Práctica Guiada

Introducción al desarrollo de videojuegos con Java Swing y Java 2D

El objetivo de esta práctica es desarrollar una aplicación que sirva como ejemplo de cómo realizar animaciones usando clases de Java Swing y Java 2D. Para ello, vamos a implementar una animación sencilla que va a consistir en el movimiento de una bola sobre una superficie rectangular, haciendo que rebote cuando choque con alguno de los bordes. El resultado final será útil como base para el desarrollo de videojuegos 2D simples.

Para completar el desarrollo de la aplicación crearemos tres clases: `Main`, `Lienzo` y `Bola`.

La clase `Main` extenderá a la clase `JFrame` para crear la ventana principal de la aplicación y también servirá como punto de entrada a la misma. El código Java siguiente es el resultado de una definición incompleta de esta clase:

```
public class Main extends JFrame {

	private static final long serialVersionUID = 1L;

	public Main(Lienzo lienzo) {
		super("Bola Rebotando");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		lienzo = new Lienzo(1200, 900, 60L);
		setContentPane(lienzo);
		pack();
		setLocationRelativeTo(null);
	}

	public static void iniciar() {
		setVisible(true);
		lienzo.iniciar();
	}

	public static void main(String[] args) {
		SwingUtilities.invokeLater(new Main()::iniciar);
	}

}
```

Como se puede ver, queda pendiente agregar a la ventana principal la superficie donde se llevará a cabo la animación e iniciar ésta después de que dicha ventana se haga visible.

La clase `Lienzo` representará la superficie donde tendrá lugar la animación y, por tanto:

- Extenderá a la clase `JPanel`.
- Definirá la tarea encargada de llevar a cabo la animación.
- Se encargará de la creación de un hilo en el que se ejecutará la tarea anterior, así como la gestión de éste en lo que se refiere a la posibilidad de pausar y reanudar la animación, y a la finalización ordenada de su ejecución.

Comenzamos su definición escribiendo el código Java siguiente:

```
public class Lienzo extends JPanel {

	private static final long serialVersionUID = 1L;
	private long fps;
	private Thread t;
	private boolean pausado;
	private Bola bola;
	private BufferedImage buffer;
```

```
        private Graphics2D g;

        public Lienzo(int w, int h) {
                setPreferredSize(new Dimension(w, h));
                this.fps = fps;

                bola = new Bola(this, (w - 31) / 2, (h - 31) / 2, 45d, 60l);
                buffer = new BufferedImage((int) w, (int) h, BufferedImage.TYPE_INT_ARGB);
                g = (Graphics2D) buffer.getGraphics();
                g.setRenderingHint(
                        RenderingHints.KEY_ANTIALIASING,
                        RenderingHints.VALUE_ANTIALIAS_ON);

        }
}
```

Para lleva a cabo la animación vamos a usar una técnica denominada *rendering activo*, que se describe más adelante. Para evitar ciertos problemas derivados del uso de esta técnica es necesario utilizar un buffer de pantalla intermedio, y por esta razón se crea en el constructor un objeto de tipo `BufferedImage`. Como veremos, cada *frame* de la animación se dibujará en el contexto gráfico de esta imagen para posteriormente volcarla al contexto gráfico del `JPanel`.

A continuación, añadimos a la clase `Lienzo` el método de instancia `ejecutarAnimación`, en el que se implementa la tarea que lleva a cabo la animación. Este método, que se ejecutará en un hilo que vamos a crear más adelante, consiste básicamente en un bucle en el que se realizan las acciones siguientes:

- Medir el tiempo (en nanosegundos) empleado en generar y mostrar cada *frame* de la animación.
- Invocar a los métodos que se encargan de:
    - Generar el siguiente *frame* actualizando la posición de los objetos animados en función del tiempo transcurrido desde que se generó y mostró el *frame* anterior.
    - Dibujar el nuevo *frame*.

La definición de este método se muestra a continuación:

```
private void ejecutarAnimación() {
        long t0 = System.nanoTime();
        long t1;
        long lapso = 0;
        long nanosXframe = 1000000000L / fps;
        while (!Thread.currentThread().isInterrupted()) {
                synchronized (this) {
                        if (pausado) {
                                try {
                                        wait();
                                } catch (InterruptedException e) {
                                        Thread.currentThread().interrupt();
                                }
                                t0 = System.nanoTime();
                        }
                }
                t1 = System.nanoTime();
                lapso += t1 - t0;
                t0 = t1;
                if (lapso >= nanosXframe) {
                        lapso -= nanosXframe;
                        actualizar(nanosXframe);
                        dibujarFrame();
                }
        }
}
```

El método anterior está preparado para que la animación se pueda detener, pausar, reanudar y comprobar si la animación está pausada. Estas acciones la implementamos añadiendo los métodos siguientes a la clase `Lienzo`:

```
public void detener() {
        t.interrupt();
}


public synchronized void pausar() {
        pausado = true;
}

public synchronized void reanudar() {
        if (pausado) {
                pausado = false;
                notify();
        }
}

public synchronized boolean pausado() {
        return pausado;
}
```

El código de los métodos `actualizar` y `dibujarFrame` es el siguiente:

```
private void actualizar(long lapso) {
       bola.mover(lapso);
}

private void dibujarFrame() {
        Graphics2D g = (Graphics2D) getGraphics();
        dibujarFrame(g);
        g.dispose();
}

private void dibujarFrame(Graphics2D gp) {
        // se dibuja el frame en el buffer
        g.setColor(getBackground());
        g.fillRect(0, 0, getWidth(), getHeight());
        bola.paint(g);
        // se vuelca el buffer al panel
        gp.drawImage(buffer, 0, 0, this);
}
```

A continuación, vamos a definir en la clase Lienzo el método que iniciará la animación:

```
public void iniciar() {
        t = new Thread(this::ejecutarAnimación);
        t.start();
}
```

Finalizamos la creación de la clase `Lienzo` sobrescribiendo el método `paintComponent` heredado de la clase `JPanel`:

```
@Override
protected void paintComponent(Graphics g) {
        dibujarFrame((Graphics2D) g);
}
```

Completamos el desarrollo de la aplicación con la creación de la clase Bola:

```
public class Bola {

        private Lienzo lienzo;
        private double x;
        private double y;
```

```
        private int diametro;
        private double vx;
        private double vy;

        public Bola(Lienzo lienzo, double x, double y, int diámetro, double dirección, double velocidad) {
                this.x = x;
                this.y = y;
                this.diametro = diámetro;
                vx = Math.cos(dirección) * velocidad;
                vy = Math.sin(dirección) * velocidad;
                this.lienzo = lienzo;
        }

        public void paint(Graphics2D g) {
                g.setColor(Color.BLACK);
                g.fillOval((int) x, (int) y, diametro, diametro);
        }

        public void mover(long lapso) {
                x += (lapso * vx) / 1000000000L;
                y += (lapso * vy) / 1000000000L;
                if (x + diametro > lienzo.getWidth()) {
                        x = 2 * lienzo.getWidth() - x - 2 * diametro;
                        vx *= -1;
                }
                else if (x < 0) {
                        x = -x;
                        vx *= -1;
                }
                else if (y + diametro > lienzo.getHeight()) {
                        y = 2 * lienzo.getHeight() - y - 2 * diametro;
                        vy *= -1;
                }
                else if (y < 0) {
                        y = -y;
                        vy *= -1;
                }
        }

}
```

Ejercicios de ampliación:

1. Añade en el lugar apropiado el código Java para:
    a. Que la animación se pueda pausar y reanudar mediante la pulsación repetida de la tecla P.
    b. Que la animación se detenga de forma ordenada cuando se cierra la ventanta principal, y antes de que finalice la aplicación.
2. Usa esta práctica como base para crear otra aplicación que permita jugar al Pong.