
11	3 3 3 9 3 3 6 3 3 9 3 3 6 3 3 9 3 3 3
1*1*	3 3 6 3 6 3 3 27 3 3 6 3 6 3 3
11**	9 9 9 27 9 9 9
111*	3 3 3 9 3 3 3 27 3 3 3 9 3 3 3

De aquí se deduce que hacen falta 496 polígonos para dibujar una alfombra de Sierpinski de orden 4.

Por si alguien tiene la duda, este es el esquema para el caso $n = 2$:

Número de línea	Esquema a aplicar
**	9
1*	3 3 3

D.2.4. El programa

```
# Dibuja una alfombra de Sierpinski de orden :p y arista :lado
para alfombra :lado :p
haz "unidad :lado/(potencia 3 :p)
  si :p=0 [ rec :lado :lado alto]
  si :p=1 [repite 4 [rec :lado :unidad avanza :lado giraderecha 90 ] alto]
  repitepara (lista "x 1 potencia 3 :p)
    [ hazlocal "cantorx cantor :x :p []
# no dibujan los elementos que tengan un 1 en ultima posicion
      si  no (1=ultimo :cantorx)
       [ hazlocal "nom evalua menosultimo :cantorx "
         dibujacolumna :x devuelvepropiedad "map :nom ]
    ]
fin
```

```
# Devuelve la descomposicion en base 3 del numero x
# p indice de profundidad 3^p
# :lista lista vacia inicialmente

para cantor :x :p :lista
  si :p=0 [devuelve :lista ]
  hazlocal "a potencia 3 :p-1
  si :x<= :a
    [ devuelve cantor  :x :p-1  frase :lista 0]
    [ si :x<=2*:a
        [devuelve cantor  :x-:a :p-1  frase :lista 1]
      devuelve cantor :x-2*:a :p-1 frase :lista 0]
fin

# trazado de la columna numero x de acuerdo al esquema de
#    construccion definido en la lista
para dibujacolumna :x :lista
  subelapiz
    giraderecha 90 avanza (:x-1)*:unidad giraizquierda 90
  bajalapiz
  des :lista
  subelapiz giraizquierda 90 avanza (:x-1)*:unidad
  giraderecha 90 avanza :x*:unidad giraderecha 90 bajalapiz des :lista
  subelapiz
    giraizquierda 90 retrocede :x*:unidad
  bajalapiz
fin

# trazado de un rectangulo con las medidas dadas
# el pologono se guarda para verlo en 3d
para rec :lo :la
  haz "contador :contador+1
  empiezapoligono
    repite 2
     [avanza :lo giraderecha 90 avanza :la giraderecha 90]
  finpoligono
fin

# Inicializa las diferentes columnas posibles para la alfombra de orden 1 a 4
para initmap
  ponpropiedad "map 111 [3 3 3 9 3 3 3 27 3 3 3 9 3 3 3]
  ponpropiedad "map 110 [9 9 9 27 9 9 9]
  ponpropiedad "map 101 [3 3 6 3 6 3 3 27 3 3 6 3 6 3 3]
```

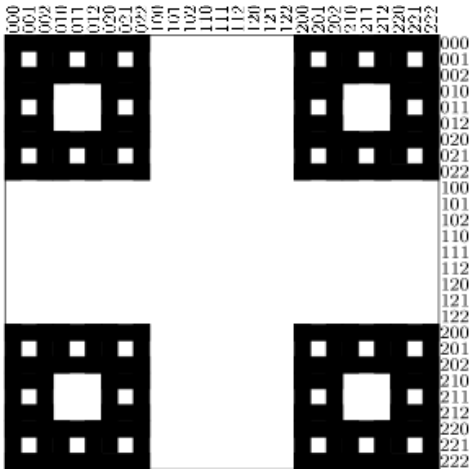
```
  ponpropiedad "map 011 [3 3 3 9 3 3 6 3 3 9 3 3 6 3 3 9 3 3 3]
  ponpropiedad "map 000 [81]
  ponpropiedad "map 100 [27 27 27]
  ponpropiedad "map 010 [9 9 18 9 18 9 9]
  ponpropiedad "map 001 [3 3 6 3 6 3 6 3 6 3 6 3 6 3 6 3 6 3 3]
  ponpropiedad "map 01 [3 3 6 3 6 3 3]
  ponpropiedad "map 00 [27]
  ponpropiedad "map 10 [9 9 9]
  ponpropiedad "map 11 [3 3 3 9 3 3 3]
  ponpropiedad "map 1 [3 3 3]
  ponpropiedad "map 0 [9]
fin

# si la decomposicion es [1 0 1] --> devuelve 101
para evalua :lista :motivo
  si vacio? :lista [devuelve :motivo]
  [ hazlocal "motivo palabra :motivo primero :lista
    devuelve evalua menosprimero :lista :motivo ]
fin

# trazado de los grupos de rectangulos de cada columna alternativamente
para des :lista
  hazlocal "suma 0
  repitepara (lista "i 1 cuenta :lista)
    [ hazlocal "elemento elemento :i :lista
      hazlocal "suma :elemento+:suma
      si par? :i
        [ subelapiz avanza :elemento*:unidad bajalapiz ]
        [ rec :elemento*:unidad :unidad
          avanza :elemento*:unidad]
    ]
  subelapiz retrocede :suma * :unidad bajalapiz
fin

# prueba si un numero es par
para par? :i
  devuelve 0 = resto :i 2
fin

para sierpinski :p
  borrapantalla perspectiva ocultatortuga initmap
  haz "contador 0
  alfombra 810 :p
```

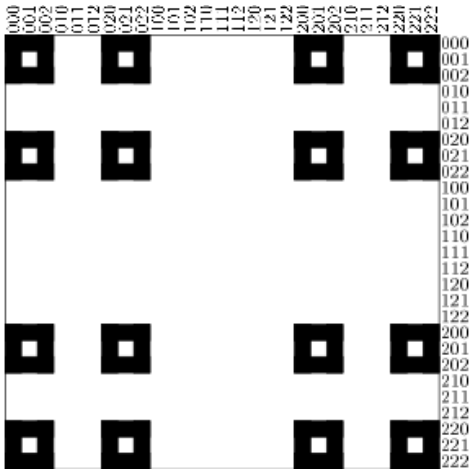
```
  mecanografia "Numero\ de\ poligonos:\  escribe :contador
  vistapoligono
fin
```

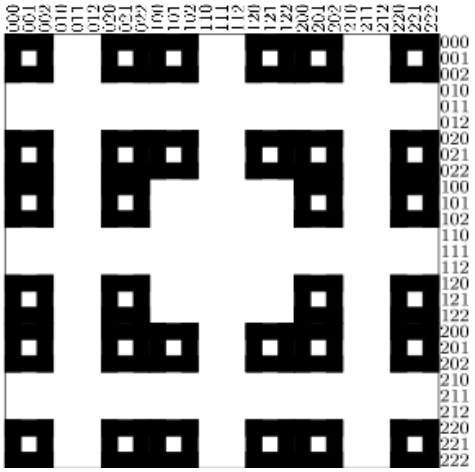
`sierpinski 3` dibuja una alfombra de Sierpinski de orden 3 y lado 810. Primera etapa conseguida, podemos volver a la esponja de Menger.

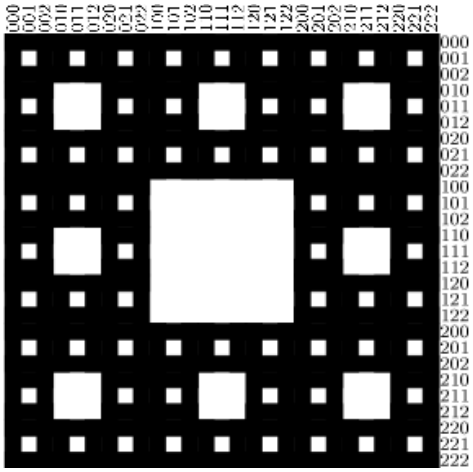
D.2.5. La esponja de Menger de orden 4

La esponja de Menger possee múltiples propiedades de simetría. Para generar la esponja, vamos a trazar las diferentes secciones a lo largo del plano (xOy) y las repetiremos sobre los planos (yOz) y (xOz). Para explicar qué pasa, echemos un vistazo a la esponja de orden 3. Cuando cortamos la esponja con un plano vertical, podemos obtener cuatro motivos distintos:









Para trazar una esponja de orden 3, vamos a recorrer los números de 1 a 27, es decir, de 001 a 222 en base 3. Para cada número, aplicaremos la sección adecuada que nos generará la figura en las 3 direcciones: (Ox), (Oy) y (Oz).

El código

Este programa dibujará los sólidos de Menger de orden 0, 1, 2, 3 y 4. El número de procedimientos es importante, y precisa de algunas explicaciones.

```
# Comando de Inicio: esponja 3
#
# Dibuja una alfombra de Sierpinski de orden :p y arista :lado
para alfombra :lado :p
  haz "unidad :lado/(potencia 3 :p)
  si :p = 0 [ rec :lado :lado alto ]
  si :p = 1 [ repite 4 [rec :lado :unidad avanza :lado giraderecha 90 ] alto]
  repitepara (lista "x 1 potencia 3 :p)
    [ hazlocal "cantorx cantor :x :p []
# no traza elementos que tienen un 1 en la ultima posicion
      si  no (1 = ultimo :cantorx)
        [ hazlocal "nom evalue menosultimo :cantorx "
          dibujacolumna :x devuelvepropiedad "map :nom ]
    ]
fin

# devuelve la decomposicion en base 3 del numero x
# p indica de profundidad 3^p
# :lista lista vacia al principio

para cantor :x :p :lista
  si :p = 0 [devuelve :lista]
  hazlocal "a potencia 3 :p-1
  si :x<= :a
    [ devuelve cantor  :x :p-1  frase :lista 0 ]
    [ si :x<=2*:a [devuelve cantor  :x-:a :p-1  frase :lista 1]
      devuelve cantor :x-2*:a :p-1 frase :lista 2]
fin

# trazado de la columna numero x respecto al esquema de construccion
#    definido en la lista
para dibujacolumna :x :lista
  subelapiz
    giraderecha 90 avanza (:x-1)*:unidad giraizquierda 90
```

```
  bajalapiz
  des :lista
  subelapiz
    giraizquierda 90 avanza (:x-1)*:unidad giraderecha 90
    avanza :x*:unidad giraderecha 90
  bajalapiz
  des :lista
  subelapiz giraizquierda 90 retrocede :x*:unidad bajalapiz
fin

# dibuja un rectangulo con las  dimensiones dadas
# el poligono se guarda para el visor 3D
para rec :lo :la
  haz "contador :contador+1
  empiezapoligono
    repite 2 [avanza :lo giraderecha 90 avanza :la giraderecha 90]
  finpoligono
fin

# Inicializa las diferentes columnas posibles para las alfombras de orden 1 a 4
para initmap
  ponpropiedad "map 111 [3 3 3 9 3 3 3 27 3 3 3 9 3 3 3]
  ponpropiedad "map 110 [9 9 9 27 9 9 9]
  ponpropiedad "map 101 [3 3 6 3 6 3 3 27 3 3 6 3 6 3 3]
  ponpropiedad "map 011 [3 3 3 9 3 3 6 3 3 9 3 3 6 3 3 9 3 3 3]
  ponpropiedad "map 000 [81]
  ponpropiedad "map 100 [27 27 27]
  ponpropiedad "map 010 [9 9 18 9 18 9 9]
  ponpropiedad "map 001 [3 3 6 3 6 3 6 3 6 3 6 3 6 3 6 3 6 3 3]
  ponpropiedad "map 01 [3 3 6 3 6 3 3]
  ponpropiedad "map 00 [27]
  ponpropiedad "map 10 [9 9 9]
  ponpropiedad "map 11 [3 3 3 9 3 3 3]
  ponpropiedad "map 1 [3 3 3]
  ponpropiedad "map 0 [9]
fin

# si la decomposicion es [1 0 1] --> devuelve 101
# si la decomposicion es [1 0 2] --> devuelve 100
# los elementos de la lista son concatenados en una palabra.
#  los 2 son reemplazados por dos ceros
para evalue :lista :mot
  si vacio? :lista
```

```
    [ devuelve :mot ]
    [ hazlocal "primera primero :lista
      si :primera=2 [hazlocal "primera 0 ]
      hazlocal "mot palabra :mot :primera
      devuelve evalue menosprimero :lista :mot
    ]
fin

# dibujamos los grupos de rectangulos de cada columna, alternativamente
para des :lista
  hazlocal "suma 0
 repitepara (lista "i 1 cuenta :lista)
   [ hazlocal "elemento elemento :i :lista
     hazlocal "suma :elemento + :suma
     si par? :i
      [ subelapiz avanza :elemento*:unidad bajalapiz ]
      [ rec :elemento*:unidad :unidad avanza :elemento*:unidad]
   ]
   subelapiz retrocede :suma * :unidad bajalapiz
fin

# prueba si un numero es par
para par? :i
  devuelve 0 = resto :i 2
fin

para sierpinski :p
  borrapantalla perspectiva ocultatortuga
  initmap
  haz "contador 0
  alfombra 810 :p
  mecanografia "numero\ de\ poligonos:\  escribe :contador
  vistapoligono
fin

# suprime el ultimo 1 de la lista :lista
para borraultimouno :lista
  repitepara (lista "i cuenta :lista 1 cambiasigno 1)
    [ hazlocal "elemento elemento :i :lista
      si :elemento = 1
        [ hazlocal "lista reemplaza :lista :i 0 alto ]
        [ si :elemento = 2 [alto]]
    ]
```

```
  devuelve :lista
fin

# Esponja de Menger de arista dada y de profundidad :p

para menger :lado :p
  haz "unidade :lado/(potencia 3 :p)
  repitepara (lista "z 1 potencia 3 :p)
    [ hazlocal "cantorz cantor :z :p []
      hazlocal "last ultimo :cantorz
      hazlocal "cantorz menosultimo :cantorz
      si :last = 0
        [ hazlocal "orden evalue borraultimouno :cantorz "]
        [ hazlocal "orden evalue :cantorz "]
      hazlocal "orden palabra "corte :orden
      draw3alfombra :lado :orden :z
      subelapiz cabeceaarriba 90 avanza :unidad cabeceaabajo 90 bajalapiz
    ]
    draw3alfombra :lado :orden (potencia 3 :p)+1
fin

# trazado de las alfombras de sierpinski de orden :p
# a lo largo de cada eje (ox), (oy) y (oz)
# a la altitud :z
para draw3alfombra :lado :orden :z
  subelapiz centro
  cabeceaarriba 90 avanza (:z-1)*:unidade cabeceaabajo 90 bajalapiz
  poncolorlapiz azul ejecuta :orden :lado
  subelapiz centro
  balanceaizquierda 90 avanza (:z-1)*:unidade cabeceaabajo 90  bajalapiz
  poncolorlapiz amarillo ejecuta :orden :lado
  subelapiz centro
  cabeceaarriba 90 avanza :lado giraderecha 90
  avanza (:z-1)*:unidade cabeceaabajo 90 bajalapiz
  poncolorlapiz magenta ejecuta :orden :lado
fin


# procedimiento principal
# dibuja una esponja de Menger d profundidad p
para esponja :p
  borrapantalla perspectiva ocultatortuga
  hazlocal "tiempo tiempo
```

```
  initmap
  haz "contador 0
  si :p=0 [cubo 405] [menger 405 :p]
# muestra el tiempo y establece el numero de poligonos necesarios
#    para la construccion
  mecanografia "Numero\ de\ poligonos:\  escribe :contador
  mecanografia "Tiempo\ transcurrido:\  escribe tiempo -:tiempo
  vistapoligono
fin

# seccion para Menger de orden 2
para corte1 :lado
  repite 4
    [alfombra :lado/3 1 subelapiz avanza :lado giraderecha 90 bajalapiz]
fin

para corte0 :lado
  alfombra :lado 2
fin

# seccion para Menger de orden 3
para corte10 :lado
  repite 4
    [ alfombra :lado/3 2 subelapiz avanza :lado giraderecha 90 bajalapiz]
fin

para corte01 :lado
  repite 4
    [ repite 2 [corte1 :lado/3 subelapiz avanza :lado/3 bajalapiz]
      avanza :lado/3 giraderecha 90 ]
fin

para corte11 :lado
  repite 4
    [ corte1 :lado/3 subelapiz avanza :lado giraderecha 90 bajalapiz]
fin

para corte00 :lado
  alfombra :lado 3
fin

# seccion para Menger de orden 4
para corte000 :lado
```

```
  alfombra :lado 4
fin

para corte100 :lado
  repite 4
    [ alfombra :lado/3 3 subelapiz avanza :lado giraderecha 90 bajalapiz]
fin

para corte010 :lado
  repite 4
   [ repite 2 [corte10 :lado/3 subelapiz avanza :lado/3 bajalapiz]
     avanza :lado/3 giraderecha 90]
fin

para corte001 :lado
  repite 4
    [ repite 2 [corte01 :lado/3 subelapiz avanza :lado/3 bajalapiz]
      avanza :lado/3 giraderecha 90]
fin

para corte110 :lado
  repite 4
    [ corte10 :lado/3 subelapiz avanza :lado bajalapiz giraderecha 90 ]
fin

para corte111 :lado
  repite 4
    [ corte11 :lado/3 subelapiz avanza :lado giraderecha 90 bajalapiz]
fin

para corte101 :lado
  repite 4
    [ corte01 :lado/3 subelapiz avanza :lado giraderecha 90 bajalapiz]
fin

para corte011 :lado
  repite 4
    [ repite 2 [corte11 :lado/3 subelapiz avanza :lado/3 bajalapiz]
      avanza :lado/3 giraderecha 90]
fin

para corte :lado
  alfombra :lado 1
```

```
fin

para cubo :lado
  repite 2
    [ poncolorlapiz azul rec :lado :lado
      subelapiz avanza :lado cabeceaabajo 90 bajalapiz
      poncolorlapiz amarillo rec :lado :lado
      subelapiz avanza :lado cabeceaabajo 90  bajalapiz ]
  poncolorlapiz magenta
  subelapiz
     balanceaizquierda 90 giraizquierda 90 avanza :lado giraderecha 90
  bajalapiz rec :lado :lado
  subelapiz
     giraderecha 90 avanza :lado giraizquierda 90 balanceaderecha 90
     giraderecha 90 avanza :lado giraizquierda 90 balanceaderecha 90
  bajalapiz rec :lado  :lado
  balanceaizquierda 90 giraizquierda 90 avanza :lado giraderecha 90
fin

para cubos
  borrapantalla perspectiva ocultatortuga
  hazlocal "temps tiempo
  initmap
  haz "contador 0
  repite 4
    [ si contador=1 [cubo 405]
                    [menger 405 contador-1]
     subelapiz avanza 1000 giraderecha 90 bajalapiz ]
# muestra el tiempo y el numero de poligonos necesarios en la construcion
  mecanografia "Numero\ de\ poligonos:\  escribe :contador
  mecanografia "Tiempo\ empleado:\  escribe tiempo -:temps
  vistapoligono
fin
```

Con todo lo anterior, y tras ajustar la memoria disponible para xLogo a 640 Mb, si tecleamos

```
esponja 4:
```