Ingeniería Informática:
Sistemas Multimedia.

Alberto Argente del Castillo Garrido.
76654048Q
aargente@correo.ugr.es

2 de agosto de 2018.

Índice

1. Introducción

En esta práctica se pretende realizar una aplicación multimedia que permita gestionar diferentes tipos de medios de forma integral. Para ello de desarrollara un entorno basado en escritorio en el que se distinguirán diferentes tipos de ventana en función del medio que se esté mostrando: **gráficos**, **imágenes**, **sonido** y **vídeo**. Sobre cada uno de dichos medios se podrán realizar diferentes tareas que, en función del medio, abarcarán desde la creación y/o captura hasta la edición, reproducción y procesamiento. Para ello, la aplicación, ver 1, contará con un conjunto de menús y barras de herramientas que permitan llevar a cabo dchas tareas.

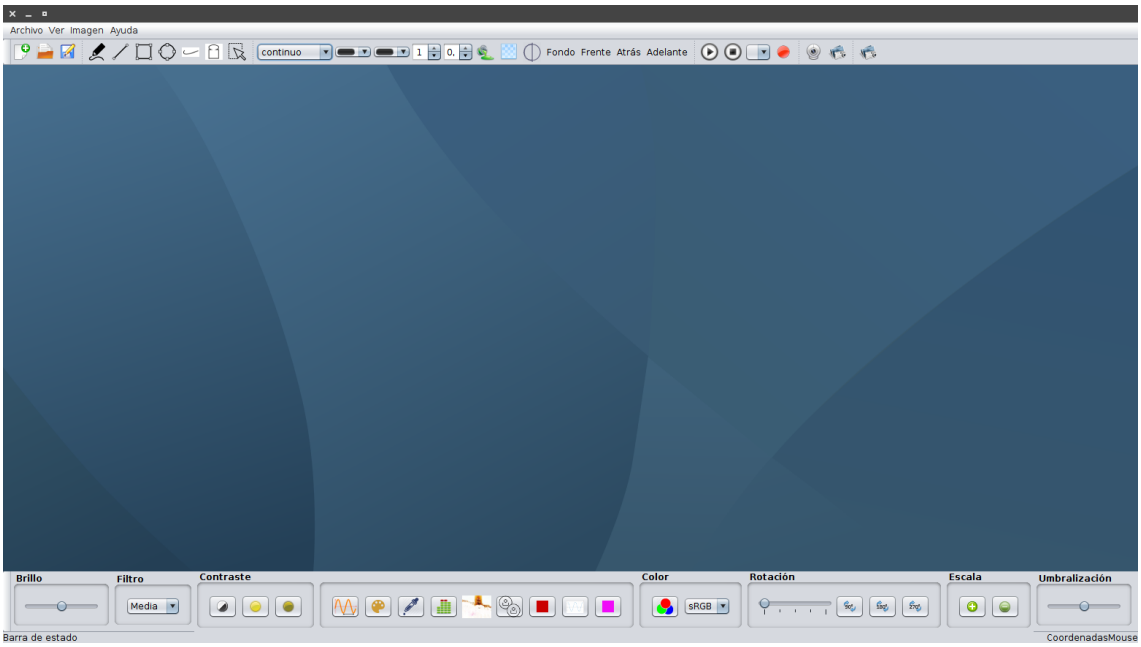

Figura 1: Aplicación creada.

2. Requisitos

2.1. Requisitos funcionales

En esta sección se enumerarán los requisitos funcionales que se han generado en la aplicación.

- **RF1, Nuevo**: Este botón permitirá al usuario crear una nueva ventana sobre la que podrá dibujar las distintas figuras que hay disponibles en la aplicación.

- **RF2, Abrir**: Este botón permitirá al usuario abrir cualquier tipo de archivo dentro de los soportados por la aplicación, disponiendo a su vez de distintos filtros para así ayudarle en la búsqueda del archivo que quiera abrir, ya sea imagen, sonido o vídeo.

- **RF3, Guardar**: Este botón permitirá al usuario guardar la imagen que se encuentre activa, ya sea una que haya abierto previamente, una que se haya generado tras una captura de pantalla o bien la de un lienzo sobre el que ha dibujado.

- **RF4, Dibujar figura**: Este requisito abarca los botones que permiten al usuario dibujar figuras, estas figuras son:

  - Trazo libre: El usuario podrá dibujar sobre el lienzo mientras mantenga el botón izquierdo del ratón pulsado.
  - Línea: El usuario podrá dibujar una linea sobre el lienzo.
  - Rectángulo: El usuario podrá dibujar un rectángulo sobre el lienzo.
  - Elipse: El usuario podrá dibujar una elipse obre el lienzo.

- Curva con un punto de control: El usuario podrá dibujar una curva con un punto de control, para ello primero se definirán los puntos de inicio y final de la curva y posteriormente el punto de control.
- Figura personalizada: El usuario podrá dibujar la figura personalizada diseñada por mí, se trata de una figura simple, una elipse encima de un rectángulo.

- **RF5, Mover figuras**: Este botón permitirá al usuario mover la figura seleccionada, la cuál será rodeada de un rectángulo discontinuo.

- **RF6, Editar atributos figura**: Se pueden modificar los atributos de una figura, ya sean los de la figura seleccionada o los de las última figura dibujada.

- **RF7, Cambiar trazo**: Con esta lista desplegable se podrá seleccionar el tipo de trazo para la figura, es decir, continuo o discontinuo.

- **RF8, Cambiar color trazo**: Este botón permitirá al usuario cambiar el color de trazo de la figura seleccionada, además de cambiar el color con el que se pintarán las siguientes figuras.

- **RF9, Cambiar color fondo**: Este botón permitirá al usuario cambiar el color con el que se pinte el relleno de la figura siempre y cuando esté activado, además de que será el color de fondo que tengan las siguientes figuras a dibujar.

- **RF10, Cambiar grosor**: Se permitirá al usuario elegir el grosor de las figuras tanto a dibujar como las seleccionadas.

- **RF11, Cambiar grado transparencia**: Se permitirá al usuario cambiar el grado de transparencia de la figura, estando en el intervalo [0,1].

- **RF12, Rellenar**: El usuario podrá activar y desactivar que se esté aplicando o no el relleno sobre la figura seleccionada o la figura a dibujar.

- **RF13, Transparencia**: Este botón permitirá al usuario que se aplica la transparencia o no sobre la figura seleccionada o las figuras a dibujar.

- **RF14, Alisar**: Con este botón el usuario podrá activar o desactivar el alisado sobre la figura seleccionada o las próximas figuras a dibujar, de forma que se mejore el dibujo de las mismas.

- **RF15, Enviar al fondo**: Con este botón se podrá enviar una figura al fondo del lienzo, pudiendo quedar tapada por otras figuras.

- **RF16, Enviar al frente**: Con este botón se podrá mandar al frente una figura del lienzo, quedando por encima del resto de las figuras y no pudiendo ser tapada por estas.

- **RF17, Enviar atrás**: Con este botón se podrá mandar un posición atrás a la figura seleccionada, de forma que sólo quede tapada por la anterior figura dibujada a ella y las siguientes figuras a dibujar.

- **RF18, Enviar adelante**: Con este botón se podrá mandar una posición adelante la figura seleccionada, de forma que pueda tapar a la figura dibujada tras ella.

- **RF19, Editar atributos de la figura seleccionada**: El usuario podrá modificar cualquier atributo de la figura seleccionada.

- **RF20, Mantener todas las figuras**: Se tienen que mantener todas las figuras dibujadas sobre el lienzo.

- **RF21, Play && Record Audio**: El sistema podrá grabar audios y reproducir audios en los formatos aceptados por Java.

- **RF22, WebCam**: El usuario podrá abrir, si lo desea, la WebCam que se encuentre disponible en el sistema.

- **RF23, Capturar WebCam**: El usuario podrá realizar capturas de pantalla sobre lo que se esté viendo en la WebCam.

- **RF24, Play && Stop Vídeo**: El usuario podrá abrir un archivo de vídeo y de audio y reproducirlo mediante el reproductor VLC.

- **RF25, Subir/Bajar volumen**: El usuario podrá subir y bajar el volumen del archivo de vídeo/audio que esté reproduciendo mediante la ventana de vídeo.

- **RF26, Seleccionar audio**: El usuario podrá seleccionar qué audio de los que se encuentran en la lista se debe poner para reproducir.

- **RF27, Capturar video**: Con este botón el usuario podrá realizar una captura de lo que se esté viendo en ese momento en el reproductor.

- **RF28, Ver barra de estados**: El usuario podrá activar/desactivar la barra de estados.

- **RF29, Ver barra de formas**: El usuario podrá activar/desactivar la barra de formas.

- **RF30, Ver barra atributos**: El usuario podrá activar/desactivar la barra de atributos.

- **RF31, Cambiar tamaño imagen**: El usuario podrá cambiar el tamaño del lienzo sobre el que se esté visualizando la imagen.

- **RF32, Operaciones imagen**: El usuario podrá aplicar diferentes operaciones sobre la imagen que se encuentre en el lienzo en ese momento. Estas operaciones son:

  - Brillo: Mediante un deslizador podrá variar el brillo de la imagen.
  - Podrá aplicar los siguientes filtros:
    - Media
    - Binomial
    - Enfoque
    - Relieve
    - Fronteras
  - Podrá aplicar contraste sobre la imagen
  - Iluminar más la imagen
  - Oscurecer más la imagen
  - Aplicar la función seno sobre la imagen
  - Aplicar el filtro sepia sobre la imagen
  - Tintar la imagen con el color de trazo seleccionado
  - Aplicar una ecualización sobre la imagen

- Aplicar el negativo sobre la imagen, es decir, invertir colores
- Duplicar, crea una imagen copia de la actual
- Operación RedOp de diseño propio, aplicada de pixel a pixel
- Operación LookUpTable diseño propio
- Operación CompOp de diseño propio, aplicada componente a componente
- Rotar la imagen, bien con deslizador o bien con los botones (90,180,270)
- Aplicar Zoom +/- sobre la imagen
- Aplicar un umbral, para dejar a partir de ese umbral la imagen en blanco y negro
- Obtener la bandas de la imagen
- Cambiar el espacio de color de la imagen, para ello se creará una nueva ventana con la imagen en otro espacio de color

- **RF33, Ver información de la aplicación**: Podrá ver quién es el autor de la aplicación, la versión en la que se encuentra y el nombre de la misma.

2.2. Requisitos no funcionales

En esta sección se pondrán los requisitos no funcionales del sistema, los cuales son:

- **RNF1**: Cada vez que se cree una ventana nueva de dibujo el usuario podrá decidir el tamaño de la misma.

- **RNF2**: Se mantendrá activo en la ventana principal los atributos de la ventana de dibujo que se tenga seleccionada.

- **RNF3**: Se mantendrán en el lienzo todas las figuras dibujadas.

- **RNF4**: Cada figura dibujada tendrá sus propios atributos.

- **RNF5**: Se podrán modificar los atributos de las figuras dibujadas.

- **RNF6**: Sólo se podrán guardar los archivos de tipo imagen.

- **RNF7**: Sólo se abrirán correctamente los archivos que permiten los filtros.

- **RNF8**: Se puede cambiar el tamaño de la ventana de dibujo en cualquier momento.

3. Análisis

Además de la ventana principal de la aplicación, ver 1, vamos a necesitar de más ventanas, para ser más concretos tres, una para **gráficos** e **imágenes**, otra para la **webcam** y otra para **vídeo**.

4. Diseño

La aplicación consta de una ventana principal en la que se podrán realizar todas las operaciones disponibles. Esta ventana principal cons-

ta de una barra de menú y dos barras de herramientas en las que se encuentran las distintas operaciones a realizar, una de ellas es exclusiva para operaciones con imágenes.

Para poder satisfacer los requisitos de esta práctica se han realizado 3 tipos distintos de ventanas internas, en las que se visualizará el contenido en función del tipo de ventana interna que sea:

- **VentanaInternaImage**, esta ventana interna se usará para poder visualizar todas las imagénes que sean abiertas desde la barra de menú como un lienzo totalmente vacío donde el usuario podrá pintar las distintas figuras disponibles.
- **VentanaInternaWebaCam**, esta vetana interna se usará para la WebCam.
- **VentanaInternaMedia**, esta ventana interna se usará para satisfacer los requisitos de video.

Para que la **VentanaInternaImage** pueda funcionar correctamente, se ha añadido un **Lienzo2DImagen** que hereda de Lienzo permitiendo así que se puedan visualizar imágenes y a la vez pintar en el lienzo con las distintas figuras que hay. En Lienzo2DImagen sólo se han añadido las funciones que nos permiten añadir una imagen, obtener la imagen que se está visualizando y además obtener las bandas de la imagen que se está visualizando, esta última funcionalidad se usará para algunas operaciones con imágenes.

En la clase **Lienzo** se ha incluido la funcionalidad para que el usuario pueda dibujar las distintas figuras disponibles mediante los eventos, **MousePressed**, **MouseDragged** y **MouseReleased**, con los se controlan los eventos de las figuras de un solo paso y dos pasos, en este caso sólo la Curva con un punto de control es dibujada de esta forma,

además de la funcionalidad para poder mover las figuras al frente o al fondo, según lo desee el usuario.

Para poder dibujar las figuras y que se haga acorde a como pedía la documentación, se ha tenido que realizar un diseño propio de clases, en el cual se ha creado la clase **Shape** como clase superclase del resto, que incluía a la clase contenedora **Atributtes** y las variables para poder dibujar el rectángulo que recubría la figura a la hora de ser editada. Además en esta clase se implementaron los métodos **draw**, **setLocation**, **contains** y **isNear**, los cuales luego serán implementados por las distintas subclases, ya que en esta clase estarán vacíos.

La clase contenedora **Atributtes** se ha creado para tener en ella todos los atributos de la figura, y a la vez distintos constructores por si es conveniente utilizar en función de alguna figura una u otra, como por ejemplo para el caso de dibujar una línea que no es necesario el atributo fill (relleno), pero en mi caso se ha añadido aunque posteriormente para facilitar la implementación se ha usado un inicializador de todos los atributos, ya que para el caso de la línea o la curva no se dibujarán aunque se le diga.

Partiendo de la super clase **Shape**, se han creado las distintas figuras que eran necesarias para cumplir los requisitos de la práctica, en los que en cada una de ellas se ha aprovechado las distintas clases de **Java2D**, es decir, para mi clase **Linea**, he creado una clase que hereda de mi super clase **Shape** y que tiene como atributo un objeto de la clase **Line2D**, de forma que así no hay que reescribir código.

Se ha optado por este diseño, ya que el problema principal que he encontrado era a la hora de tener atributos propios, o de mover las distintas figuras, ya que en caso de no tener este diseño de clases era necesario el uso de **Instance of**, y al ser un método común a todas ya

no es necesario utilizarlo, además de que podemos modificar en todo momento los atributos de las figuras.

Además de esto a la clase **Lienzo** se le ha añadido el **paint** un **clipArea** para definir el área en la que se puede dibujar, pudiendo ser cambiada en todo momento, y a la hora de crear una nueva **VentanaInternaImage**.

Para satisfacer las operaciones que se puedan realizar sobre la imagen, se han tenido que realizar las siguientes operaciones que no habían sido realizadas previamente en las prácticas:

- Negativo
- Operador LookUpOp definido por el estudiante
- Operación pixel a pixel definida por el estudiante
- Operación componente a componente definida por el estudiante

Para la realización de la operación **Negativo**, se ha creado un método de la clase **LookupTableExtension**, que hereda de LookupTable, y en simplemente se invierte el color que había en cada pixel de la imagen.

Para el operador propio **LookupOp**, probé distintos pero que no conseguí ajustar dado que los resultados en la imagen eran muy malos, por lo que obté por tomar la fórmula $sin^2(x)+cos^2(x) = 1$, y de la cual despejando $sin(x)$ queda de la siguiente forma: $sin(x) = \sqrt{1 - cos^2(x)}$, obteniendo un resultado distinto al que se conseguía con la función $sin(x)$ implementada en el guión de prácticas, pero aún así un resultado en el que se puede ver que se altera la imagen sin "destrozarla".

Para la operación pixel a pixel de diseño propio, he creado la clase **RedOp**, la cual hereda de la clase **BufferedImageOpAdapter** y no necesita de ningún parámetro a la hora de crearse. Al aplicar el filtro con esta operación lo que se hace es que a partir de un valor mínimo del **R**, y un valor máximo tanto para **G** como para **B**, se hace que el valor **R** de ese pixel sea 255, y los valores **G** y **B** sean 0, y en caso contrario se realiza la suma de estos valores, consiguiendo así destacar de entre la imagen todos los objetos rojos, dejando el resto de la imagen en blanco y negro.

Para la operación componente a componente de diseño propio, he creado la clase **CompOp**, la cual hereda de la clase **BufferedImageOpAdapter** y no necesita ningún parámetro para su inicialización, y cuyo método filter lo que hace es convertir la imagen a morada, para ello actuamos en función de la banda en la que estemos, ya que si estamos en la banda **0** o en la banda **2** le restamos 20 al valor y si estamos en la banda **1** le aumentamos el valor en 10, de forma que finalmente se consiga una tonalidad morada, aunque haya rasgos azulados, en la imagen.

Para satisfacer los requisitos del medio **sonido**, se han añadido en la **VentanaPrincipal** botones para realizar el play, stop y record, y una lista para así tener una lista de reproducción. Desde la barra de herramientas se pueden añadir archivos a la lista de reproducción aquellos ficheros de sonido que son reconocidos por Java, .wav y .au, de forma que se irán añadiendo a la lista de reproducción conforme los añadamos y serán elegidos para ser los siguientes a reproducir.

Para hacer la grabación se ha usado la clase **SMSoundRecorder**, para así poder añadir el manejador de eventos, **ManejadorAudio**, para que si iniciamos una grabación no se pueda iniciar otra. Además se ha realizado para que el usuario clicke en el botón de **record** y hasta

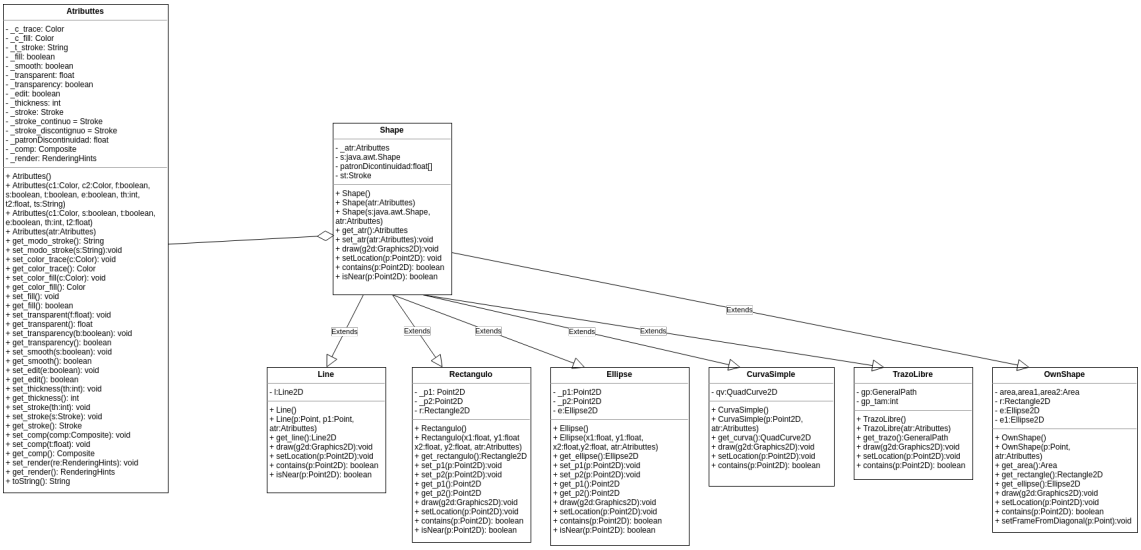

Figura 2: Diagrama de clases propio.

que no indique que quiere parar la grabación, con el botón **stop**, para así elegir nombre y fichero donde guardar la grabación que posteriormente será añadida a la lista de reproducción.

Para el play y el stop, he usado la clase **SMSoundPlayer** para, al igual que con el recorder, poder añadirle el manejador de eventos, **ManejadorAudio**, de forma que si el usuario está reprodución un audio ya, no pueda empezar a reproducir un segundo audio hasta que pulse stop.

Para la WebCam se creó la **VentanaInternaWebCam** en la que se ha añadido toda la funcionalidad, añadiendo un objeto de tipo **Webcam**, el cual proviene de la librería creada por **sarxos**, y también se ha usado el panel **WebCamPanel**, también creada por **sarxos**, para así poder situar la WebCam y poder visualizarla correctamente. Adicionalmente se pedía que se pudiera realizar una captura de pantalla de lo que se estuviera viendo en la ventana de la WebCam, por tanto

se creó el método **capture()**, que devuelve una imagen de tipo **BufferedImage** y que llamaremos desde **VentanaPrincipal**, creando así una nueva ventana de tipo **VentanaInternaImage**.

Por último para el medio **vídeo**, se ha creado la **VentanaInternaMedia**, en la que se ha utilizado las librerías proporcionadas por **VLC**, para así usar este reproductor que además nos permite reproducir una mayor variedad de formatos de sonido y de vídeo. Para iniciar esta ventana tenemos que seleccionar un archivo de audio/vídeo según sea indicado en el filtro **Video**, con la que se abrirá una ventana interna en la que el usuario podrá iniciar la reproducción, pausarla o pararla.

Adicionalmente he creado el manejador de eventos **VideoListener** que hereda de **MediaPlayerEventAdapter**, y he añadido la funcionalidad de que el usuario pueda variar el volumen del reproductor, también se ha añadido una barra de reproducción para que el usuario sepa cuánto lleva reproducido el vídeo del total de su duración pero no he logrado hacer que se pueda mover en función de cómo el usuario quiera, ya que al hacer click en esta barra debería colocar el tiempo donde se ha indicado, pero en vez de esto inicia la reproducción de 0.

También intenté crear una lista de reproducción añadiendo una barra de menú dentro de esta ventana interna, añadiendo así los ficheros, pero el reproductor utilizado no permite realizar este cambio, ya que para poder crear una lista hay que utilizar otro objeto de tipo **MediaListPlayer**, aún así intenté realizarla con el que se utiliza en la práctica de sonido pero no funcionada. Con el evento finished quise que si hubiera un solo archivo, no haría falta cambiar el reproductor, se iniciara de 0 pero tampoco funcionaba, ya que al mirar por Internet encontré que había eventos que no funcionaban correctamente y que habría que realizar más trabajo del supuesto para poder completar estas operaciones.

Por último el botón **Sig** se creó inicialmente para pasar a la siguiente canción pero al no poder realizar esto intenté hacer que si se clickaba sobre este botón se pusiera el reproductor en pantalla completa, pero no conseguí que funcionase, no sé si por fallo mío o por fallo de la librería.

5. Codificación

La API del código se mandó en la primera entrega de la práctica junto al código fuente del que fue generado esta mediante NetBeans. En esta API se encuentran comentados todos los métodos que se han implementado en la biblioteca propia, además la utilidad que representa cada clase que ha sido usada en el sistema y todas las variables que se han necesitado durante la realización de la aplicación.