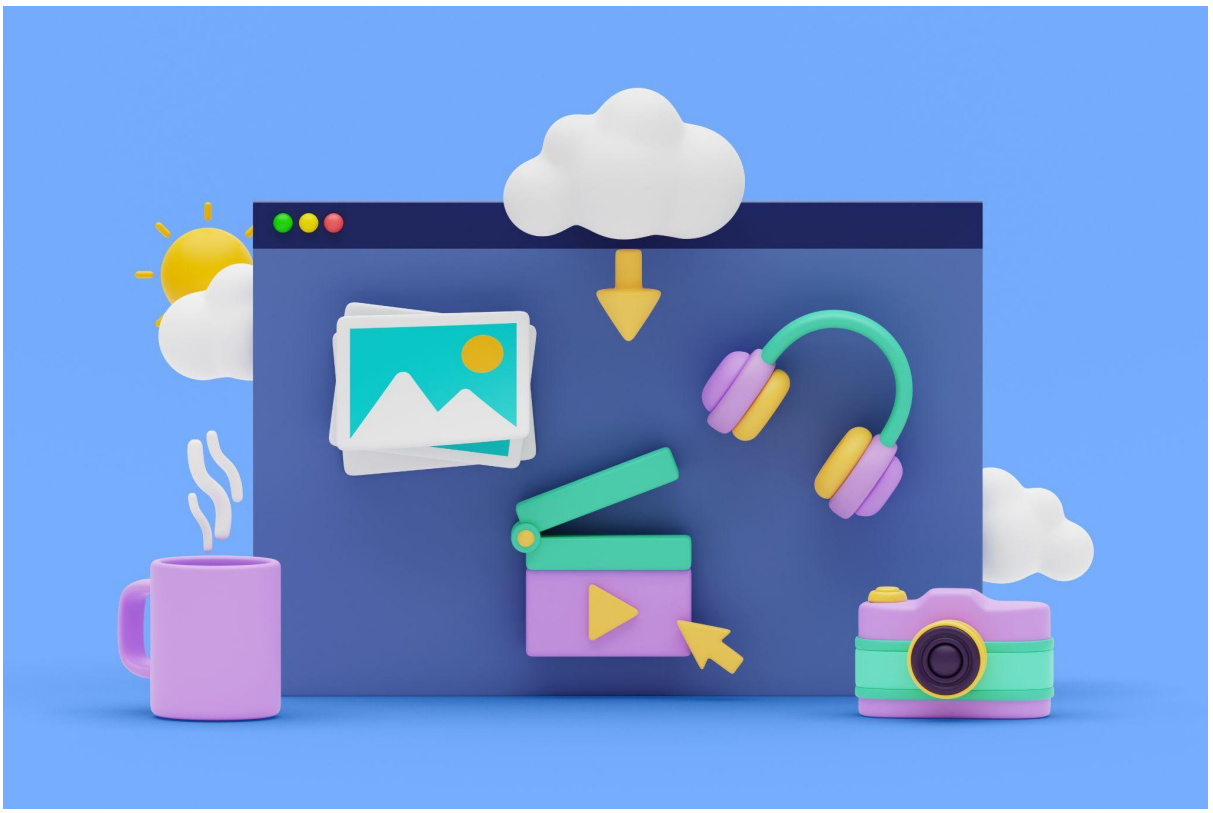
# Memoria práctica final de Sistemas Multimedia

## *Aplicación Multimedia*



Marina Jun Carranza Sánchez
Grado en Ingeniería Informática
Sistemas Multimedia - Curso 22/23

# ÍNDICE

# 0) Introducción

Esta memoria se divide en cinco bloques o módulos principales: gráficos, imágenes, sonido, vídeo e interfaz general. A su vez, los cuatro bloques principales se documentan siguiendo los pasos de la Ingeniería del Software: requisitos de la aplicación, análisis de los mismos, diseño del programa y codificación.

En este caso, se ha optado por la combinación de la fase de análisis y diseño en una sola parte, donde se describe tanto la comprensión de los requisitos como las decisiones de diseño tomadas, en lugar de separarlas, para asemejarse más al proceso de desarrollo seguido. Además, la codificación viene incluida en la documentación Javadoc generada a partir del proyecto.

# 1) Gráficos

## 1.1) Requisitos

La aplicación debe permitir dibujar diferentes formas geométricas (líneas, rectángulos, etc.) con distintos atributos (color, grosor, etc.) sobre un lienzo blanco o una imagen cualquiera.

**Aspectos del dibujo**

El usuario podrá dibujar sobre cualquier imagen la forma, con sus atributos seleccionados, para lo que se necesitan los siguientes requisitos:

- ➢ El lienzo mantendrá todas las figuras que se vayan dibujando.
- ➢ Cada figura tendrá sus propios atributos, independientes del resto de formas.
- ➢ Se debe poder visualizar una lista de las figuras que se van dibujando.
- ➢ El usuario podrá mover las figuras dibujadas “arrastrándolas” con el ratón.
- ➢ El usuario podrá volcar (dibujar) en la imagen las figuras seleccionadas en la lista de figuras, que pueden ser varias a la vez.
- ➢ El usuario podrá editar (los atributos de) las figuras, una vez creadas.

**Formas de dibujo y sus atributos**

La aplicación permite dibujar las siguientes formas geométricas:

- ➢ Trazo libre
- ➢ Línea recta
- ➢ Rectángulo
- ➢ Elipse
- ➢ Curva con un punto de control
- ➢ Forma que defina una cara sonriente

El usuario podrá elegir los atributos con los que se pintarán las formas.

- ➢ Color: podrá elegir el color con el que se pinta el trazo y el relleno, que pueden tratarse de colores distintos.
- ➢ Trazo: se podrán modificar el grosor y el tipo de discontinuidad del trazo (posibilidad de dibujar líneas continuas o líneas punteadas).
- ➢ Relleno: se podrá elegir entre rellenar con color liso, no rellenar o rellenar con degradado. En este último caso, se pueden seleccionar los colores que definen el degradado (el de relleno y el que coincide con el del trazo) y se ofrecen dos direcciones (horizontal y vertical).
- ➢ Alisado de bordes: se podrá (des)activar la mejora en el proceso de renderizado relacionado con el alisado de bordes.
- ➢ Transparencia: se podrá establecer un grado (cuyo valor será modificable) de transparencia asociado a la forma.

Cuando se cambia de una ventana a otra, deberán actualizarse las opciones seleccionadas en función de la ventana activa actual; es decir, tener memoria de la forma y atributos seleccionados en cada ventana.

## 1.2) Análisis y diseño

Se cuenta con una barra de herramientas que brinda acceso a todos los elementos necesarios para poder dibujar, incluyendo formas y atributos (cada elemento tiene asociado un “ToolTipText”).

**Aspectos del dibujo**

En una librería propia, se ha definido una clase propia para los lienzos llamada `Lienzo2D`.

- ➢ Para que el lienzo mantenga todas las figuras que se van dibujando, se tiene un vector de formas (lista de tipo Forma2D personalizada que hereda de Shape) donde se van guardando todas las formas que se pintan.
- ➢ Cada figura tendrá sus propios atributos, lo que es posible gracias a la definición de una jerarquía de clases asociadas a las formas y sus atributos. Cuando se dibuja una forma, esta se pinta usando los atributos activos en ese momento.
- ➢ El usuario podrá mover las figuras dibujadas “arrastrándolas” con el ratón tras activar el botón de selección representado con un icono de cursor.
- ➢ Se incluirá un panel dividido que en la parte derecha, muestre la lista de formas que hay pintadas en la ventana activa. Así, cada vez que se pinta una nueva, esta se añade a dicha lista. En la parte izquierda del panel dividido se encuentra el panel escritorio con todas las ventanas internas.
- ➢ Función de volcado en la imagen: las figuras que se seleccionen de la lista de formas, pueden dibujarse sobre la imagen tras pulsar el botón de volcado. Esto implica que esas figuras se eliminen del vector de figuras del lienzo y de la lista.
- ➢ Función de editar los atributos de figuras ya existentes: pulsando el botón “**`Ed.`**”, se podrá hacer click sobre cualquier figura ya existente (la seleccionada será rodeada por un rectángulo gris discontinuo) y cambiar los atributos que se desee, para actualizar la forma de inmediato. Cuando se seleccione cualquier figura para dibujar una nueva, automáticamente se saldrá del modo edición.

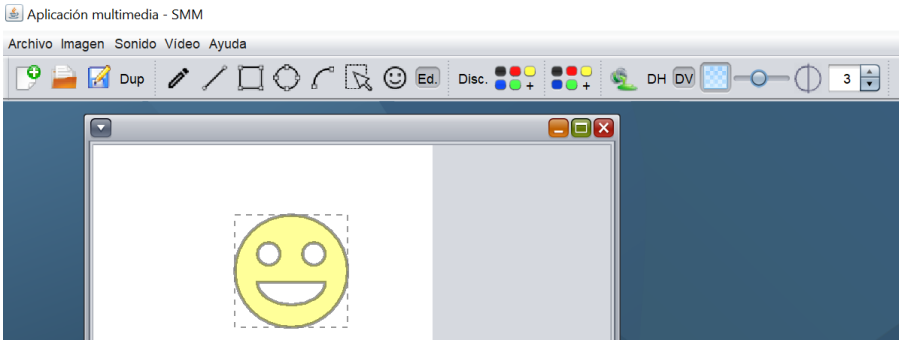



Figura 1: Ejemplo de edición interactiva de forma Smile2D

**Formas de dibujo y sus atributos**

En el paquete `sm.mcs.gráficos` de mi librería, se han definido clases propias para cada forma que puede dibujarse:

- Trazo libre → `TrazoLibre2D`
- Línea recta → `Línea2D`
- Rectángulo → `Rectángulo2D`
- Elipse → `Elipse2D`
- Curva con un punto de control → `Curva2D`
- Forma personalizada (área) que defina una cara sonriente → `Smile2D`

Cada figura dispone de un botón de dos posiciones, representado por un icono en la barra de herramientas, para poder dibujarla.

Además, cada instancia de figura será la responsable de almacenar sus propios atributos en variables privadas de clase: color de trazo, trazo (con grosor y tipo), la activación de alisado de bordes y el grado de transparencia. Y para las figuras distintas de la línea, los relacionados con el relleno (color, tipo, etc). Para ello, cada clase de forma tiene un método de dibujo que permite pintar la figura a partir de los atributos guardados, así como un método que devuelve el boundbox de cada figura (que será usado para la edición).

Mediante la barra de herramientas de dibujo, se pueden establecer los siguientes atributos:

- Color: se mostrará un panel de colores predeterminados, pudiendo lanzar un diálogo de selección de colores (“**+**”), donde elegir el color de trazo y el relleno, pudiendo ser diferentes, por lo que se dispone de dos grupos de selección de colores distintos.
- Trazo: se podrá modificar el grosor mediante y seleccionar el tipo de trazo, con el botón de discontinuidad (**`Disc.`**) para pintar líneas punteadas o continuas.
- Relleno: dependiendo del botón seleccionado, se podrá elegir entre rellenar con color liso, no rellenar (ninguna opción de relleno) o tener un degradado, en cuyo caso se pueden elegir los dos colores (uno el del relleno y el otro coincide con el del trazo), así como la dirección del degradado (botón **DH**: horizontal; **DV**: vertical).
- Alisado de bordes: se puede aplicar la mejora en el proceso de renderizado relacionada con el alisado de bordes, activando su botón correspondiente.
- Transparencia: se puede establecer un grado de transparencia a la forma, cuyo valor puede modificarse a través de un deslizador.

Conforme se cambia de una ventana a otra, se debe actualizar la siguiente información, dependiendo de la ventana activa:

- ➢ Los estados de los botones de forma y atributos de la barra de herramientas
- ➢ La lista de figuras que se han dibujado en dicha ventana, y que podrán volcarse.

Lo anterior se consigue definiendo una clase Manejadora propia. Los pasos son:

1) Definir la clase Manejadora
2) Crear el objeto manejador
3) Enlazar el generador con el manejador

Entonces, en la Ventana Principal se define una clase propia que se llamará `manejadorVentanaInterna` que hereda de `InternalFrameAdapter`. Dentro de la clase se definen y sobrecargan los dos métodos siguientes:

- ➢ `internalFrameActivated(...)`: se crea un objeto con la ventana interna actualmente seleccionada, y a partir de su lienzo, se obtiene el estado de los botones de formas, pudiendo así actualizarlos. Para la lista de figuras, se crea un nuevo modelo al que se le añadirán las formas contenidas en el vector del lienzo.
- ➢ `internalFrameClosing(...)`: se eliminan todos los elementos de la lista de formas al cerrar la ventana.

Para la gestión de eventos propios dentro de la clase Lienzo2D, se han creado clases:

- ➢ LienzoAdapter → el adaptador que implementa los métodos del listener
- ➢ LienzoEvent → guarda el objeto forma y atributos como el color
- ➢ LienzoListener → interfaz con los métodos gestiona cada evento

En la Ventana Principal se define una nueva clase manejadora propia para los eventos del lienzo: `MiManejadorLienzo`, que heredará de `LienzoAdapter` e implementará los métodos que actualizan la lista de figuras (`shapeAdded(...)`) y el valor RGB del píxel actual (`updateRGBPixel(...)`).

Cuando se cree una figura, se llamará al método `notifyShapeAddedEvent(...)`, pasándole como parámetro el nuevo evento de añadir una figura al lienzo. Este notificará el evento recién creado a la lista de listeners del lienzo.

## 1.3) Codificación

Toda la documentación javadoc de las clases contenidas en los paquetes:

- ➢ `sm.mcs.gráficos` → `Forma2D` y las clases para cada tipo de figura
- ➢ sm.mcs.iu → `Lienzo2D`
- ➢ `sm.mcs.eventos` → gestión de eventos del lienzo

Todos los métodos de Ventana Principal relacionados con el módulo de gráficos.

# 2) Imágenes

## 2.1) Requisitos (+ diseño)

El programa permite abrir cualquier imagen cuyo formato sea compatible (jpg, png, etc) y, como se ha visto en el anterior apartado, dibujarse sobre ella.

Como requisitos, se cumplen las siguientes operaciones, con sus botones asociados:

- Duplicar: creará una VentanaInternaImagen con una copia de la imagen → "**Dup**"
- Modificar el brillo y el contraste mediante deslizadores → Sliders Brillo y Contraste.
- Filtros de emborronamiento (incluyendo el horizontal de tamaños 5x1, 7x1 y 10x1) enfoque y relieve → Combo box de Filtros
- Contraste normal, iluminado y oscurecido → botones con iconos correspondientes
- Negativo (invertir colores) → "**N**"
- Operador cuadrático con deslizador para modificar el parámetro → botón con icono de función cuadrática y su correspondiente slider inferior
- Operador basado en la función spline lineal con un punto de inflexión
    - botón de dos posiciones para (des)activarlo
    - dos deslizadores con los que modificar los parámetros a y b de la función
    - Tras activar la opción, se crea una ventana nueva donde se dibujará la función spline a partir de los valores actuales de a y b, los cuales se mostrarán también.

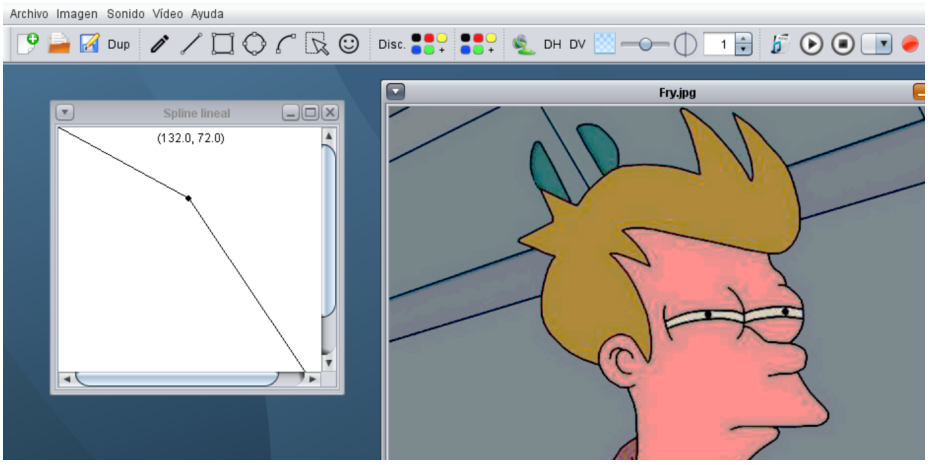



Figura 2.1: Visualización de función spline lineal, modificando los parámetros "a" y "b"

- Extracción de bandas → botón de extracción de bandas
- Conversión a los espacios RGB, YCC y GRAY → Combobox de espacios de colores
- Giro libre → slider junto a botón de giro de 180º
- Escalado (aumentar y disminuir) → botones con icono de aumentar y disminuir
- Tintado (con el color de trazo seleccionado en ese momento) → botón con icono de tintado y deslizador para variar el grado de mezcla (alpha)
- Sepia → botón de operador de sepia
- Ecualización → botón de ecualización

- ➢ Posterización (reducción del número de niveles por banda) → deslizador posterizar
- ➢ Resaltado del rojo (mantener el color original rojo y el resto a niveles de gris) → botón RojoOp, con deslizador que permite variar el umbral de selección de tono rojo
- ➢ Operador que permita variar de forma interactiva el tono de los colores (manteniendo su saturación y brillo) → slider para variar tonos
- ➢ Un operador "LookupOp" basado en una función propia con, al menos, un parámetro → explicado en el siguiente apartado.
- ➢ Una nueva operación de diseño propio aplicada pixel a pixel con, al menos, un parámetro → explicada en el siguiente apartado.

Todas las operaciones con slider se irán aplicando de forma concatenada, es decir, se aplicará sobre el resultado de operaciones aplicadas anteriormente.

## 2.2) Análisis y diseño

**Operador "LookupOp" basado en una función propia que tiene un parámetro:**

Para aplicar el operador Lookup, se puede pulsar el botón asociado a él, representado por una **M** de MiFunción, lo que aplicará la función con m=0 por defecto, o directamente usar el slider inferior para modificar el valor del parámetro "m" de la función.


Figura 2.2: Localización de botón de mi función Lookup

La función es `f(x):= m·arctan(x-120)+120`, donde "m" toma valores entre -100 y 100.

Su representación gráfica es:

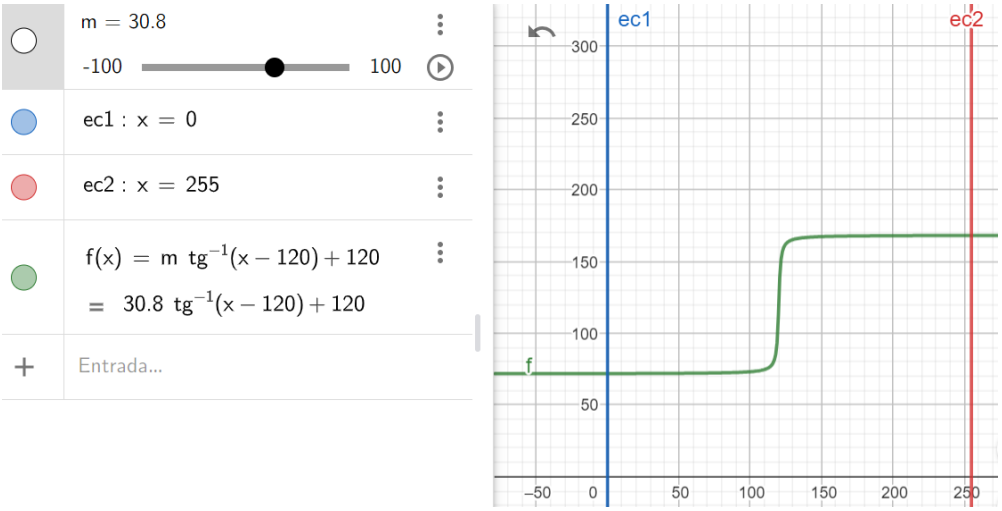



Figura 2.3: Representación de mi función con m=30.8

Variando el valor del parámetro “m”, se puede deducir que, cuando m pasa a ser mayor que 0 (o por el contrario, un número negativo), se produce un cambio drástico en el comportamiento, pues la función arcotangente se “invierte”.

Esto se puede comprobar al revisar el aspecto que toma la función cuando m < 0:

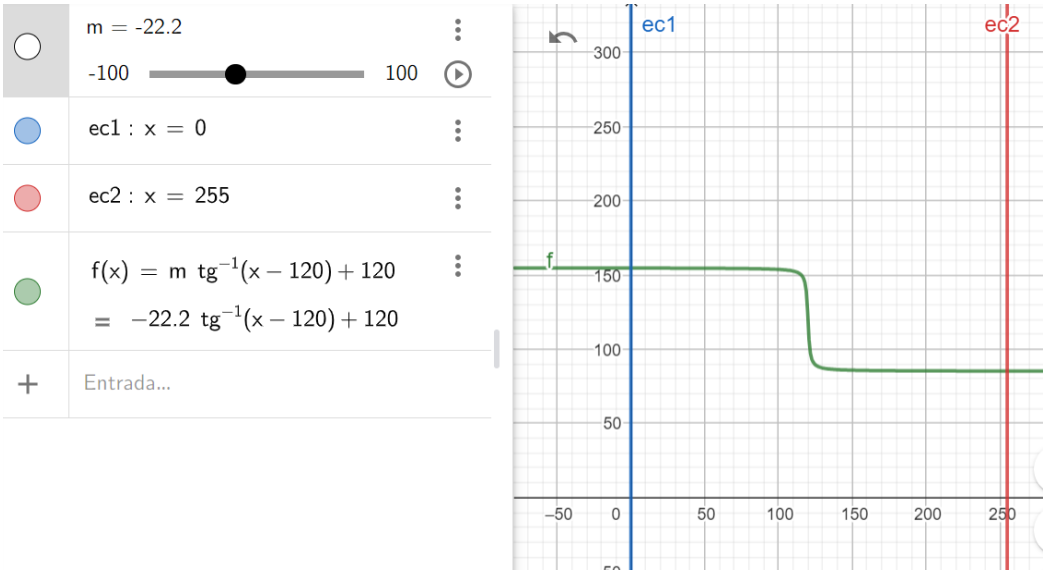



Figura 2.4: Representación de mi función cuando m=-22.2

Por tanto, m = 0 es un punto de inflexión, por así decirlo, que determinará si la imagen toma un aspecto distinto u otro.

**Operación de diseño propio, definida en una clase y aplicada píxel a píxel:**

`MiOperador` es una clase propia que define una operación aplicada píxel a píxel con dos parámetros, uno que determina si aplicar o deshacer un efecto de viñeteado, y otro que determina el factor de intensidad de dicho efecto.

Para aplicar el efecto del operador, primero se debe activar el botón toggle representado con una **V** de viñeteado. Después, se puede mover el slider inferior para modificar el valor del parámetro que determina la intensidad del efecto.


Figura 2.5: Localización de botón de mi operador

Si se quiere deshacer el efecto (bajo el riesgo de alterar los colores y calidad de la imagen original), basta con desactivar el botón y mover el slider de nuevo, esta vez revirtiendo el efecto de viñeteado.

El método filter de `MiOperador` funciona de la siguiente forma:

Comienza verificando si la imagen de origen es nula (si lo es, se lanza una excepción), después verifica si la imagen de destino es nula (en caso afirmativo, se crea una imagen de destino compatible).

A continuación, se calculan las coordenadas del centro de la imagen de origen y la distancia máxima desde el centro hasta un píxel en la imagen, mediante la fórmula de distancia euclidiana.

Se entra en un bucle anidado, iterando cada uno de los píxeles de la imagen de origen:

1) Para cada píxel, se obtiene el valor RGB y se descompone en sus valores de rojo, verde y azul, usando desplazamientos y máscaras bitwise.
2) Se procede a calcular la distancia desde el píxel actual hasta el centro de la imagen y luego se calcula un factor de atenuación basado en la distancia y el parámetro atributo de la clase.
3) Dependiendo del valor del atributo boolean de la clase, se aplicará (multiplicando los componentes RGB por el factor calculado anteriormente) o revertirá (dividiendo en lugar de multiplicando) el efecto del filtro. Además, los componentes se ajustan para no salirse del rango [0, 255].
4) Finalmente, se combinan los componentes RGB en un valor entero, y se establece este nuevo valor en la imagen de destino.

El bucle continúa iterando sobre todos los píxeles de la imagen, aplicando el filtro a cada uno y almacenando los resultados en la imagen destino, que será devuelta por el método.


Figura 2.6: Ejemplo de efecto de viñeteado en una imagen

Adicionalmente, la barra de estado localizada en el panel inferior de la aplicación informa de la posición del ratón y del valor RGB del píxel en dichas coordenadas.

## 2.3) Codificación

Documentación en javadoc del paquete `sm.mcs.imagen`, y todos los métodos de la Ventana Principal relacionados con el módulo de imágenes.

Las siguientes secciones, correspondientes al módulo de sonido y vídeo, se verán en menor profundidad, dado que ya fueron abordadas en las prácticas y no incorporan cambios o mejoras significativas.

# 3) Sonido

## 3.1) Requisitos

La aplicación permitirá:

- ➢ Abrir sonidos que se guardarán en una lista.
- ➢ La reproducción de dichos sonidos.
- ➢ La grabación y guardado de audios.

## 3.2) Análisis y diseño

Se incluirá una lista de reproducción, de manera que al abrir un nuevo fichero de audio, se añadirá a dicha lista desplegable asociada.

La interfaz de la aplicación tendrá:

- ➢ Un botón para reproducir sonido. Al pulsarlo, se reproducirá el audio que esté seleccionado en ese momento en la lista de reproducción.
- ➢ Un botón para detener o pausar la reproducción o grabación.
- ➢ Un botón para el grabado de audio, que iniciará el proceso de grabación, el cual finaliza al pulsar el botón de parada. El sonido se grabará en el fichero indicado por el usuario y se añadirá a la lista de reproducción.

La Ventana Principal tendrá dos atributos relacionados con la gestión de sonido: un `SMClipPlayer` y un `SMSoundRecorder`, ambos de la librería `sm.sound`.

Se definirá un manejador de sonido propio llamado `ManejadorAudio`, para actualizar los botones de reproducción y stop.

## 3.3) Codificación

Todos los métodos de la Ventana Principal relacionados con el módulo de sonido.

# 4) Vídeo

## 4.1) Requisitos

La aplicación permitirá:

- ➢ Reproducir vídeos: también podrá pausarlos y reanudarlos.
- ➢ Abrir una ventana con lo que capta la webcam.
- ➢ Tomar capturas de los vídeos que se reproducen.
- ➢ Tomar capturas de lo que capta la webcam.

## 4.2) Análisis y diseño

La aplicación permitirá tanto la reproducción de video como la captura de la cámara; para ello, se definirán ventanas internas específicas para cada tarea:

- ➢ Reproducción de vídeo → `VentanaInternaVídeo`: dispone de una zona de visualización central donde se muestra el vídeo. La ventana tiene como título el nombre del fichero que se está reproduciendo, pudiendo existir tantas ventanas como vídeos haya abiertos.
- ➢ Cámara (webcam) → `VentanaInternaCámara`: dispone de una zona de visualización central donde se muestra la secuencia que esté captando la cámara.

Además, la interfaz contará con los siguientes elementos:

- ➢ Un botón para reproducir video, que es el mismo usado para la reproducción de sonido. Al pulsarlo, se reproducirá el video de la ventana de tipo vídeo seleccionada en ese momento.
- ➢ Un botón para detener o pausar la reproducción, que es el mismo usado para la reproducción de sonido.
- ➢ Adicionalmente, la captura a través de webcam también requerirá un tipo específico de ventana interna que vaya Para abordar los requisitos anteriores, la aplicación deberá incluir:
- ➢ Un botón para activar la cámara, que lanzará una ventana que muestre la secuencia que esté captando la webcam, usando la resolución configurada por defecto (la máxima).
- ➢ Un botón para realizar capturas de imágenes de lo que está captando la cámara o del vídeo que se esté reproduciendo, y mostrarlas en una ventana interna nueva.

## 4.3) Codificación

Todos los métodos de la Ventana Principal relacionados con el módulo de vídeo.

# 5) Interfaz general

La aplicación cuenta con una barra de herramientas de carácter general que incluye los siguientes botones asociados a las clásicas opciones de archivo (todos ellos tienen asociado un icono y un “ToolTipText”):

- Nuevo: permite crear una nueva imagen (que aparecerá en una nueva ventana). Además, el usuario podrá indicar el tamaño de la imagen a través de un diálogo que se lance en el momento de la creación.
- Abrir: existen tres botones de abrir dependiendo del tipo de fichero (imagen, sonido y vídeo), cada uno se mostrará en un tipo de ventana interna (o en el caso de los audios, en la lista de reproducción). Además, se establece un filtro para que solo muestre archivos con extensiones de formato permitidas.
- Guardar: permite guardar la imagen de la ventana seleccionada, incluyendo las figuras volcadas en la imagen (esta opción estará desactivada para el vídeo). Además, se podrá almacenar en cualquiera de los formatos reconocidos por Java (JPG, PNG, etc.), que se obtendrá a partir de la extensión indicada por el usuario. Al igual que con Abrir, se establece un filtro para que solo muestre extensiones válidas.
- Ayuda: viene incluido en la barra de menú junto con del menú Archivo (que incluye las opciones “Nuevo”, “Abrir” y “Guardar” indicadas anteriormente). Este dispone de un ítem “Acerca de”, que lanza un diálogo con el nombre del programa, versión y autor.