PRG (ETS de Ingeniería Informática) - Curso 2019-2020

Práctica 1. Resolución recursiva del dibujo de una figura

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Índice

1. `RSquare-A`. Definición recursiva

Una `RSquare` es una figura geométrica plana que se obtiene mediante la repetición de un patrón básico: un cuadrado, en cuyas cuatro esquinas se pueden disponer otros cuadrados de lado una fracción del original. A su vez, en las esquinas de estos cuadrados más pequeños se pueden disponer otros cuadrados todavía más pequeños, y así sucesivamente. El número de veces $n \geq 1$ que se repite este patrón en el dibujo se denomina el *orden* de la figura.

Por sencillez, estas figuras se van a clasificar por la forma en que dichos cuadrados se superponen. De momento, se considerará el tipo de figura denominado `RSquare-A`, que se define recursivamente de la forma siguiente:

- Una `RSquare-A` de *orden* 1, lados de *longitud* l y *centro* en el punto (x, y), es un cuadrado, con reborde resaltado[1], de lado l y *centro* en (x, y).
- Una `RSquare-A` de *orden* $n > 1$ y *longitud* l, con *centro* en el punto (x, y), es una figura similar a la de orden 1 pero que tiene, por debajo, en cada una de sus cuatro esquinas, una figura `RSquare-A` de *orden* $n - 1$, *longitud* $l/2$ y *centro* en dichas esquinas.

  El cuadrado central se dibuja superpuesto a las subfiguras de las esquinas.

Por ejemplo, en la figura 1 se pueden ver unas figuras `RSquare-A` de *orden* 1, 2 y 3, todas ellas de la misma longitud l.

[1] Esto es, coloreado de forma distinta al del resto del cuadrado.

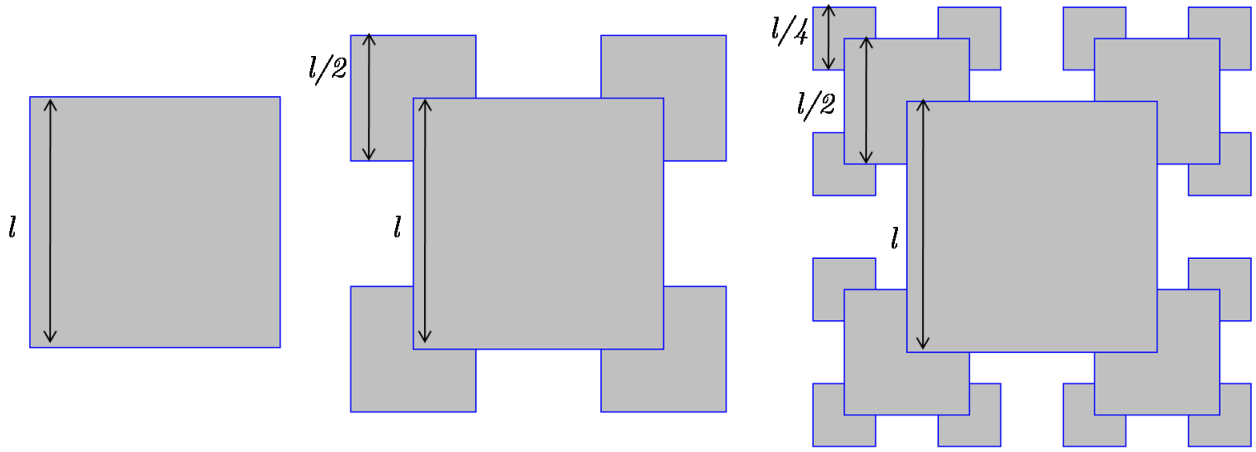


Figura 1: Figuras `RSquare-A` de longitud l y orden 1, 2 y 3, respectivamente.

Notar que, siguiendo la definición anterior, el dibujo de una figura de este tipo de *orden* 4 (por ejemplo) supone el dibujo de cuatro figuras de *orden* 3 (y longitud la original dividida por 2), situadas cada una de ellas en los extremos de la figura original y, a su vez, por aplicación recursiva, el dibujo de cada una de las de *orden* 3 implicará el de cuatro figuras de *orden* 2 (de longitud dividida) y así, sucesivamente, hasta el caso base (figura de *orden* 1).

2. Dibujo de una `RSquare-A` con la librería `Graph2D`

En esta práctica se debe escribir una clase Java que permita dibujar figuras `RSquare-A` de *orden* 1 en adelante. Esta tarea se simplifica con el uso de la librería gráfica `Graph2D` (usada ya en IIP), dado que permite definir ventanas gráficas, del tamaño que se requiera, y dibujar en ellas rectángulos y otras figuras geométricas en la ubicación y con las dimensiones que se especifiquen.

A la hora de escribir el código, se tendrán en cuenta las siguientes consideraciones:

- Por las características de la definición de `RSquare-A`, una figura de orden n (por elevado que sea este) y de lado l, se inscribe en un cuadrado de tamaño acotado de lado

$$l + \frac{l}{2} + \frac{l}{2^2} + \cdots + \frac{l}{2^{n-1}} = 2l - \frac{l}{2^{n-1}}$$

  y que en el límite vale $2l$. [2]

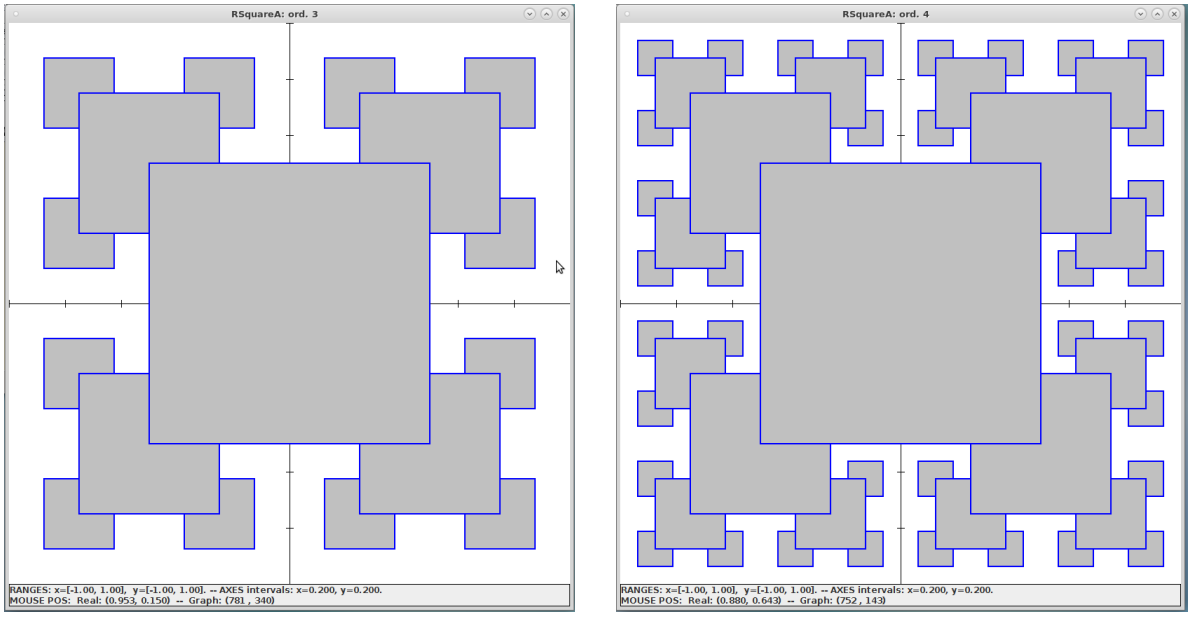
  Por simplificar, aunque no es estrictamente necesario, se considerará que la figura de mayor orden que el programa vaya a dibujar, estará centrada en el punto $(0, 0)$ de coordenadas de la ventana, y que su cuadrado central tendrá longitud unitaria. Este es el criterio seguido en los dibujos de la figura 2.

[2] El sumatorio $l + \frac{l}{2} + \frac{l}{2^2} + \cdots + \frac{l}{2^{n-1}}$ es la suma de los n primeros términos de una serie geométrica en la que l es el primer término y $\frac{1}{2}$ es la razón común. Así, $\sum_{k=0}^{n-1} l\ \frac{1}{2^k} = l\ \frac{1 - \frac{1}{2^n}}{1 - \frac{1}{2}} = l\ \frac{2(2^n - 1)}{2^n} = 2l - \frac{l}{2^{n-1}}$

Según el resultado anterior, ello implica que si se va a dibujar una figura de lado unitario, basta con que la ventana de dibujo se cree con unas dimensiones que estén incluidas tanto en abscisas como en ordenadas entre -1 y $+1$ (un cuadrado de lado 2). Un ejemplo del resultado se muestra en la figura 3.

- El dibujo de un cuadrado con reborde se realiza dibujando primero el cuadrado relleno (método `fillRect` de la librería gráfica) y después el mismo sin relleno (método `drawRect`). Notar que, tal y como indica su documentación, estos métodos deben recibir las coordenadas de la esquina superior izquierda del rectángulo.
- Para que todas las partes de la figura se dibujen recursivamente en la misma ventana, se implementará un método que reciba como parámetro la ventana de dibujo, además del orden, la dimensión y la ubicación.

  Adicionalmente, en la misma clase se escribirá otro método que tenga como parámetros únicamente el orden de la figura, y que se encargue de crear la ventana de tamaño suficiente, y de invocar al método anterior para desencadenar el cálculo del dibujo de la figura con los datos iniciales preestablecidos: lado de longitud 1 y centro en $(0, 0)$.



(a) `RSquare-A de orden 3.` (b) `RSquare-A de orden 4.`

Figura 2: Figuras `RSquare-A` de orden 3 y 4, respectivamente, ambas de longitud 1, centradas en una ventana de tamaño 2×2.

3. Actividades en el laboratorio

3.1. Actividad 1: proyecto `prg`. Instalación de la librería gráfica

a) Crea un proyecto *BlueJ* de nombre `prg` en `$HOME/DiscoW`. De ahora en adelante, en este proyecto se desarrollará el código de las prácticas de la asignatura.

b) Crea dentro del proyecto un paquete `pract1` para la primera práctica.

c) Descarga, de la carpeta *PRG:recursos/Laboratorio/Práctica 1* de *PoliformaT*, la clase `RSquare` y agrégala al paquete `pract1`.

d) La librería gráfica (clase `Graph2D` del paquete `graph2D`) se facilita como una librería en el fichero `graphLib.jar` y su documentación se proporciona en el fichero `docGraph2D.zip` (en la carpeta *PRG:recursos/Laboratorio/Librería gráfica* de *PoliformaT*).

   Se debe cargar esta librería de la manera adecuada (indicando dónde se encuentra el fichero `jar`, que se habrá descargado previamente). Si cursaste la asignatura IIP en el primer semestre de este curso, ya la tendrás cargada y, por tanto, puedes obviar los pasos d.1) y d.2) descritos a continuación. Si no la cursaste:

   d.1) Sitúa el fichero `graphLib.jar` en el proyecto de prácticas `prg` y, desde el menú *Preferencias/Librerías* de *BlueJ*, opción *Add File*, añade el fichero `graphLib.jar`. Tendrás que arrancar nuevamente *BlueJ* para que la librería se cargue.

   d.2) Descomprime el fichero `docGraph2D.zip` en el proyecto `prg` para poder consultar la documentación de la clase `Graph2D` (fichero `Graph2D.html`) cuando sea necesario.

e) Descarga, de la carpeta *PRG:recursos/Laboratorio/Práctica 1* de *PoliformaT*, la clase `FigRecSimple` y agrégala al paquete `pract1`. Esta clase es un ejemplo de uso de la librería gráfica en el dibujo de una figura recursiva.

3.2. Actividad 2: implementación del cálculo de una `RSquare-A`

a) Completa el código del siguiente método auxiliar en la clase `RSquare`:

```
/** Dibuja en la ventana gd un cuadrado sólido de color gris y enmarcado
 *  en azul, con centro en (cX, cY) y lado l.
 */
public static void drawCentSquare(Graph2D gd, double cX, double cY, double l)
```

   Una vez implementado, para probarlo, crea en el *code pad* un `Graph2D` con el constructor por defecto (rango de abscisas $[-1.0, 1.0]$, ídem de ordenadas) y ejecútalo para que dibuje en la ventana creada un cuadrado con centro en $(0, 0)$ y de lado 1. Pruébalo con otras ubicaciones y dimensiones del cuadrado.

   Nota que este método invoca a un método privado auxiliar `delay()` ya implementado en la clase, que se encarga de retrasar lo suficiente el dibujo del cuadrado. Con ello se pretende que, en ejecución, se pueda observar a simple vista el orden en que se van superponiendo los cuadrados que constituyen la figura.

b) Escribe un método recursivo con perfil:

```
/** Dibuja en la ventana gd una RSquare-A de orden n >= 1, con centro
 *  en (cX, cY) y cuadrado central de lado l.
 */
public static void rSquareA(Graph2D gd, int n, double cX, double cY, double l)
```

c) Escribe un método guía o lanzadera, con perfil:

```
/** Dibuja una RSquare-A de orden n >= 1, longitud 1 y centrada en (0, 0). */
public static void rSquareA(int n)
```

que cree una ventana en el rango de abscisas $[-1.0, 1.0]$ y de ordenadas $[-1.0, 1.0]$, y título `"RSquareA: ord. " + n`, y que invocando al método anterior dibuje en la ventana la figura de orden n (con cuadrado central de lado 1 y centro en $(0, 0)$).

d) Prueba la ejecución de estos métodos con órdenes n igual a 3, 4 y 7 y observa que el resultado coincida con el de las figuras 2(a), 2(b) y 3, respectivamente.

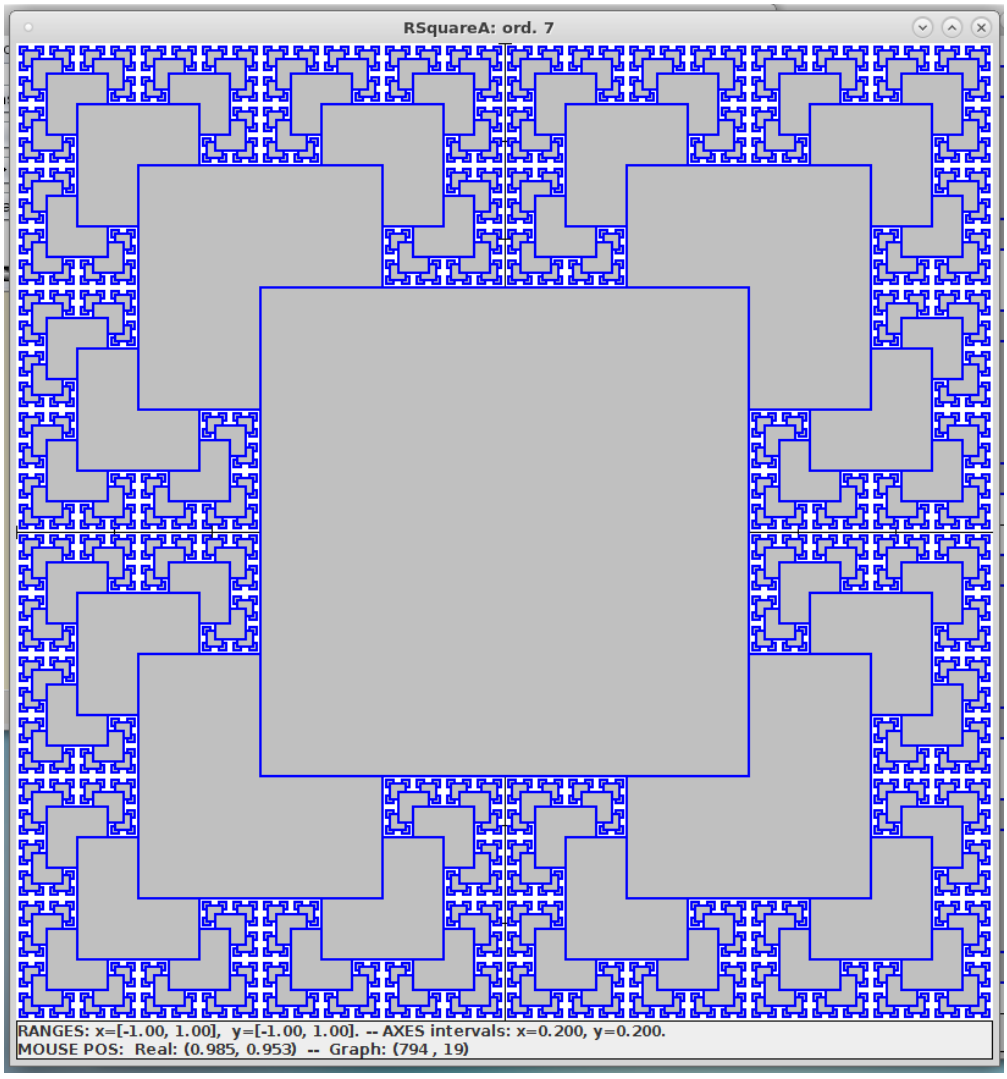
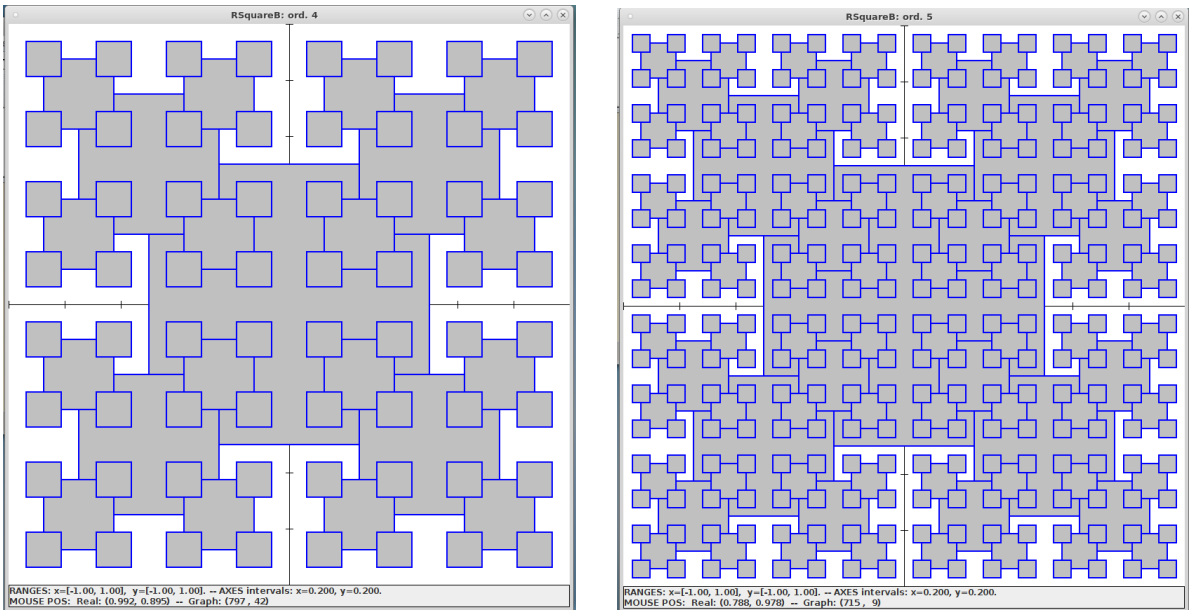


Figura 3: La `RSquare-A` de orden 7 y lado l ocupa algo menos de un cuadrado de lado $2l$.

3.3. Actividad 3: implementación del cálculo de una `RSquare-B`

Como se ha comentado en el primer apartado, las figuras `RSquare` se pueden clasificar según la forma en que los cuadrados se superpongan entre si. En la figura 4 se muestra otro tipo de `RSquare`, denominado `RSquare-B`, en la que los cuadrados menores van siempre superpuestos sobre los mayores, lo que da origen a una distribución distinta.

Estas figuras tienen una definición recursiva análoga a la de una `RSquare-A`, excepto que, en el caso general, primero hay que dibujar el cuadrado de mayor tamaño, y luego superponerle las subfiguras de menor orden. Con ello, en el dibujo resultante quedarán al frente los cuadrados más pequeños: los dibujados en las llamadas al caso base, como se muestra en los ejemplos de la figura 4.



(a) `RSquare-B` de orden 4. (b) `RSquare-B` de orden 5.

Figura 4: Figuras `RSquare-B` de orden 4 y 5, respectivamente.

En esta actividad, deberás añadir a la clase los métodos recursivos análogos a los de la Actividad 2, y hacer las pruebas correspondientes:

```
/** Dibuja en la ventana gd una RSquare-B de orden n >= 1, con centro
 *  en (cX, cY) y cuadrado central de lado l.
 */
public static void rSquareB(Graph2D gd, int n, double cX, double cY, double l)

/** Dibuja una RSquare-B de orden n >= 1, longitud 1 y centrada en (0, 0). */
public static void rSquareB(int n)
```