Para realizar esta obra, me he basado en el concepto de una teselación. Una teselación del plano es un patrón que cubre completamente el plano por figuras, sin dejar huecos entre estas ni superponerse. Estas figuras se obtienen aplicando movimientos rígidos a una figura inicial, que suele ser un polígono regular.

En este caso, he elegido un cuadrado para teselar el espacio. Para hacer una construcción iterativa del teselado, hemos de tener cuidado del orden que elegimos para colocar las teselas. Más formalmente, si entendemos una teselación por un recubrimiento por teselas (en este caso los mencionados cuadrados) y llamamos A_n al conjunto formado por el n-ésimo cuadrado que colocamos en el plano, debemos procurar encontrar un orden tal que $\bigcup_{n\in\aleph} A_n = \Re^2$, en donde identificamos, como es natural, el plano que vamos a recubrir como el espacio euclídeo bidimensional.

El orden que he elegido es el de una espiral, que creo que es lo que más natural puede llegar a resultar. Para visualizar que este orden de iteración consigue teselar el plano, lo más sencillo es identificar cada cuadrado con su centro. De esta forma, una demostración de que este orden tesela el plano se reduciría a demostrar que existe una biyección entre el conjunto de los números naturales y el conjunto de pares de números enteros.

Sin embargo, convencerse de que este orden tesela el plano es mucho más sencillo si simplemente se ve como itera la construcción de GeoGebra, por lo que nos podemos creer que esto es cierto y seguir con los detalles de cómo se construye esta figura.

Para entender cómo se construye la figura, primero tenemos que entender cómo se construye cada cuadrado. Esto es bastante sencillo. Definimos el giro $G(\theta, x, y) = (Gx(\theta, x, y), Gy(\theta, x, y))$, con $Gx(\theta, x, y) = cos\theta\, x + sen\theta\, y$, $Gy(\theta, x, y) = -sen\theta\, x + cos\theta\, y$. Como geogebra nos permite trabajar con funciones reales de variable vectorial (campos escalares), pero no con funciones que lleguen a $\Re^2$ (campos vectoriales), utilizaremos Gx y Gy para trabajar en geogebra, mientras que G es el giro que tenemos en mente y el que usaré para explicar las ideas.

Definimos 4 puntos, A,B,C y D, como vértices del cuadrado. Para conseguir que el cuadrado crezca como lo hace en la construcción, tomamos estos puntos para que tengan la forma:

$$A = G(\pi\cdot t\text{^}(1/3), t, -t) \quad B = G(\pi\cdot t\text{^}(1/3), -t, -t)$$
$$C = G(\pi\cdot t\text{^}(1/3), -t, t) \quad D = G(\pi\cdot t\text{^}(1/3), t, t)$$

Con t un número real que crece desde 0 hasta 1 (es un deslizador).

Nos fijamos que con estas elecciones vamos a crear un cuadrado de lado 1, al que también se le conoce como bola unidad cerrada cuando se trabaja con la norma del máximo. Viendo el cuadrado como dicha bola, se puede probar que los puntos se quedan siempre dentro del cuadrado (cosa que no pasa si no ponemos la raíz cúbica en el ángulo).

Activando el rastro de los segmentos AB, BC, CD y DA, conseguimos colorear el cuadrado por dentro. Para los colores dinámicos, utilizamos los colores HSV con matiz t, saturación 1 y valor 1. De esta forma, conseguimos el efecto de un solo cuadrado. Para conseguir el

resto, tan solo tenemos que aplicar una transición al que acabamos de construir y dejar que sea el rastro de los segmentos el que pinte el nuevo cuadrado. No obstante, en la construcción a veces el giro se invierte. Para conseguir esto, basta multiplicar el ángulo por -1 cuando queramos que esto ocurra.

Ahora que sabemos cómo construir un cuadrado, vamos a explicar como moverlo para crear el efecto de la teselación.

Primero, con el deslizador t que va de 0 a 1 vamos a detectar cuando se termina de construir un cuadrado. De esta forma, sabremos cuando podemos moverlo.

Parece lógico pensar que si t es igual a 1, entonces hemos terminado y podemos pasar a pintar el siguiente cuadrado. Sin embargo, nos encontramos con un problema común en informática, los errores de redondeo. Como 0.999 es muy parecido a 1, cuando un deslizador incrementa no notamos diferencia alguna en construcciones continuas, pero a la hora de comprobar si el valor del deslizador es igual a 1, esta comparación devuelve un valor de falso. En geogebra, estos errores se suelen acumular bastante en los deslizadores que imitan un número real.

Una solución típica en informática es permitir un pequeño margen de error (Comúnmente conocido como un epsilon). Sin embargo, en geogebra el error es bastante difícil de controlar de esta forma, especialmente cuando los deslizadores tienen velocidades elevadas. Debemos buscar una alternativa.

Otra cosa que podemos detectar es cuando el deslizador cambia a un menor que el anterior. Cómo pasará de un valor cercano a 1 a un valor cercano a 0, no nos tenemos que preocupar por los errores de redondeo. Optamos por una solución similar, inspirada en este concepto. Almacenamos en un valor lógico (al que llamo masDeMitad) si t > 0.5. La primera vez que se de t <= 0.5 (cuando pasa de 1 a 0), será el momento en el que moveremos el cuadrado al nuevo lugar.

Esto se puede implementar en geogebra con el comando Si, que evalúa si una condición es cierta y ejecuta un comando en consecuencia. El único inconveniente de esto es que la condición se tendrá que escribir como Si (masDeMitad && t < 0.5 … , …), lo cual es un poco engorroso.

Además, no hay una forma cómoda de ejecutar varias instrucciones con un único Sí. Como voy a tener que cambiar varios valores en una única condición, defino el valor lógico variosPasos, para usarlo de esta forma

```
Si(condicion, Valor(variosPasos,True))
Si(variosPasos, paso 1)
Si(variosPasos, paso 2)
…
Valor(variosPasos, false)
```

De esta forma, puedo ejecutar varias instrucciones según una misma condición lógica.

Nos queda explicar como conseguir el movimiento en espiral. Para mover el cuadrado, lo más sencillo que podemos hacer es aplicarle una translación por el vector (tx, ty). Si cambiamos los valores de tx y de ty acorde a una espiral, tendremos el movimiento deseado.

Para eso defino cuatro valores lógicos, uno por cada caso a considerar, y todos empiezan con valor false. Si estoy en una esquina, cambio el valor Ílogico para indicar que el tipo de traslación a añadir es en otra dirección.

Por último, defino un número $p \in \{-1, 1\}$, que denota la paridad de la suma de las componentes del vector de traslación con sus coordenadas divididas a la mitad. Esto es un detalle técnico para hacer que se vea más bonita la obra.

En la siguiente tabla, añado el código de scripting GeoGebra que utilicé. Con comentarios para ayudar a la comprensión. Nótese que separo en dos la parte en la que elijo en qué situación estoy y en la que doy el paso, por simplificar la complejidad del script.

```
#Por errores de coma flotante, es mejor detectar el salto como pasar de la segunda mitad a la primera

#detectar estoy en segunda mitad
Si(!masDeMitad && t > 0.5,Valor(masDeMitad,true))

#elegir la siguiente tesela cuando esté en la primera mitad:

#fila superior hacia la derecha termina
Si(masDeMitad && t < 0.5 && (tx == ty) &&supDcha,Valor(variosPasos,true))
Si(variosPasos,Valor(supDcha,false))
Si(variosPasos,Valor(dchaAbajo,true))
Valor(variosPasos,false)

#columna derecha hacia abajo termina
Si(masDeMitad && t < 0.5 && (tx == -ty) &&dchaAbajo,Valor(variosPasos,true))
Si(variosPasos,Valor(dchaAbajo,false))
Si(variosPasos,Valor(inflzda,true))
Valor(variosPasos,false)

#fila inferior hacia la izquierda termina
Si(masDeMitad && t < 0.5 && (tx == ty) &&inflzda,Valor(variosPasos,true))
Si(variosPasos,Valor(inflzda,false))
Si(variosPasos,Valor(izdaArriba,true))
Valor(variosPasos,false)

#columna izquierda hacia arriba termina
Si(masDeMitad && t < 0.5 && (-tx == ty+2) &&izdaArriba,Valor(variosPasos,true))
Si(variosPasos,Valor(izdaArriba,false))
#nos preparamos para dar un salto
Valor(variosPasos,false)
```

```
#elegimos el paso a dar

#paso a la dcha
Si(masDeMitad && t < 0.5 && supDcha, Valor(tx, tx+2))
#paso abajo
Si(masDeMitad && t < 0.5 && dchaAbajo, Valor(ty, ty-2))
#paso a la izda
Si(masDeMitad && t < 0.5 && inflzda, Valor(tx, tx-2))
#paso arriba
Si(masDeMitad && t < 0.5 && izdaArriba, Valor(ty, ty+2))

#el 0,0 es un caso especial
Si(masDeMitad && t < 0.5 && tx == 0 && ty == 0, Valor(variosPasos,true))
Si(variosPasos, Valor(tx,-2))
Si(variosPasos,Valor(ty,2))
Si(variosPasos,Valor(supDcha,true))
Valor(variosPasos,false)

#Salto hacia el siguiente nivel
Si(masDeMitad && t < 0.5 &&!supDcha && !inflzda && !dchaAbajo && !izdaArriba,
Valor(variosPasos,true))
Si(variosPasos, Valor(tx, tx-2))
Si(variosPasos, Valor(ty, ty + 4))
Si(variosPasos, Valor(supDcha,true))
Valor(variosPasos,false)

#paridad del giro por efecto visual
Si(mod(abs(tx+ty)/2,2) == 0, Valor(p,1), Valor(p,-1))

#marcamos que ya hemos dado el paso
Si(masDeMitad && t < 0.5, Valor(masDeMitad,false))
```