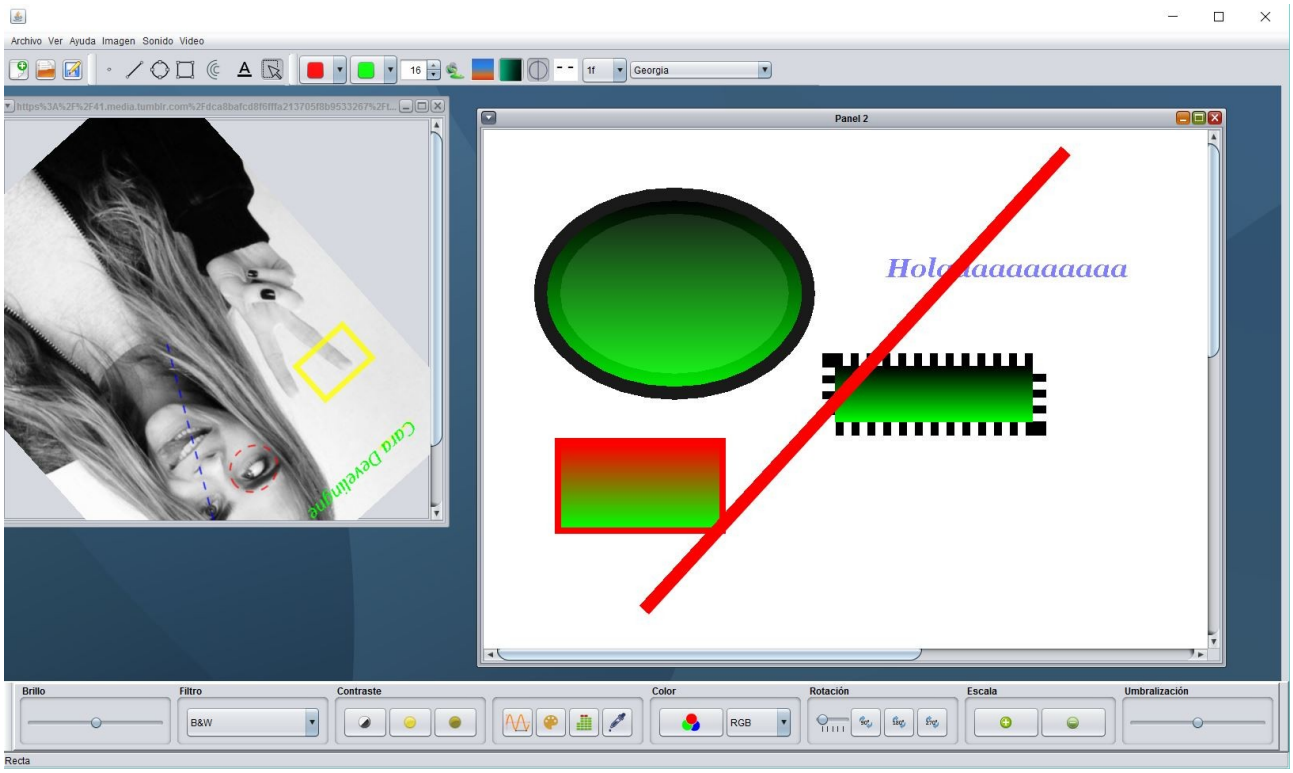
Documentación del Proyecto Final de Sistemas Multimedia



Ana Alicia Vílchez Ceballos

Introducción

Esta documentación se ha realizado para explicar la biblioteca creada por mi para el diseño de la aplicación de la asignatura Sistemas Multimedia así como la aplicación en sí. Se hablará de las clases de cada uno de sus paquetes, incluyendo su funcionalidad dentro de la aplicación y cómo es aplicada dicha funcionalidad.

Esta aplicación se puede utilizar para editar de forma variada contenido multimedia, centrándose más concretamente en imágenes: se podrán aplicar diversos filtros, variar el brillo gracias a un slider, escalado, variación del contraste, aplicar la función seno…
Además de editar imágenes podremos dibujar formas sobre las mismas, ya sean lineas, cuadrados, elipses, texto… Pudiendo editar dichas formas antes de realizar el volcado de la figura realizada sobre la imagen: cambiando el color, la fuente en caso de ser texto, añadiendo degradados (ya sea horizontal o vertical), dándole transparencia, cambiando el grosor del "borde" de la figura, etc.

La aplicación descrita también podrá reproducir formatos de audio, en concreto el formato WAV y grabar sonidos en formato WAV.
También será capaz de reproducir contenido de video (aunque en escasos formatos), lanzar una ventana que activará y mostrará lo que ve la cámara "WebCam" de nuestro ordenador y podrá obtener capturas en forma de imagen de dicha ventana, o de la ventana de reproducción de video.

Qué puede hacer la aplicación

En este apartado se van a explicar todas las funcionalidades que puede llevar a cabo la aplicación:

Opciones menú superior

En esta imagen se pueden ver las opciones que tiene el menú superior de la aplicación.
En el menú "Archivo" podremos abrir un nuevo lienzo "en blanco", podremos abrir una imagen o podremos guardar la imagen editada.
En el menú "Ver" podremos mostrar/ocultar la barra de herramientas, la barra de atributos y /o la barra de edición de imagen.
En el menú Ayuda tendremos una opción que nos lleva al nombre de la aplicación, mi nombre, y la versión.
En el menú "Imagen´" tendremos la opción de duplicar la imagen que consiste en realizar una copia exacta sin referenciar a la imagen que se está duplicando.
En el menú "Sonido" tendremos las opciones de abrir y grabar un sonido.
Por último en el menú video podremos abrir un video, mostrar lo que ve la WebCam o capturar una imagen de dicha ventana WebCam o de la ventana video.

Atributos y otros ajustes

Con los tres primeros botones de está barra de herramientas podremos abrir un nuevo lienzo, abrir una imagen o guardar la imagen editada. A continuación se muestran las formas de dibujo: podremos dibujar un punto, una recta, una elipse, un rectángulo, la opción de la curva que no se ha implementado (no pinta bien la curva), texto (que no es posible moverlo por el lienzo/imagen) y la opción de editar, que estando seleccionada es capaz de mover la figura a cualquier parte del lienzo.

A continuación tenemos dos ComboBox para elegir el color del “borde” de la figura y el relleno, además de implementar con estos dos el degradado ya sea horizontal o vertical.
Con el siguiente componente podremos darle el grosor elegido a nuestra figura (mínimo grosor 1).
A partir de las opciones siguientes (consisten en un grupo de botones) podremos darle relleno liso gracias al segundo comboBox de colores, degradado horizontal o degradado vertical.
El siguiente botón se utiliza para mejorar el aspecto de la figura (eliminando irregularidades).
A continuación con el siguiente botón podremos dibujar la figura con el borde discontinuo.
Con el posterior componente podremos fijar la transparencia de nuestra figura (del 0.1f al 1f). Por último con el siguiente componente podremos fijar el tipo de fuente para el texto.

Barra de Herramientas edición de imágenes

Con esta barra de herramientas podremos variar el brillo de nuestra imagen como primera opción, cambiar los filtros como segunda, marcar el contraste, aplicarle la función seno, sepia, aplicarle la ecualización o variar el color según el color que tengamos en la barra de atributos. Después podremos descomponer la imagen en sus distintas bandas, o abrir la imagen de nuevo como los tres formatos de color más conocidos. Las siguientes opciones son para rotar la imagen. A continuación podremos rotar la imagen según veamos conveniente, escalarla o aplicarle la umbralización.

Explicación de la biblioteca utilizada: SM.AVC.Biblitoteca

El paquete “Gráficos”

En este paquete se encuentran las clases necesarias para implementar las formas de dibujo. La jerarquía consiste en lo siguiente:
Tenemos una clase abstracta llamada “MyShape” que se encarga de definir los métodos abstractos necesarios para implementar las figuras que son los que se muestran en la siguiente imagen

```
//Métodos abstractos a implementar en las subclases que heredan de MyShape:

public abstract void formarFigura();

public abstract void setLocation(Point2D point);

public abstract void updateShape();

/**
```

Métodos abstractos de MyShape

Además cumplirá con la función de aplicar los atributos que se definen en la interfaz a las figuras creadas o por crear.

```
//Atributos de las formas inicializados porque no existe el constructor:
private int grosor = 1;
private Color colorAsignado = Color.BLACK;
private boolean editar = false;
private boolean alisado = false;
private float nivelTransparencia = 1;
String tipoG; //para establecer el tipo de graciente: horizontal, vertical o relleno

RenderingHints render = new RenderingHints(RenderingHints.KEY_ANTIALIASING,
        RenderingHints.VALUE_ANTIALIAS_ON); //para el alisado, mejorando la renderizacion de la figura

private final float dash1[] = {10.0f};
private BasicStroke dashed;
private Color colorRelleno;
private boolean discontinuidad = false;

//atributos para el texto
String tf; // para establecer el tipo de fuente del texto
```

Atributos de las figuras

Los atributos serán asignados gracias al método paintShape (excepto los degradados que son atributos específicos del rectángulo y la elipse), que se encargará de aplicar los atributos al objeto de tipo Graphics2D recibido como parámetro (método que deberá ser sobrecargado en las distintas subclases para poder dibujar la figura).

```
public void paintShape(Graphics2D g){

    g.setPaint(colorAsignado);
    g.setStroke(new BasicStroke(grosor));
    if(nivelTransparencia < 1)
        g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, nivelTransparencia));
    if(alisado) //si activamos alisado, se aplicará la renderizacion
        g.setRenderingHints(render);
    if(discontinuidad)
        g.setStroke(new BasicStroke(grosor,
                    BasicStroke.CAP_BUTT,
                    BasicStroke.JOIN_MITER,
                    10.0f, dash1, 0.0f));

    Font fuente = new Font(tf, Font.BOLD | Font.ITALIC, grosor);
    g.setFont(fuente);


}
```

Método paintShape de la clase MyShape

Esta clase la he definido como abstracta porque no se va a implementar ningún objeto general de MyShape. Los objetos implementados para las figuras serán del tipo de las distintas subclases que heredan de MyShape.

El resto de clases que se van a comentar a continuación (las utilizadas en la aplicación) se corresponden con las subclases que heredan de MyShape.

La clase “ARectangulo” contiene como atributo principal un objeto de tipo Rectangle.Double y dos puntos del tipo Point2D necesarios para dibujar el rectángulo a partir del objeto Graphics2D que le pasaremos al método paintShape de Arectangulo.

```
@Override
 public void paintShape(Graphics2D g){
    //va a llamar a la clase de la que hereda para establecer los atributos
    super.paintShape(g);
    //dibujamos la forma
    g.draw(rectangulo);
    //estos atributos los aplicamos aqui para que cojan el color después de hacer el draw
    //para rellenar la figura con el nuevo color especifico
    if(getTipoG() == "relleno"){
        g.setPaint(getColorRelleno());
        g.fill(rectangulo);
    }

    else if( getTipoG() == "gradienteV"){
        if(p1 != null && p2 != null){
            GradientPaint degradado = new GradientPaint((float)p1.getX(), (float)p1.getY(),
                    getColorAsignado(), (float)p2.getX(), (float)p1.getX() , getColorRelleno());
            g.setPaint(degradado);
            g.fill(rectangulo);
       }
    }
    else if(getTipoG() == "gradienteH" ){
       if(p1 != null && p2 != null){
            GradientPaint degradado = new GradientPaint((float)p1.getX(), (float)p1.getY(), getColorAsignado(),
                    (float)p1.getX() , (float)p2.getY(), getColorRelleno());
            g.setPaint(degradado);
            g.fill(rectangulo);
       }
    }
 }
```

Método paintShape de ARectangulo

Antes de pintar el objeto creado en los métodos que comentaremos más adelante, decir que primero tenemos que llamar al método paintShape de MyShape para que aplique los atributos al objeto Graphics2D y que luego éste pinte la figura con los atributos deseados aplicados.
En este método podemos ver que según el tipo de relleno aplicado, se implementará un relleno fijo con el color de relleno, o se creará un degradado horizontal o vertical a partir del color de relleno y el color asignado para el borde, y se dibujará con el método “fill” de Graphics2D.

Ahora se van a comentar los métodos que tienen que ver con la creación del rectángulo y con la edición (es decir, con mover el mismo).
Empezaremos hablando del método “formarFigura” que empieza formando la figura, aunque no la acaba pues el segundo punto todavía no quedaría establecidd (hablaremos más adelante de los eventos de la clase lienzoProyecto). Lo que establecemos es el comienzo de la diagonal del rectángulo.
Otro método abstracto de MyShape a implementar es el “setLocation” que en este caso se encarga de situar el rectángulo en el punto pasado como argumento (manteniendo el ancho y alto). Será necesario para la edición de la figura

Por último tenemos el método “updateShape”. Este método termina de formar el rectángulo estableciendo la primera y la última coordenada de la diagonal del mismo.

La clase AOvalo es muy similar a la ya comentada ARectangulo, por lo que se pasará a comentar la ALinea2. Esta clase contiene un objeto Line2D.Double que se formará con los siguientes métodos:
En el método crearFigura de dicha clase empezamos dándole el primer punto donde se empezará a formar la linea gracias al constructor de la clase Line2D.Double, por lo que será inicializado el objeto comentado en este método.
Después tenemos el método updateShape que emplea el método setLine de Line2D para actualizar la creación de la figura a partir del primer y segundo punto.
Por último tenemos el método SetLocation que al igual que el setLocation de ARectangulo, que se encarga de situar la linea gracias a las coordenadas creadas a partir del punto pasado como argumento con setLine . Comentar que Line2D no tiene un setLocation como lo puede tener la clase Rectangle.
El método paintShape simplemente pinta el objeto de tipo Line2D ya creado y actualizado, heredando antes los atriutos implementados en MyShape:

```
@Override
 public void paintShape(Graphics2D g){
      super.paintShape(g);
      g.draw(linea);
 }
```

Por último he implementado la clase ATexto, que en este caso el método updateShape no era necesario, y el setLocation no ha sido implementado porque no se mueve el texto. El formarFigura podría haber inicializado el String al igual que el punto, aunque como no crea nada, porque eso lo tiene que hacer el método paintShape, esa inicialización la lleva a cabo el método setTexto.
En el método paintShape es donde se pinta el texto a partir del método drawString de Graphics2D:

```
@Override
   public void paintShape(Graphics2D g){
      super.paintShape(g);
         //dibujamos la forma
      if(texto != null){
        g.drawString(texto, (int)p.getX(), (int)(p.getY()));
      }
    }
```

El paquete “IU”

De este paquete se va a comentar la clase lienzoProyecto, que es la encargada de formar el lienzo de la aplicación y de llamar a los eventos para el dibujo, además de volcar las figuras en la “imagen blanca” del lienzo.
Debido a que los objetos (rectangulo, recta…) de las clases propias han tenido que ser inicializados al crearlos para evitar incidencias de Java, en el método createShape de esta clase empiezo poniendo todos los objetos a null, puesto que no serán inicializados hasta que no se produzcan ciertos eventos como el evento pressed unido a que se encuentra seleccionado el botón elipse, por ejemplo. Ya tenemos una pista de que este método va a ser llamado en el evento pressed.

En caso de que se haya seleccionado alguna de las herramientas de dibujo, se procederá a inicializar la figura deseada a partir de un punto que se corresponderá con el punto del evento pressed y se inicializarán también todos los atributos seleccionados en la interfaz de nuestra aplicación.

En el método que gestiona el evento pressed se distinguen dos opciones: que se encuentre seleccionado el botón “editar” para poder mover la figura, o que lo esté una herramienta (esto gracias al createShape) de dibujo. Si alguna de las figuras creadas se encuentra en estado “edición”, se procede a calcular las coordenadas del punto de modo que evitamos que la figura se mueva en torno al origen de coordenadas (0,0).

En caso de que se proceda a crear una figura (porque no está activada la edición), se inicializa el punto inicial y llamamos a createShape.

Con el método que gestiona al evento dragged ocurre algo similar pero que se identifica con terminar de formar la figura o con terminar de mover la figura.

Por supuesto que estos dos métodos deberán de llamar a repaint que llama al método paint, encargado de dibujar la imagen en el lienzo blanco y de dibujar las figuras sobre el lienzo (llamando a los métodos paintShape de las diferentes figuras).

Para realizar el volcado de las figuras sobre la imagen, he acudido al evento entered, es decir, salimos del lienzo, y cuando volvemos a entrar si hay una figura que ha sido creada y ésta no se encuetra en modo edición, la volcamos gracias al método volcarImagen. Dicho método lo que hace es llamar al método paint y le pasa como parámetro la imagen con image.createGraphics() para pintar lo fijado.