DOCUMENTACION

ALUMNO: JORGE VALENZUELA GARCIA

1. INTRODUCCION:

La siguiente documentación trata el desarrollo de la aplicación para la asignatura Sistemas Multimedia.

La aplicación realizada abarca funcionalidades multimedia como:

1. Pintar y editar formas geométricas juntos con sus atributos (color, trazo...) dentro de un lienzo activo.
2. Poder tratar y procesar imágenes de forma que se puedan editar sus características como el brillo o contraste, o bien aplicarle filtros entre otras funcionalidades. Además de lo mencionado tambien se podrá pintar sobre dichas imágenes.
3. Reproducir y grabar audio.
4. Reproducir video o capturas instantáneas del mismo.

La aplicación abarca estas funcionalidades dentro de un entorno multiventana.
Más adelante analizaremos mas detenidamente estos campos.

2. REQUISITOS FUNCIONALES:

Vamos a dar una lista de requisitos funcionales con respecto a las característcas que debe cubrir la aplicación:

1. REQUISITOS GENERALES:

RF - 1.1. Aplicación multiventana.

RF - 1.2. Crear ventana correspondiente a audio, video, imagen o dibujo.

RF - 1.3. Reproducir archivos de audio.

RF - 1.4. Grabar archivo de audio.

RF - 1.5. Reproducir archivos de video.

RF - 1.6. Capturar desde video.

RF - 1.7. Pintar sobre ventana de imagen

RF - 1.8. Debe poseer una barra de herramientas por cada funcionalidad (audio/video, dibujo e imagen).

2. REQUISITOS DE VENTANAS:

RF - 2.1. Ventanas de imágenes:

RF - 2.1.1. Mostrar una imagen. Nueva, abierta o capturada.

RF - 2.1.2. Guardar imagen actual en fichero.

RF - 2.1.3. Mostrar barras de desplazamiento si es mayor que la zona visible.

RF - 2.1.4. Se podrá dibujar sobre las imágenes.

RF - 2.1.5. Realizar opciones de procesamiento sobre imagen.

RF - 2.1.6. Mostrar valor de la coordenada y valor RGB en la barra de estado de la aplicación.

RF - 2.2. Ventanas de reproducción de Sonido:

RF – 2.2.1. Reproducir archivo de audio.

RF – 2.2.2. Asignar al título de la ventana el nombre del archivo.

RF - 2.3. Ventanas de grabación de Audio:

RF – 2.3.1. Almacenar en fichero el sonido grabado por el micrófono.

RF - 2.4. Ventanas de Video:

RF – 2.4.1. Reproducir un archivo de video.

RF – 2.4.2. Asignar al título de la ventana el nombre del fichero.

3. REQUISITOS DE DIBUJO

RF - 3.1. Pintar un punto.

RF - 3.2. Pintar una linea.

RF - 3.3. Pintar un rectángulo.

RF - 3.4. Pintar una elipse.

RF - 3.5. Pintar una linea curva.

RF - 3.6. Pintar un polígono.

RF - 3.7. Asignar un color a cada forma geométrica.

RF - 3.8. Asignar un grosor a cada forma geométrica.

RF - 3.9. Asignar un relleno a aquellas formas que lo permitan.

RF - 3.10. Asignar un relleno degradado a cada forma geométrica que lo permita.

RF - .11. Asignar un interlineado al trazo de la figura.

RF - 3.12. Alisar los bordes de las figuras.

RF - 3.13. Ajustar la transparencia de las figuras.

RF - 3.14. Mover las formas dibujadas.

RF - 3.15. Asignar nuevos valores a una figura ya dibujada.

RF - 3.16. Que todas las figuras dibujadas se queden en el lienzo.

4. REQUISITOS DE LAS IMÁGENES:

RF - 4.1. Duplicar la imagen actual y mostrarla en una nueva ventana.

RF - 4.2. Modificar el brillo de la imagen actual.

RF - 4.3. Modificar el contraste de la imagen actual.

RF - 4.4. Oscurecer o iluminar la imagen actual.

RF - 4.5. Aplicarle filtros de emborronamiento, enfoque o relieve.

RF - 4.6. Obtener el negativo de la imagen actual.

RF - 4.7. Pasar a blanco y negro la imagen actual.

RF - 4.8. Aplicarle el filtro sepia.

RF - 4.9. Rotar la imagen libremente sobre si misma.

RF - 4.10. Escalar la imagen actual (aumentar o disminuir).

RF - 4.11. Mezclar dos imágenes.

RF - 4.12. Aplicar umbralización.

RF - 4.13. Nueva operación de diseño propio.

5. REQUISITOS DE AUDIO Y VIDEO:

RF - 5.1. Reproducir audio.

RF - 5.2. Grabar audio desde micrófono.

RF - 5.3. Reproducir video.

RF - 5.4. Capturar desde la cámara o desde el video actual.

6. REQUISITOS MENU PRINCIPAL:

RF - 6.1. Crear nueva imagen.

RF - 6.2. Abrir archivo video/imagen.

RF - 6.3. Guardar imagen actual.

RF - 6.4. Ver/Ocultar las diferentes barras de herramientas

RF - 6.5. Mostrar "Acerca de" que contiene la información del programa (autor, versión, etc.)

REQUISITOS NO FUNCIONALES:

RNF - 1. Habrá un escritorio central donde iremos añadiendo las diferentes ventanas creadas.

RNF - 2. En caso de dibujo o imagen, cada ventana tendrá una única imagen o lienzo respectivamente.

RNF - 3. El título de la ventana corresponderá a "Nuevo" si es una imagen creada por el usuario, "Titulo del archivo" si es una imagen o video abierto desde fichero, y "Captura" si la imagen es capturada.

RNF - 4. Solo se podrá dibujar sobre el área de la imagen, por tanto hay que delimitarlo con un Clip.

RNF - 5. Cada figura dibujada tendrá sus propios valores.

RNF - 6. Todas las figuras dibujadas deben quedarse en el lienzo.

RNF - 7. La reproducción de audio será controlada por botones en la misma ventana de Audio o en la barra de herramientas de Audio/Video.

RNF - 8. La grabación de audio se controlará mediante botones dentro de la ventana de grabación o desde la barra de herramientas de Audio/Video.

RNF - 9. A la hora de grabar audio se elegirá la ruta del archivo antes de comenzar a grabar.

RNF – 10. A la hora de reproducir un video tendremos botones dentro de la ventana de video o en la barra de herramientas de Audio/Video para controlar la reproducción del mismo.

RNF - 11. En la barra de dibujo habrá un icono por forma geométrica a dibujar además de los botones para cambiar o asignar los atributos de las figuras (color, trazo, alisado...).

RNF - 12. La figura que seleccionemos dentro del lienzo para editar, tendrá un BoundingBox a su alrededor para identificarla más fácilmente.

RNF - 13. Tendremos una barra de herramientas para las funcionalidades de la imagen, tales como el cambio de brillo o aplicación de filtros entre otros.

RNF - 14. El giro de la imagen se realizará mediante un deslizador, al igual que la aplicación del brillo, la umbralizacion y la mezcla de imágenes.

3. ANÁLISIS Y DISEÑO

Para analizar el proyecto de una forma mas sencilla iré analizando la aplicación por partes:

- **Interfaz o Ventana Principal.**
- **Dibujo.**
- **Imagen.**
- **Sonido.**
- **Video**

INTERFAZ O VENTANA PRINCIPAL:

Para comenzar la interfaz de la aplicación la vamos a basar en la **VentanaPrincipal**, la cual es una clase que hereda de **JFrame.** Sobre esta ventana vamos a ir incluyendo todos los controladores de la aplicación en sus correspondientes barras de herramientas las cuales heredan de **JToolBar**.

Esta clase será la encargada de mandar los mensajes a las ventanas internas que tenga activas, las cuales heredan de **JInternalFrame** y son, **VentanaInterna** para imágenes y dibujo, **VentanaInternaAudio** para reproducción de audio, **VentanaInternaGrabacion** para grabación de audio y **VentanaInternaJMFPlayer** para la reproducción de video o captura. *Ver Figura 1*.

Figura 1.

La clase central es VentanaPrincipal, y las de los alrededores son: VentanaInterna, VentanaInternaAudio, VentanaInternaGrabacion y VentanaInternaJMFPlayer.
Para ver mejor las figuras las dejaré en una carpeta en la raiz del proyecto.

Estas ventanas internas se irán añadiendo al escritorio de la aplicación, el cual hereda de **JDesktopPane**.

Dentro de estas ventanas podemos ver que tenemos:

- **VentanaInterna:** Dentro tenemos un **Lienzo2DImagen** el cual hereda de **Lienzo2D** que a su vez hereda de **JPanel**. Para saber mas sobre estas clases ver apartado de imagen y dibujo.
- **VentanaInternaAudio:** Esta ventana hereda de **JInternalFrame**. Dentro tenemos los dos botones para controlar los métodos necesarios para reproducir y parar el audio que tenga abierto. Para saber mas ir al apartado de Sonido.

- **VentanaInternaGrabacion:** Esta ventana hereda de **JInternalFrame**. Dentro tenemos los dos botones para controlar los métodos necesarios para grabar y parar el audio que esté capturando el micrófono. Para saber mas ir al apartado de Sonido.
- **VentanaInternaJMFPlayer:** Esta ventana hereda de JInternalFrame, y dentro contiene los métodos necesarios para la reproducción de video, y captura de instantáneas de cámara o video.

DIBUJO:

Para llevar a cabo la función de dibujo que se nos pide en los requisitos hemos creado la clase **Lienzo2D**, la cual hereda de **JPanel** .

Sobre la clase **Lienzo2D** hemos creado una serie de métodos y atributos que nos permiten satisfacer los requisitos pedidos para esta parte, además de que estamos usando la clase **Graphics2D** para dibujar. Pasamos a explicar esta clase:

Para dibujar estamos usando la clase **Graphics2D** dentro de **Lienzo2D**, la cual nos permite dibujar formas de la clase **Shape,** pero, con respecto a los requisitos que se nos piden, nos damos cuenta de que esta clase no es suficiente, ya que la clase **Lienzo2D** debe de contener los atributos como color **(Color),** trazo **(Stroke)** entre otras de cada objeto de tipo **Shape** dibujado, lo cual hace que cuando queramos dibujar cada forma con sus propios atributos se vuelva algo tedioso, más aún cuando queramos escoger una figura de las dibujadas y volver a asignarle nuevos atributos, lo cual se vuelve insostenible.

Por ello he tomado la decisión de crear una clase propia para tener así un mejor diseño y fucionalidad aunando estos atributos junto a la geometria dada por la clase **Shape**. Así surge la clase abstracta **Figura**.

Con la clase **Figura** lo que estamos consiguiendo es aunar la geometría de las formas **(Shape)** y los atributos de las mismas, además de crear un método propio para pintar. De esta forma la clase **Lienzo2D** queda mucho mas descargada y podemos hacer uso de una clase para pintar formas juntos a atributos de una forma mucho mas sencilla, haciendo asi que la clase **Lienzo2D** solo deba de tener una variable para cada atributo, y sea la encargada de crear cada objeto de la clase **Figura** según los parámetros enviados desde la **VentanaPrincipal,** una vez hayamos creado la figura se almacenará en un vector de objetos de la clase **Figura** y **Lienzo2D** solo tendrá que mandar pintar ese vector mediante el uso de su método **paint()**.

Como añadido esta clase propia **Figura** dentro de ella tiene otras clases las cuales heredan de ella, y que serán las formas que queramos pintar. A saber:

- **JVGPunto,** la cual sirve para almacenar la información de un punto.
- **JVGLinea,** la cual nos sirve para almacenar la informacion acerca de una linea.
- **JVGRectangulo,** la cual nos sirve para almacenar la informacion acerca de un rectangulo.
- **JVGElipse,** la cual nos sirve para almacenar la informacion acerca de una elipse.
- **JVGCurva,** la cual nos sirve para almacenar la informacion acerca de una linea curva.
- **JVGPoligono,** la cual nos sirve para almacenar la informacion acerca de un polígono.

Ver Figura 2:

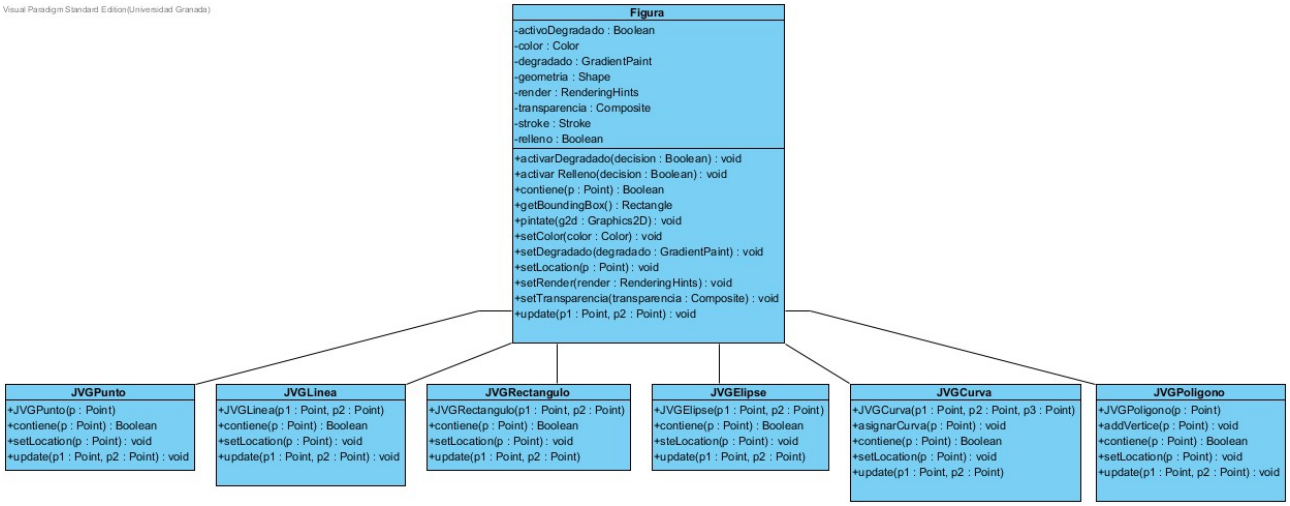

Figura 2.
*Clase abstracta **Figura,** y las clases **JVGPunto, JVGLinea, JVGRectangulo, JVGElipse, JVGCurva y JVGPoligono** que heredan de ella.*

Todas estas clases tienen además definidos sus propios métodos abstractos, como **setLocation()** o **contiene()** entre otros, para así, solo desde la llamada a un objeto de clase **Figura** podamos usar estos métodos, los cuales en caso de usar la clase **Shape** sabiendo que dentro de un objeto de clase Shape podemos tener un objeto de clase **Rectangle2D** o **Line2D,** tendríamos mas problemas ya que Shape no reflja estos problemas.

IMAGEN:

Para satisfacer los requisitos de las imágenes nos vemos en la situación de tener que crear una neva clase, **Lienzo2DImagen**.
Esta clase **Lienzo2DImagen** hereda de **Lienzo2D,** para asi extender dicha clase e incorporarle nuevas caracaterística como la de poseer una imagen y poder pintar sobre ella.

Esta clase **(Lienzo2DImagen)** es la que vamos a asignarle a la clase **VentanaInterna,** para que asi **VentanaInterna** sea una ventana donde podamos tanto dibujar, como visualizar imágenes, y pintar sobre estas imágenes.

Bien, pero como podemos observar para los requisitos pedidos en imagen, no basta con solo visualizar y pintar sobre una imagen, sino que además necesitamos aplicarles efectos y poder procesarlas.

De momento la solución para el procesamiento de imágenes es que la **VentanaPrincipal** , a través de sus controladores, aplique los procedimientos correspondientes a la imagen que tenga la **VentanaInterna** activa del **escritorio** de la **VentanaPrincipal,** como por ejemplo, aumentar brillo, iluminar o contrastar haciendo uso de las siguientes clases entre otras:

- **RescaleOp.**
- **LookUpOp**

- **ByteLookupTable**
- **AffineTransform**
- **Kernel**
- **ColorConvertOp**
- **etc.**

Evidentemente en la codificación encontraremos más clases para el procesamiento de imágenes.

Todas estas clases han sido aconsejadas por el guión de prácticas de la asignatura **SM,** y se encuentran dentro de **NetBeans,** es decir, no son clases propias.

Como se puede observar, se usan muchas clases para efectos que podrían estar agrupados por su resultado de vista al **usuario,** por ejemplo, dentro del programa podemos encontrar la aplicación de blanco y negro a la imagen o la aplicacion de su negativo, y veíamos que las clases usadas para este fin eran diferentes, por este motivo, para la aplicación de lo que se pueden ver como filtros he creado la clase **Filtro.**

La clase **Filtro** lo que consigue es juntar esos efectos aplicados a la imagen que retocan su aspecto, es decir, pasar a blanco y negro, desenfocar, marcar el relieve, aplicar negativo, etc. De esta forma solo con la creacion de un objeto **Filtro** (el cual tiene un **BufferedImage** como variable privada) podemos llamar a sus metodos como **blancoyNegro()** o **enfoque()** entre otros y nos devolverá una **BufferedImage** con el efecto ya aplicado. De esta forma agrupamos, y es más sencillo aplicar los efectos desde la **VentanaPrincipal.**

Pero no solo se ha creada la clase **Filtro,** sino que también se han creado las clases **Rotacion, SepiaOp, ExponerRGB** y **UmbralizacionOp.** Pasamos a explicar dichas clases:

La clase de **Rotacion** se ha creado por el simple hecho de simplificar el proceso de rotacion de una imagen. Al igual que la clase Filtro, ésta posee un BufferedImage como variable privada a la cual le aplicaremos la transformacion en tantos grados como se indique en el **método rotar(double grados**) y nos devolverá un BufferedImage rotado.

La clase **SepiaOp** se ha creado debido al seguimiento de guión de prácticas de la asignatura **SM.** Esta clase hereda de **sm.image.BufferedImageOpAdapter,** paquete aportado por la asignatura de **SM.** En dicha clase lo que hacemos es, a través de su método filter aplicar pixel a pixel un cambio en sus componentes RGB para hacer que la imagen resultado de la operación parezca que tiene un filtro Sepia.

La clase **ExponerRGB** es de creación propia y también hereda de **sm.image.BufferedImageOpAdapter**,. Sigue una estructura similar a la anterior, solo que en este caso en vez de aplicar mediante el metodo **filter()** el efecto sepia sobre la imagen, lo que hacemos es, pixel a pixel, quedarnos con el valor del componente R, G o B mayor, es decir, si en un determinado pixel el componente R es mayor que el de G y B, a éstos últimos vamos a aplicarles un valor nulo, solo dejando el valor de R. Finalmente obtendremos una imagen que nos deja ver en que lugares predomina mas un componente que los otros.

Por último queda explicar la clase **UmbralizacionOp** que tambien hereda de **sm.image.BufferedImageOpAdapter.** Esta clase se ha creado para analizar que pixeles estan por encima de un umbral. Aquellos pixeles de la imagen que estén por encima de iluminarán mientras que los que estén por debajo se oscurecerán.

Para ver mejor la representación de las clases ver Figura 3:

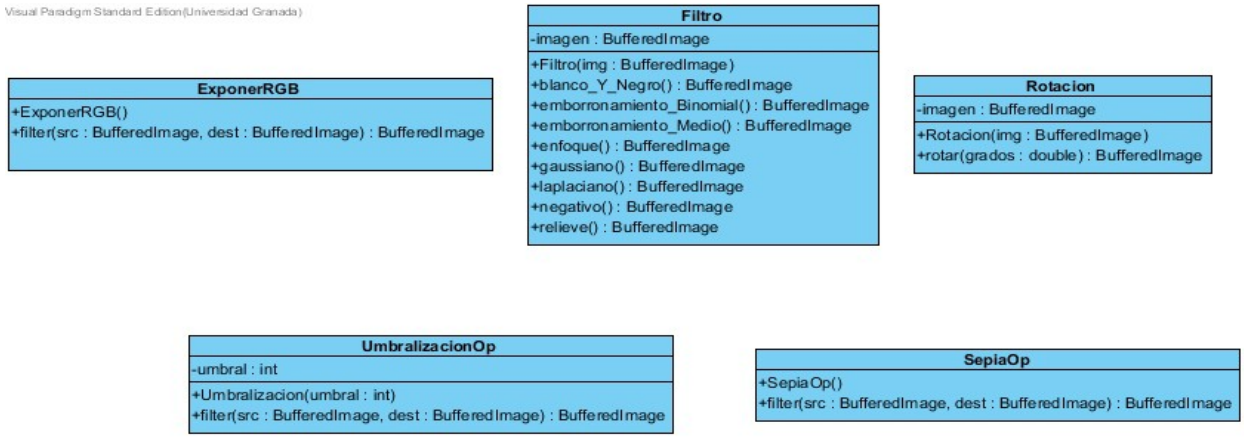

Figura 3.
Clases ***ExponerRGB, Filtro, Rotacion, SepiaOp y UmbralizacionOp.***

Con esto quedaría resuelto el apartado de imagen del análisis de la aplicación.

SONIDO:

Para abarcar el problema de que tenemos que hacer que el programa reproduzca audio, lo que hemos hecho ha sido crear la clase **VentanaInternaAudio** y la clase **VentanaInternaGrabacion,** las cuales heredan de **JInternalFrame.**

Por un lado, para la **reproducción,** hemos hecho uso del paquete aportado por la asigntura de **SM sm.sound,** donde encontramos los métodos **play()** y **stop()** para asi controlar un reproductor de clase **SMPlayer.**
Finalmente para el control de la reproducción se han añadido dos botones, uno para reproducir y otro para parar el audio que se encuentre en reproducción.

Por otro lado para la grabación también hemos usadi el paquete sm.sound, que tiene los metodos record() y stop() aplicables a una variable de tipo SMRecorder.
Al igual que para la reproducción hemos hecho uso de dos controladores, uno para comenzar la grabación y otro para detenerla.

VIDEO:

Para el apartado de video hemos hecho uso del paquete **JMF** de **Java** y he aplicado como sugiere el guión de prácticas y las transparencias.

Hemos creado una clase VentanaInternaJMFPlayer que hereda de JInternalFrame, la cual tendrá el reproductor de video cuando se cree.

Además de la reproducción de video esta clase es capaz de capturar una instántanea y exportarla como BufferedImage, para asi poder recogerla desde VentanaPrincipal y mandarsela a una nueva VentanaInterna donde poder tratarla o procesarla al modo que el usuario desee.

Con esto quedan expuesto el análisis y el diseño todos los apartados del programa Multimedia.

4. BIBLIOGRAFIA

- Material aportado en la asignatura de SM.
- http://jc-mouse.blogspot.com.es/2012/07/creacion-y-uso-de-javadoc-con-netbeans.html
- http://docs.oracle.com/javase/tutorial/2d/geometry/arbitrary.html
- http://docs.oracle.com/javase/tutorial/2d/geometry/strokeandfill.html
- http://docs.oracle.com/javase/tutorial/2d/geometry/primitives.html
- http://stackoverflow.com/questions/4051659/identifying-double-click-in-java
- http://tavmjong.free.fr/INKSCAPE/MANUAL/html_es/Filters-Pixel.html
- http://jagonzalez.org/usar-joptionpane-en-java/
- https://docs.gimp.org/es/plug-in-convmatrix.html