

Funciones gráficas

Hemos introducido la noción de contexto gráfico en las páginas anteriores, cuando se ha creado el primer applet, que muestra un mensaje. En este capítulo se estudiarán algunas funciones gráficas definidas en la clase *Graphics*, y tres clases que sirven de apoyo para dibujar en el contexto gráfico: la clase *Color* que describe los colores, la clase *Font* que nos permite crear fuentes de texto y la clase *FontMetrics* que nos proporciona sus características.

El contexto gráfico

La función *paint* y *update* nos suministran el contexto gráfico del applet o del componente, en otros casos, hemos de obtener el contexto gráfico del componente mediante la función *getGraphics*. Una vez obtenido el contexto gráfico podemos llamar desde este objeto a las funciones gráficas definidas en la clase *Graphics*.

```
public void paint(Graphics g){
    //usar el contexto gráfico g
}
public void update(Graphics g){
    //usar el contexto gráfico g
}
void funcion(){
    Graphics g=getGraphics();
    //usar el contexto gráfico g
    g.dispose();
}
```

Como vemos en esta porción de código existe una sutil diferencia entre suministrar y obtener el contexto gráfico *g*. Solamente es necesario liberar los recursos asociados al contexto *g*, mediante la llamada a la función *dispose*, cuando se obtiene el contexto gráfico mediante *getGraphics*.

La clase *Graphics* es abstracta por lo que no se pueden crear mediante **new** objetos de esta clase, pero se pueden guardar en una referencia *g* de la clase *Graphics* los contextos gráficos concretos de los distintos componentes.

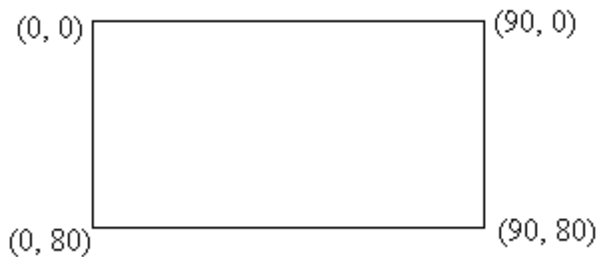
Un contexto gráfico es como la hoja en blanco situada en un trazador (plotter). Para dibujar en dicha hoja se toma una pluma, se dibuja, se toma otra pluma de distinto color o grosor, se dibuja otra porción del gráfico, y así sucesivamente. Cuando no se selecciona explícitamente, se dibuja con una pluma que se establece por defecto.

Las librerías gráficas como la de Windows, disponen de plumas de distinto grosor para dibujar líneas con distintos estilos, brochas para rellenar el interior de una figura cerrada con un color sólido, con una determinada trama o figura, y fuentes de texto, para dibujar texto con distintas fuentes y estilos. La librería gráfica que viene con la versión 1.1 de Java es muy limitada. No hay objetos pinceles, ni brochas. Las líneas tienen un único

grosor y estilo, solamente se pueden cambiar de color, las figuras cerradas solamente se pueden rellenar con un color sólido, y las fuentes de texto disponibles son muy pocas.

La clase *Graphics* describe el contexto gráfico y proporciona un conjunto de funciones para dibujar las siguientes figuras

- Líneas
- Círculos y elipses
- Rectángulos y polígonos
- Imágenes
- Texto



El sistema de coordenadas que se usa en Java es similar a Windows. El área de trabajo del applet está compuesta por una matriz bidimensional de puntos o pixels. Decimos que un punto tiene de coordenadas (x, y) cuando está en la columna *x* medida desde la izquierda, y está en la fila *y*, medida

desde arriba.

La esquina superior izquierda es el origen (0, 0).

La esquina inferior derecha viene determinada por las dimensiones del componente. La función *getSize* nos devuelve un objeto de la clase *Dimension* cuyos miembros *width* y *height* nos suministran la anchura y altura del componente.

```
int ancho=getSize().width;  
int alto=getSize().height;
```

Funciones gráficas

No vamos a examinar completamente la clase *Graphics*, pero si vamos a mostrar mediante un applet el uso de algunas funciones de esta clase. La función *paint* nos va a proporcionar el objeto *g* de la clase *Graphics* que denominamos contexto gráfico del componente (applet). Desde dicho objeto llamaremos a las funciones miembro de la clase *Graphics*.

Establecer un color

El color negro es el color por defecto del contexto gráfico. Para establecer otro color, como veremos en la página siguiente, se utiliza la función *setColor*, y se le pasa un color predefinido o definido por el usuario.

```
g.setColor(Color.cyan);
```

Dibujar una línea

Para dibujar una línea recta se llama a la función *drawLine*, le pasamos el punto inicial y el punto final. Para dibujar una línea diagonal desde el origen (0, 0) o esquina superior izquierda, hasta la esquina inferior derecha, obtenemos las dimensiones del applet mediante la función *getSize*, que devuelve un objeto de la clase *Dimension*. El miembro *width* nos proporciona la anchura y el miembro *height* la altura.

```
g.drawLine(0, 0, getSize().width-1, getSize().height-1);
```

Dibujar un rectángulo

Un rectángulo viene definido por un origen (esquina superior izquierda), su anchura y altura. La siguiente sentencia dibuja un rectángulo cuyo origen es el punto 50, 150, que tiene una anchura de 50, y una altura de 60. La función *drawRect* dibuja el contorno del color seleccionado, y *fillRect* dibuja el rectángulo pintando su interior del color seleccionado, en este caso de color rojo.

```
g.setColor(Color.red);  
g.fillRect(50, 150, 50, 60);
```

Dibujar un arco

Los elipses (oval), arcos (arc), se dibujan en el interior del rectángulo circundante. Una elipse se dibuja mediante *drawOval* o *fillOval*, con los mismos parámetros que el rectángulo. Un arco requiere dos parámetros más el ángulo inicial y el ángulo final. Las sentencias que vienen a continuación, dibujan un arco en el interior del rectángulo cuyo origen es el punto 10, 10, cuya anchura es 150, y cuya altura es 100. El ángulo inicial es 0 y el ángulo final es 270, expresado en grados.

```
g.setColor(Color.cyan);  
g.fillArc(10, 10, 150, 100, 0, 270);  
g.setColor(Color.black);  
g.drawArc(10, 10, 150, 100, 0, 270);
```

Dibujar un polígono

Para dibujar un polígono, se requieren un array de puntos. Un polígono y una polilínea son parecidos, el primero es una figura cerrada mientras que una polilínea es un conjunto de segmentos. Para formar un polígono a partir de una polilínea se une el punto inicial y

el punto final. El polígono precisa de un array de abscisas x , un array de ordenadas y , y la dimensión del array.

```
int[] x={100, 150, 170, 190, 200};
int[] y={120, 280, 200, 250, 60};
g.setColor(Color.blue);
g.drawPolygon(x, y, x.length);
```

Alternativamente, se puede usar un objeto de la clase *Polygon*, al cual se le añaden puntos mediante la función miembro *addPoint*.

```
Polygon poligono=new Polygon();
poligono.addPoint(100, 120);
poligono.addPoint(150, 280);
poligono.addPoint(170, 200);
poligono.addPoint(190, 250);
poligono.addPoint(200, 60);
```

Para dibujar el polígono con su interior pintado del color seleccionado se llama a la función *fillPolygon* y se le pasa el objeto *poligono* de la clase *Polygon*.

```
g.setColor(Color.yellow);
g.fillPolygon(poligono);
```

Veremos en el applet un polígono cuyo contorno está dibujado en azul y su interior en amarillo.

Dibujar una imagen

Para dibujar una imagen se requieren dos pasos:

- Cargar la imagen y crear un objeto de la clase *Image*
- Dibujar dicho objeto en el contexto gráfico

Para crear una imagen u objeto *disco* de la clase *Image* a partir del archivo disco.gif se usa la función *getImage*. Le hemos de indicar la ubicación de dicho archivo relativa a la página web que contiene el applet o al código compilado. En nuestro caso, hemos situado la imagen en el mismo subdirectorío que la página web que contiene al applet. El lugar más adecuado para cargar la imagen es en la función *init*, ya que como se ha mencionado se llama una sola vez.

```
public void init(){
    disco=getImage(getDocumentBase(), "disco.gif");
}
```

Para dibujar la imagen en el contexto gráfico *g*, se llama a la función *drawImage*. Hay varias versiones de esta función, la más simple es aquella a la que se le proporciona el objeto *disco* de la clase *Image*, las coordenadas de su esquina superior izquierda (250, 50), y el observador, el propio applet o *this*.

```
g.drawImage(disco, 250, 50, this);
```

Si deseamos obtener las dimensiones del objeto *disco* de la clase *Image*, llamamos a dos funciones de esta clase *getWidth* y *getHeight*

```
int ancho=disco.getWidth(this);
int alto=disco.getHeight(this);
```

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class FuncionesApplet extends Applet {
    Image disco;
    public void init() {
        this.setSize(400,300);
        disco=getImage(getDocumentBase(), "disco.gif");
    }

    public void paint(Graphics g){
        g.drawLine(0, 0, getSize().width-1, getSize().height-1);

        g.setColor(Color.red);
        g.fillRect(50, 150, 50, 60);

        g.setColor(Color.cyan);
        g.fillArc(10, 10, 150, 100, 0, 270);
        g.setColor(Color.black);
        g.drawArc(10, 10, 150, 100, 0, 270);

        Polygon poligono=new Polygon();
        poligono.addPoint(100, 120);
        poligono.addPoint(150, 280);
        poligono.addPoint(170, 200);
        poligono.addPoint(190, 250);
        poligono.addPoint(200, 60);
        g.setColor(Color.yellow);
        g.fillPolygon(poligono);
        int[] x={100, 150, 170, 190, 200};
        int[] y={120, 280, 200, 250, 60};
        g.setColor(Color.blue);
        g.drawPolygon(x, y, x.length);

        g.drawImage(disco, 250, 50, this);
    }
}
```