INNODOCT 2019
Valencia, 6th-8th November 2019
DOI: http://dx.doi.org/10.4995/INN2019.2019.10130

Una librería gráfica para la enseñanza de la Programación en primeros cursos universitarios

Assumpció Casanova Faus[a], Francisco Marqués Hernández[b]
Universitat Politècnica de València, València, Spain. [a]casanova@dsic.upv.es, [b]pmarques@dsic.upv.es

Resumen

En la enseñanza de la Programación en primer curso universitario, viene siendo habitual utilizar lenguajes orientados a objetos, como Java, C++ o Python. Desafortunadamente, suelen contar con una intefaz gráfica relativamente compleja que precisa el uso de elementos avanzados de la Programación Orientada a Objetos (herencia, genericidad, ...). Por ello presentamos una herramienta orientada a la Programación en Java que simplifica la realización de gráficos, abstrayéndola de elementos irrelevantes desde el punto de vista de la enseñanza básica de la Programación. Así, podemos plantear la resolución de problemas habituales en estos primeros cursos de una forma más cercana a un alumnado habituado a interactuar gráficamente con los dispositivos que usa cotidianamente.

Palabras clave: *Enseñanza básica de Programación. Java. Librería gráfica.*

1. Introducción

En las asignaturas de Programación de primer curso del Grado en Ingeniería Informática de la Universitat Politècnica de València, hemos detectado la necesidad de conjugar la realización de prácticas y ejercicios abarcables en primer curso, con la obtención de aplicaciones que despierten interés. Estas asignaturas, *Introducción a la Informática y a la Programación* y *Programación*, se imparten en dos cuatrimestres consecutivos, y cubren los tópicos desarrollados en el libro de texto (Prieto, Casanova, Marqués, Llorens, Galiano, Gómez, González, Martínez-Hinarejos, Moltó, y Piris, 2016). La matrícula es del orden de 500 alumnos, de nivel heterógeneo, y en gran parte con relativamente bajos conocimientos de Programación; en cambio, como usuarios están habituados a interfaces sofisticadas.

La herramienta que presentamos[1] se ha desarrollado, en forma de librería, para ser usada fundamentalmente en la realización de prácticas de Programación en Java, lenguaje que se

[1] Con el soporte financiero de la Escola Tècnica Superior d'Enginyeria Informàtica de la Universitat Politècnica de València.

usa en la impartición de estas asignaturas. Además de aprovechar las ventajas docentes que aporta el uso de esta librería, no es menos importante el atractivo del resultado obtenido, lo que nos consta que puede estimular el trabajo de nuestros alumnos.

2. La librería gráfica

2.1. Objetivos y características

Está comúnmente aceptado que el uso de gráficos puede ser útil para la enseñanza básica de la Programación. En términos de representaciones gráficas se pueden plantear problemas cuya resolución exige conocimientos y destrezas propios de los cursos iniciales de esta materia. Además, con su uso, dichos problemas pueden presentarse de forma más clara y atractiva que con una mera representación textual (por ejemplo, ver Fig. 1).

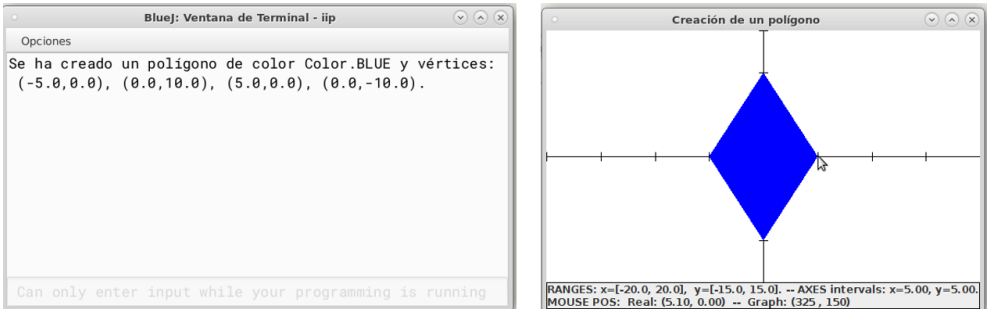

Fig. 1. Salida de texto de un dato de un programa y salida gráfica del mismo dato del programa.

En el aprendizaje de la Programación en Java resulta difícil el uso de representaciones gráficas, particularmente para los alumnos primerizos. Ello se debe a que es preciso conocer aspectos avanzados de la Programación Orientada a Objetos (herencia, genericidad) y de la gestión de eventos, que no se estudian en los cursos de introducción. El entorno de programación *BlueJ* que usamos en prácticas, disponible en www.bluej.org y ampliamente difundido en la enseñanza de la Programación en Java, facilita una entrada de datos amigable; pero, a menos que se usen ciertas librerías de Java (AWT y Swing) difíciles de dominar para un principiante, el alumno debe limitarse a mostrar el resultado de sus programas en la salida de texto estándar, como en el ejemplo de la Fig.1 izquierda.

Además, el modelo de imagen 2D propio de Java (Arnold y Gosling, 2005), obliga a tener en cuenta aspectos de transformación de espacios de representación (espacio cartesiano de representación del problema frente a espacio gráfico real de la pantalla, basado en píxeles) al tiempo que no facilita el uso de persistencia, dificultando una representación tipo "Geometría de Tortuga" como la que ofrece el lenguaje Logo (Abelson y diSessa, 1986).

Con todo ello, los principales objetivos que guiaron la realización de la librería gráfica presentada son los siguientes:

- Evitarle a los alumnos principiantes en Java la complicación del uso prematuro de librerías tales como `java.awt` y como `javax.swing`.
- Abstraer los problemas de transformación de espacios asociados a la representación gráfica, de forma que no se tenga que pensar en términos de espacios de píxeles, sino en representaciones cartesianas en el espacio dominio del problema a resolver.
- Permitir la interactuación con el espacio gráfico mediante operaciones puramente cartesianas (por ejemplo, para trazar un segmento de recta es necesario proporcionar las coordenadas de sus extremos) o tipo plóter, en las que se tiene en cuenta el estado interno de la representación y para el trazado de una recta basta con indicar cuánto hay que avanzar o girar, desde la posición y ángulo anterior.
- Desarrollar algunos métodos, de funcionamiento sencillo, para ser capaces de interactuar algo con un ratón sin tener que manejar un modelo de eventos.
- Facilitar una interfaz de usuario de la librería lo más similar posible al de las librerías gráficas 2D estándar del lenguaje. Procurando así que la transición en el uso de una a otra, caso de producirse más adelante, sea sencilla para el alumno.
- Proporcionar una documentación exhaustiva de la librería, siguiendo los estándares de Java. El relativamente reducido número de clases de la librería simplifica dicha documentación, lo que promueve su uso.

2.2. Descripción de la librería

La librería gráfica consiste en un paquete Java, formado por un grupo de definiciones y clases. Toda la interactuación del programador con la librería se produce a través de dos clases: la clase `Graph2D`, orientada a una representación cartesiana habitual (en la que se describen las posiciones de los elementos a ser dibujados), y la clase `Plotter`, derivada de la anterior, pero con la inclusión de un estado que representa la existencia (posición y ángulo) de un elemento de dibujo similar a un plóter. Adicionalmente, es posible configurar diversos parámetros de la representación tales como el grosor o color de las líneas, la existencia o no de ejes cartesianos y la información asociada a la posición del ratón.

Mediante los métodos de la librería es posible crear ventanas en las que se pueden dibujar elementos gráficos planos, tales como puntos, rectas, poligonales, arcos y óvalos. Los elementos pueden mostrarse con colores diferentes (los de un espacio RGB), varios gruesos de línea y pueden ir o no rellenos (pintados). Adicionalmente, permite el dibujo de cadenas de caracteres `String` (de diferentes tipos, tamaños y colores).

Uno de los aspectos fundamentales de la librería es que abstrae el espacio de representación sobre el que se desea dibujar, de la proyección concreta que se tiene que realizar para trazar un gráfico en la pantalla del ordenador. En todo momento se trabaja sobre un espacio cartesiano definido por el programador (alumno), sin tener que realizar traslaciones al espacio de píxeles dado por los componentes gráficos del lenguaje. Las dimensiones de la representación se definen por el usuario en el momento de la creación del objeto ventana de dibujo. Esta estrategia facilita mucho la representación de los problemas ya que se obvian las transformaciones al espacio de píxeles subyacentes. Además, la librería está diseñada para que las ventanas sean reescalables sin tener que calcular explícitamente el programador las transformaciones afines asociadas, ya que se calculan automáticamente.

Cabe indicar que estas dos características, junto con la posibilidad de interactuar de manera simple con el ratón, la diferencian de otras librerías, de propósito docente similar, usadas en otras universidades como Princenton (Sedgewick y Kevin, 2008) y Stanford (Roberts, Picard y Fredricsson, 1998).

En resumen, el usuario de la librería se limita a definir y configurar la ventana de dibujo con muy pocas líneas de código, para a continuación trazar en ella los gráficos que desee.

En el siguiente ejemplo se dibuja un cuadrado centrado en (0,0) con el color por defecto (azul), y mostrando sus diagonales en color rojo. La ventana se ha creado con las dimensiones y sistema de coordenas por defecto (ver el resultado en la Fig. 2 izquierda). Aunque algo abreviado, puede comprobarse la simplicidad del código utilizado:

```
Graph2D gd = new Graph2D();
// Draw a rectangle:
gd.drawRect(-0.5, 0.5, 1.0, 1.0);
// Change the drawing color:
gd.setForegroundColor(Color.RED);
// Draw the diagonals:
gd.drawLine(-0.5, -0.5, 0.5, 0.5);
gd.drawLine(-0.5, 0.5, 0.5, -0.5);
```

En la Fig. 2 derecha se muestra otro ejemplo más elaborado, en el que se ha usado el método `drawLine` para dibujar una espiral nautilus mediante sucesivos segmentos de recta de longitud creciente.

La clase `Plotter` de la librería facilita la realización de gráficos como la *Figura de Koch*, *SnowFlake* o *Copo de nieve*, de la Fig. 3. La realización de esta figura puede efectuarse, de forma bastante rebuscada, calculando sus vértices en el plano cartesiano. Pero resulta mucho más simple si se realiza en términos similares a los de un plóter, mediante los métodos de la clase que realizan el avance y rotación de la pluma.

El siguiente código traza un lado de la figura de orden *n*, e ilustra el uso de estos métodos:

```
static void side(Plotter p, int n, double l) {
    if (n == 0) { p.forward(l); }
    else {
        double l3 = l / 3;
        side(p, n - 1, l3);
        p.rotate(60); side(p, n - 1, l3);
        p.rotate(-120); side(p, n - 1, l3);
        p.rotate(60); side(p, n - 1, l3);
    }
}
```

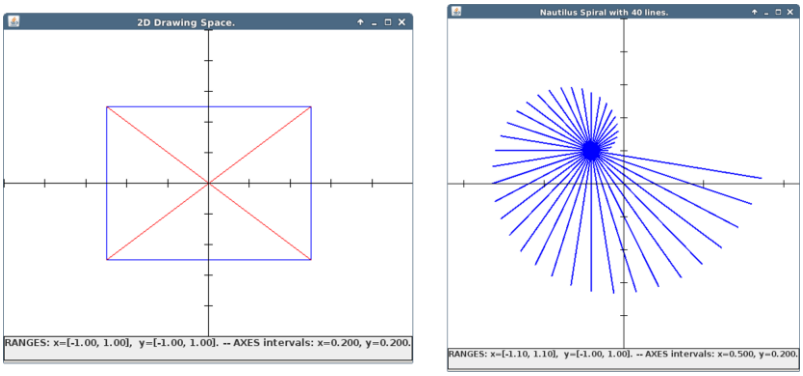

Fig. 2 Un cuadrado y sus diagonales. Una espiral nautilus dibujada con segmentos de recta.

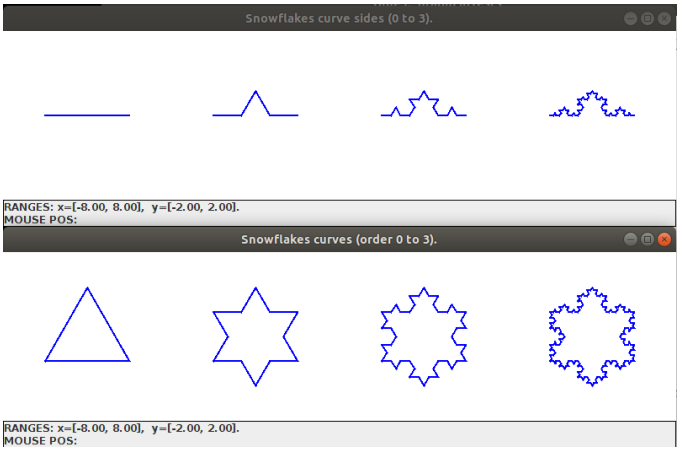

Fig. 3. Lados de una figura SnowFlake de orden 0 a 3 dibujados con el método `side`*, y las correspondientes figuras completas de los mismos órdenes, obtenidas por yuxtaposición de tres lados convenientemente girados.*

Finalmente, cabe destacar el funcionamiento del método de captura de clics de ratón definida en la librería. Este método actúa de forma análoga a la captura de la entrada de texto de teclado por métodos predefinidos de Java, y abstrae al alumno de la gestión de eventos: el método espera a que se pulse sobre la ventana gráfica, obteniendo la abscisa y la ordenada del clic. El siguiente ejemplo muestra la simplicidad de su uso:

```
System.out.println("Please, click on the image.");
double[] coord = gd.nextMousePressed(); // Cartesian coordinates of the click
```

2.3. Instalación y uso de la librería

La librería se proporciona como un único fichero compromido que el alumno puede descargarse libremente junto con la documentación `html`. El fichero puede residir donde desee el usuario (puede ser de acceso común para los alumnos, o cada usuario puede utilizarlo de forma independiente en su equipo particular). Para su uso, basta con dejar indicada la ruta del fichero, de la forma habitual en el entorno de programación que se use.

La librería se ha construido utilizando solamente Java estándar, por lo que puede ser usada inmediatamente en cualquier sistema: Windows, MacOSX y, especialmente, Linux, en el que se efectúan nuestras prácticas. Incluso puede usarse en las Raspberry PI.

3. Aplicación en las asignaturas de primer curso

La simplicidad de uso de la librería presentada nos ha permitido utilizarla, estos últimos años, en las prácticas de laboratorio de primer curso y en la realización de entregables, pudiendo experimentar sus ventajas docentes sin el coste de tener que introducir temas avanzados.

Además de que el uso de gráficos mejora la presentación de resultados y facilita su interpretación, por sí mismo proporciona un conjunto de problemas y metodologías con interés propio, y que caen dentro de los tópicos que se cubren en las prácticas. Estos son principalmente: la iteración, la recursión y el uso de arrays en estructuras de datos.

Un ejemplo de problema en un curso inicial de Programación consiste en implementar el cálculo de funciones mediante métodos iterativos, como recurrencias y desarrollos en serie; a continuación se comprueba su corrección tabulando las funciones, respetando ciertas reglas de formato del listado asociado. La representación gráfica de los resultados proporciona una variación interesante de la presentación tabulada, y puede efectuarse dibujando puntos sucesivos, o bien mediante segmentos conectados entre sí.

Así, en una práctica de iteración se proponía la implementación de las funciones raíz cuadrada y logaritmo. Una vez obtenidas (y comparadas con las del propio lenguaje), los alumnos pasaban a graficarlas, usando nuestra librería para dibujar segmentos de unión entre puntos consecutivos de la tabulación. El resultado se puede ver en la Fig. 4.

En las prácticas asociadas al tema de recursión, se plantearon unos primeros problemas de recursión lineal sobre datos numéricos y sobre cadenas de caracteres. Sin embargo, no es fácil encontrar problemas de recursión múltiple indicados para un nivel de primer curso. No obstante, si se considera el dibujo de fractales, es posible encontrar abundancia de figuras en las que se da un patrón recursivo múltiple, y al mismo tiempo fácil de descubrir y programar.

Por ejemplo, en el capítulo 2.3 de (Sedgewick y Kevin, 2008) se propone como ejercicio la realización del fractal de la Fig. 5., en sus dos variaciones de recursión no final y final. Este problema se abordó en una práctica de laboratorio usando nuestra librería, dibujando como motivo básico de la figura un cuadrado sólido, perfilado con un cuadrado de otro color.

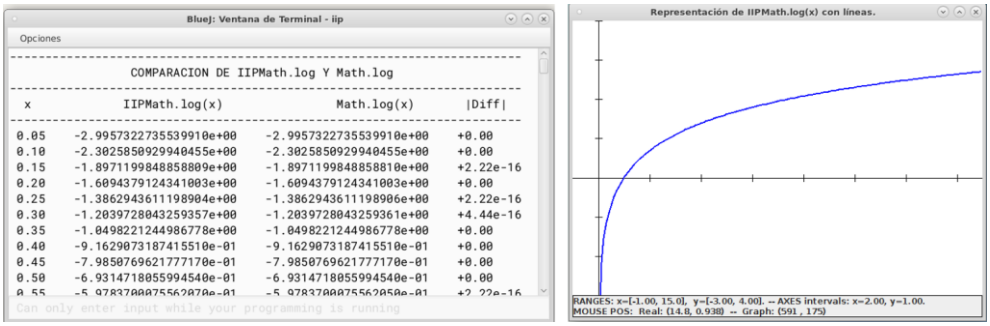

Fig. 4. Tabulación del logaritmo natural y representación gráfica de los valores de la tabla, unidos con líneas.

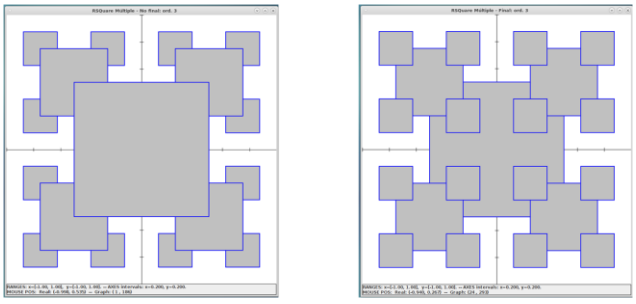

Fig. 5. Figuras fractales obtenidas mediante recursión multiple, no final y final.

Está ampliamente reconocida en la comunidad docente la necesidad de hacer trazas de los algoritmos recursivos, tanto en clase de teoría como en el laboratorio (Tucker, 1991;

Hazzan, Lapidot y Ragoni, 2011). El depurador del lenguaje es una herramienta excelente para seguir la traza de una recursión lineal, pero su uso es menos efectivo para capturar el funcionamiento de la recursión múltiple. Para subsanarlo, en la práctica del fractal se sacaba partido de la posibilidad de animar la figura: cada vez que se dibuja un cuadrado, se usa un método que produce un retraso de décimas de segundo. Así, el ojo humano puede percibir el orden en que se van construyendo recursivamente las subfiguras del dibujo.

Finalmente, otro de los objetivos primordiales de nuestras asignaturas consiste en dominar los algoritmos básicos de manipulación de arrays, y su aplicación a la estructuración de tipos de datos sencillos; para su mejor comprensión, en el laboratorio desarrollamos ejemplos concretos de cierto interés práctico. Aunque el entorno *BlueJ* proporciona herramientas para examinar los datos de estos tipos y ver cómo afectan los métodos a su estado, casi siempre es conveniente desarrollar algún programa de aplicación, más o menos costoso de elaborar, y para el que no siempre la salida estándar muestra con eficacia su comportamiento.

El ejemplo de la Fig. 6 corresponde a una práctica en la que se manejaba mediante un array un grupo de polígonos, elementos que se pueden dibujar en una ventana de la librería, superpuestos por su orden en el array. En un ejemplo como este, realizar una aplicación completa excede las limitaciones en cuanto a objetivos y tiempo de una asignatura de primer curso; no obstante, el uso de la librería gráfica permite escribir un sencillo programa de prueba en el que, creado un grupo, se visualice su estado antes y después de aplicarle las operaciones implementadas, comprobando así sus posibilidades y correcto funcionamiento.

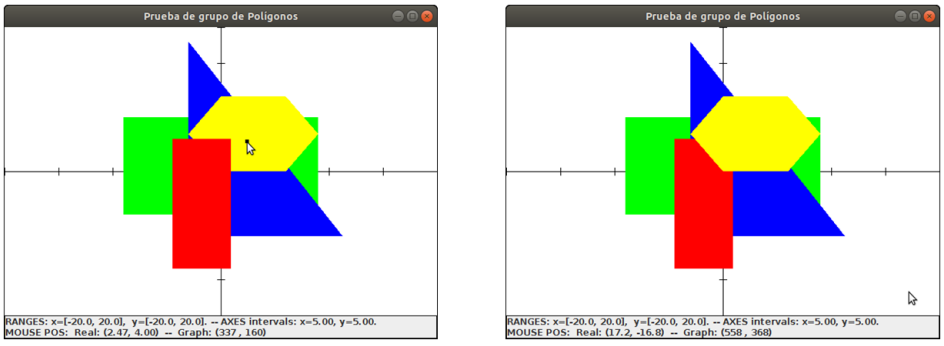

Fig. 6. Prueba en un grupo de polígonos de la ejecución de traer al frente un polígono. A la izquierda: selección mediante un clic del polígono a mover. A la derecha: estado resultante.

Referencias

Abelson, H. y diSessa., A. (1986). Geometría de tortuga. El ordenador como medio de exploración de las matemáticas. Madrid: Anaya.

Tucker A. B. (editor and co-chair). (1991). Computing Curricula 1991. Report of the ACM/IEEE-CS Joint Curriculum Task Force. New York: ACM Press, IEEE Computer Society Press.

Arnold, K., Gosling, J. & Holmes, D. (2005). The Java Programming Language, Fourth Edition. Addison Wesley Professional.

Hazzan O., Lapidot, T. & Ragoni, N. (2011). Guide to Teaching Computer Science. An Activity-Based Approach. London: Springer Verlag.

Prieto, N., Casanova, A., Marqués, F., Llorens, M. Galiano, I., Gómez, J.A González, J., Martínez-Hinarejos, C., Moltó, G. y Piris, J. (2016). Empezar a programar usando Java. València: Editorial de la Universitat Politècnica de València.

Roberts, E., Picard, A. & Fredricsson, M. (1998). Designing a Java Graphics Library for CS1. ACM SIGCSE Bulletin, Vol. 30 Issue 3, p. 213-218. New York: ACM.

Sedgewick, R. & Kevin, W. (2008). Introduction to Programming in Java: An Interdisciplinary Approach. Addison Wesley.