Sistemas Multimedia

Práctica de Evaluación - Aplicación Multimedia

César Muñoz Reinoso

Curso 2022-2023

16 de junio de 2023

Índice

1. Introducción

Se busca desarrollar una aplicación multimedia que ofrezca una solución integral para administrar diversos tipos de medios, como gráficos, imágenes, sonido y vídeo. Esta aplicación permitirá realizar diversas acciones sobre cada tipo de medio, que van desde la creación y captura hasta la edición, reproducción y procesamiento, adaptadas a las características propias de cada medio. Para facilitar estas tareas, la aplicación contará con un conjunto de menús y barras de herramientas que brindarán las funcionalidades necesarias.

Dividiremos el contenido del sistema multimedia en cinco bloques: gráficos, imagen, sonido y vídeo. En cada uno de ellos explicaremos los requisitos funcionales, el análisis realizado y el diseño implementado. Además mostraremos ejemplos de uso de cada uno con explicación de parte del código utilizado. Utilizaremos la estructura de documentación aprendida en la asignatura Fundamentos de la Ingeniería del Software.

2. Interfaz

2.1. Requisitos Funcionales

RF 1.1 La aplicación contará con una barra de herramientas que incluirá los siguientes botones asociados a las opciones de archivo, con un icono y su respectivo “ToolTipText“.

RF 1.2 La aplicación contará con una barra de menú que incluirá una opción “Archivo“ que tendrá las opciones “Nuevo“, “Abrir“ y “Guardar“.

RF 1.3 La aplicación tendrá una opción “Nuevo“ que permitirá crear una nueva imagen, la cual aparecerá en una nueva ventana.

RF 1.4 La aplicación tendrá una opción “Abrir fichero“ que permitirá seleccionar un fichero de tipo imagen, sonido o vídeo, la cual aparecerá en una nueva ventana. Dependiendo del tipo de fichero, este abrirá un tipo de ventana de imagen, vídeo o en el caso de Audio, este la añadirá a la lista de reproducción. Los tipos de ficheros serán los estándares manejados por Java.

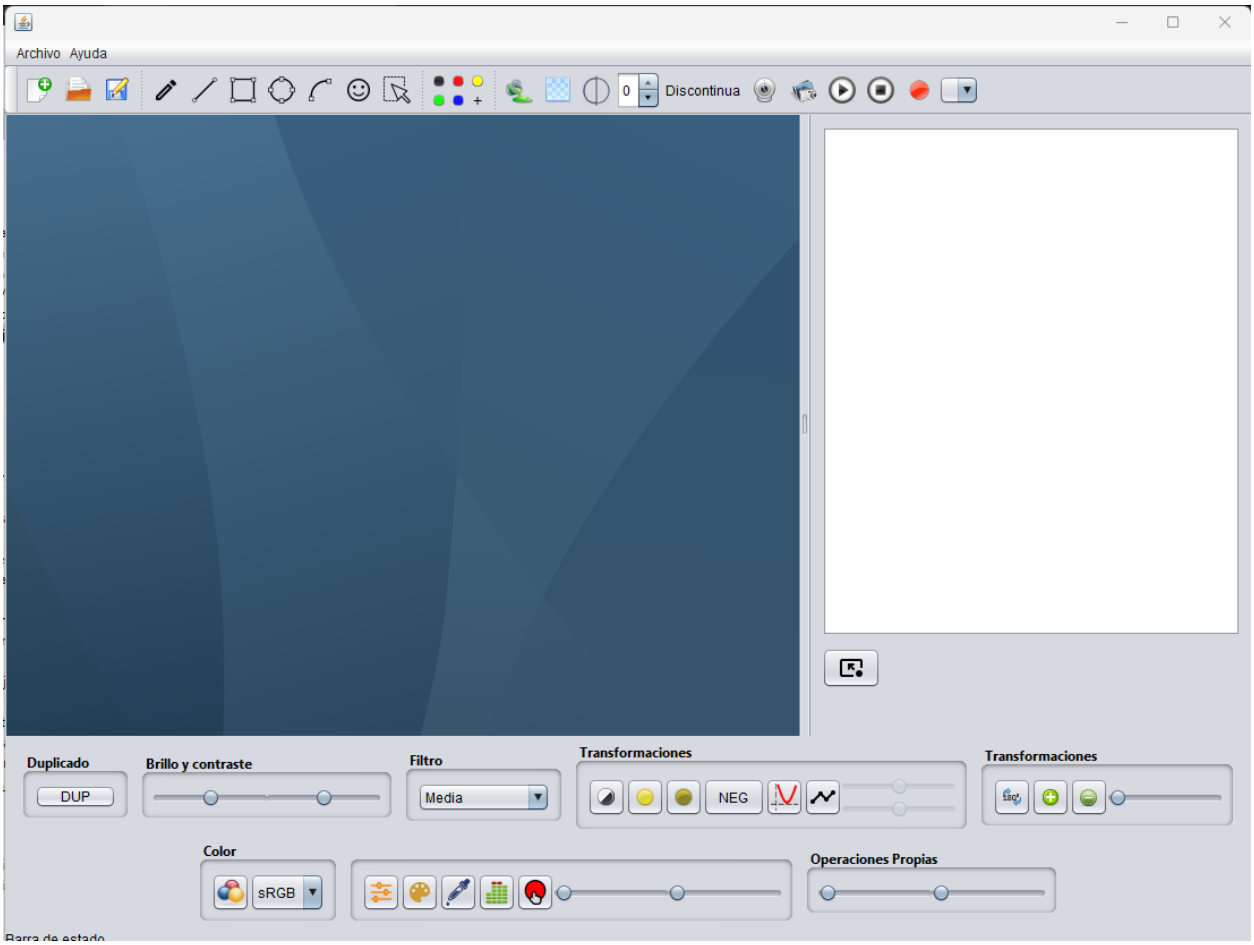
RF 1.5 La aplicación tendrá una opción “Guardar fichero“ que permitirá guardar un fichero de tipo imagen.

RF 1.6 La aplicación contará con una barra de menú que incluirá una opción “Ayuda“ con una sección “Acerca de“ que mostrará un dialogo con el nombre del programa, versión y Autor.

RF 1.7 La aplicación contará con una ventana principal en la que se irán creando las ventanas internas y se podrán redimensionar, cerrar y minimizar, así como poder movernos en ellas con un deslizador horizontal y vertical.

2.2. Análisis, Diseño y Codificación

Para poder cumplir con todos los requisitos funcionales que requiere la práctica, se ha implementado una interfaz gráfica que consta de un menú superior, en el que se encuentran las opciones Archivo, con sus correspondientes opciones Nuevo, Abrir y Guardar, y la opción de Ayuda que contiene Acerca de para mostrar la información del proyecto.



A continuación tenemos una barra de herramientas en la que tenemos también los botones Nuevo, Abrir y Guardar así como herramientas de creación de figuras como Trazo libre, Linea recta, Rectángulo, etc. Una paleta de colores, en la que incluimos un botón para elegir un color personalizado; botones para el relleno, posterizado, transparencia, tamaño de groso y la opción de elegir el trazo discontinuo.Tenemos la sección audiovisual, con botones de Play y Stop para los archivos de vídeo y audio, un botón de grabación de sonido y una lista de reproducción de audios.

En el medio de la interfaz tenemos un panel dividido donde hemos incorporado un escritorio a la izquierda en el que irán apareciendo las diferentes ventanas internas, y a la derecha una lista de figuras creadas en el lienzo en el que nos encontremos. Debajo de dicha lista hemos implementado un botón que dibujará permanentemente las figuras sobre la imagen.

Finalmente en la barra de herramientas inferior tenemos todas las herramientas para el procesado de imágenes. Todas estas herramientas las veremos en detalle en la siguiente sección por lo que no es necesario explicarlas con detenimiento por ahora.

3. Gráficos

3.1. Requisitos Funcionales

RF 2.1 La aplicación permitirá la gestión conjunta de gráficos e imágenes en un mismo tipo de ventana así, dada una imagen podremos tanto dibujar sobre ella como procesarla.

RF 2.2 El lienzo en la ventana interna de imagen mantendrá todas las figuras que se vayan dibujando.

RF 2.3 Cada figura tendrá sus propios atributos, independientes del resto de formas.

RF 2.4 Cuando se dibuje la forma por primera vez, ésta usará los atributos que estén activos en ese momento.

RF 2.5 La aplicación incluirá un panel dividido que, en su parte derecha, muestre una lista con la relación de figuras que hay en la ventana activa, de forma que cada vez que se dibuje una nueva forma, ésta se añada a dicha lista. Cuando nos cambiemos entre ventanas, esta lista se actualizará con las figuras dibujadas en ella.

RF 2.6 El usuario podrá mover las figuras dibujadas mediante una operación de arrastrar y soltar con el ratón.

RF 2.7 El usuario podrá volcar en la imagen todas las figuras seleccionadas en la lista de figuras. Se incluirá un botón debajo de la lista de forma que, al pulsarlo, volcará en la imagen las figuras seleccionadas. Esas figuras serán eliminadas del vector de figuras del lienzo y desaparecerán de la lista.

RF 2.8 La aplicación deberá permitir dibujar las formas geométricas: Trazo libre, Línea recta, Rectángulo, Elipse, Curva con un punto de control, Forma personalizada que defina una cara sonriente. Cada figura tendrá un botón de dos posiciones que estarán en el mismo grupo de botones.

3.2. Análisis

Para poder dibujar figuras sobre el lienzo, haremos uso de las bibliotecas de Java, pero estas no estas no tienen los métodos para poder moverlas o darle los atributos necesarios. Crearemos

nuestras propias clases heredadas de java.awt.geom en las que diseñaremos funciones como pertenencia de un punto a una figura, o mover dichas figuras sobre el lienzo. Además para poder darle atributos, crearemos otras clases que tendrán como variables de instancia las figuras anteriores junto con atributos como grosor, color, relleno, etc.

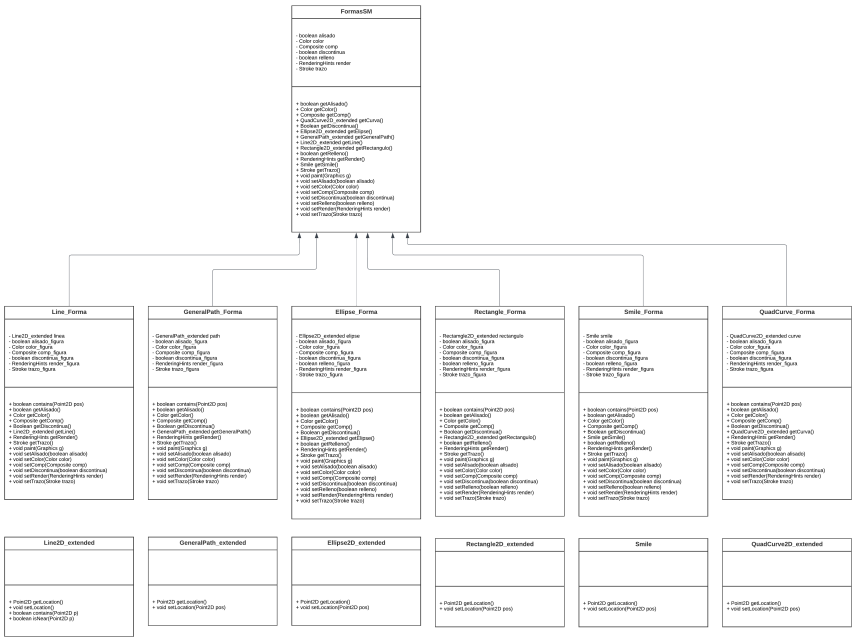
Tenemos una clase Lienzo2D a la que añadiremos dichas figuras mediante clicks y deslizamientos del ratón, además la clase tendrá atributos con las que poder crear las figuras, cada una de ellas con los suyos propios. Para la gestión de ventanas, hemos creado una clase VentanaInternaSM de la que heredan clases especificas dependiendo del tipo de multimedia que queramos mostrar. Tenemos VentanaInternaImagen para mostrar el lienzo para dibujar, así como abrir imágenes desde ficheros, VentanaInternaVideo para abrir archivos de vídeo y VentanaInternaCamara para mostrar la imagen que capta la webcam.

3.3. Diseño y Codificación

Hemos diseñado la clase Lienzo2D siguiendo las directrices de las sucesivas practicas, creando todas las variables de instancia para los diferentes atributos de las figuras, de manera que a medida que se van creando, con el método setAtributos, se le asignan los valores de los atributos actuales del lienzo a la figura recién creada. Hay figuras como por ejemplo el trazo libre, que no tiene relleno, por tanto dicho atributo no se le pasará la figura. Las distintas figuras se crean de manera bastante intuitiva salvo la Curva, esta se crea en 2 pasos, en el primer paso se crean 2 extremos y en el segundo paso se crea el punto que define el arco de la curva.

El botón de volcado (dibujado) de figuras sobre la imagen se encuentra debajo de la lista de figuras que se encuentran en el lienzo. Podemos seleccionar cuantas figuras queramos y al pulsar el botón, mediante el método volcarFiguras, este asigna la imagen actual del lienzo y elimina todas las figuras del vector del lienzo.

Para poder tener una buena visión sobre la herencia de clases de las figuras, he diseñado este diagrama de clases:



4. Imagen

4.1. Requisitos Funcionales

RF 3.1 La aplicación deberá incorporar un botón Duplicar, que creará una nueva Ventana imagen con una copia de la imagen.

RF 3.2 La aplicación deberá incorporar unos deslizadores para poder modificar el brillo y el contraste.

RF 3.3 La aplicación deberá incorporar un desplegable que incluirá filtros de emborronamiento, enfoque y relieve así como emborronamiento horizontal de tamaños 5x1, 7x1 y 10x1.

RF 3.4 La aplicación deberá incorporar un botón que aplicará el filtro Negativo a la imagen.

RF 3.5 La aplicación deberá incorporar un botón que aplicará la función de Operador Cuadrático a la imagen.

RF 3.6 La aplicación deberá incorporar un botón Spline lineal, que habilitará y deshabilitará dos deslizadores para aplicar dicha función a la imagen.

RF 3.7 La aplicación deberá incorporar un botón para realizar la extracción de bandas a la imagen.

RF 3.8 La aplicación deberá incorporar un desplegable para poder realizar la conversión entre los espacios RGB, YCC y GRAY.

RF 3.9 La aplicación deberá incorporar un deslizador para poder realizar una Rotación de la imagen.

RF 3.10 La aplicación deberá incorporar un botones para realizar un Escalado de la imagen, uno para aumentar y otro para disminuir el tamaño.

RF 3.11 La aplicación deberá incorporar un botón para cambiar el Tono de la imagen con el color seleccionado en la paleta.

RF 3.12 La aplicación deberá incorporar un botón que aplicará el filtro Sepia a la imagen.

RF 3.13 La aplicación deberá incorporar un botón que aplicará una Ecualización a la imagen.

RF 3.14 4 La aplicación deberá incorporar un botón que aplicará una Posterización a la imagen.

RF 3.15 4 La aplicación deberá incorporar un botón que aplicará un Resaltado del rojo a la imagen.

RF 3.16 4 La aplicación deberá incorporar un deslizador que realizará un Cambio de tono de la imagen, manteniendo la saturación y el brillo.

RF 3.17 La aplicación deberá incorporar un deslizador que aplicará una operador LookupOp de diseño propio.

RF 3.18 La aplicación deberá incorporar un deslizador que aplicará un operador Pixel a pixel de diseño propio.

4.2. Análisis

Como hemos comentado la VentanaInternaImagen contiene el lienzo sobre el que dibujar las distintas figuras que tenemos a nuestra disposición, moverlas además de aplicarles filtros y efectos. Comentaremos los filtros implementados además de incluir ejemplos de su resultado. Algunos de ellos están ya incluido en las bibliotecas aportadas por las prácticas de la asignatura, otras como LookupOp Propia o OperadorOp Propio serán explicados con detenimiento.

4.3. Diseño y Codificación

Ahora explicaremos algunos de los métodos implementados para el procesamiento de imágenes.

4.3.1. Cuadrática

Para comenzar implementamos una función cuadrática, con un parámetro m que indica donde se hace cero la función. En nuestro sistema multimedia hemos incorporado un botón que realiza el operador con m = 128.

$$f(x) = \frac{1}{100}(x - m), 0 \leq m \leq 255$$

El método implementado genera una LookupTable que se empleará en la función aplicarLookup. En dicha función se establece el valor máximo posible, el cual se divide por el límite superior del intervalo para obtener la constante de normalización (K). Esta constante es necesaria para que esta devuelva valores dentro del rango [0,255] en nuestra función. Se trata de un filtro que realiza la conversión de forma para cada byte, generando la tabla apropiada como resultado.

```
double Max = (m >= 128) ? (1.0 / 100) * (0 - m) * (0 - m) :
      (1.0 / 100) * (255 - m) * (255 - m);
double K = 255.0 / Max;
byte lt[] = new byte[256];
for (int l = 0; l < 256; l++) {
   lt[l] = (byte) (K * (1.0 / 100) * (l - m) * (l - m));
}
```

4.3.2. Spline Lineal

También hemos implementado un spline lineal con un punto de inflexión. Tenemos 2 deslizadores que se activan cuando se pulsa el botón diseñado para el spline lineal, con ellos podemos variar los valores a y b de la función. Al igual que en todos los delizadores, hemos implementado los eventos FocusGained, FocusLost y StateChanged para guardar la imagen original e ir cambiando la imagen a medida de que cambien los deslizadores. Esta vez al tener la función definida en dos diferentes intervalos, creamos dos valores máximos con 2 constantes de normalización y aplicamos la conversión de forma para cada byte, generando la tabla apropiada como resultado.

$$T(x;a,b) = \begin{cases} \frac{b}{a}x & x < a \\ \\ m(x-a)+b & x \geq a \end{cases}$$

$$m = \begin{cases} \frac{255-b}{255-a} & a \neq 255 \\ \\ 0 & a = 255 \end{cases}$$

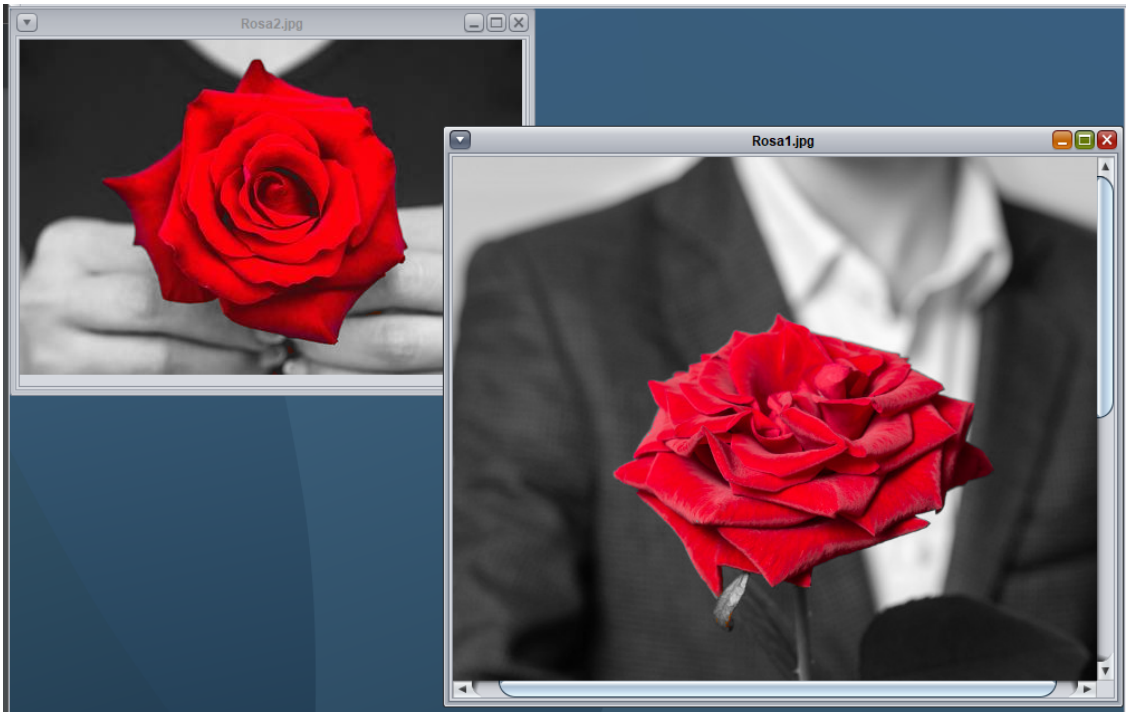
```
double Max_mayor = m * (255 - a) + b;
double Max_menor = (b / a) * 255;

double K_mayor = 255.0 / Max_mayor;
double K_menor = 255.0 / Max_menor;
byte lt[] = new byte[256];
   for (int l = 0; l < 256; l++) {
      if (l < a) {
         lt[l] = (byte) (K_menor * (b / a * l));
      } else {
         lt[l] = (byte) (K_mayor * (m * (l - a) + b));
      }
   }
```

4.3.3. Tono Rojo

Hemos incorporado un filtro que resalte el tono rojo y mantiene el resto en escala de grises, para ello haremos uso de la siguiente función, donde $f(x,y) = [R_f, G_f, B_f]$ es la imagen original y $g(x,y)$ la imagen resultado. Mediante un umbral que hemos definido a 20, comprobamos si la diferencia de verde y azul con el rojo es menor y en ese caso difumina en escala de grises ese pixel; si supera el umbral significa que el pixel es rojo y se mantiene en su color, copiándose al nuevo vector de pixeles.

$$g(x,y) = \begin{cases} f(x,y) & R_f - G_f - B_f \geq T \\ \frac{R_f - G_f - B_f}{3} & R_f - G_f - B_f < T \end{cases}$$

```
Color color = new Color(src.getRGB(x, y));
int rgb[] = {color.getRed(), color.getGreen(), color.getBlue()};
if (color.getRed() - color.getGreen() - color.getBlue() < umbral) {
   int media = (color.getRed() + color.getGreen() + color.getBlue()) / 3;
   for (int i = 0; i < pixelCompDest.length; i++) {
      pixelCompDest[i] = media;
   }
} else {
   System.arraycopy(rgb, 0, pixelCompDest, 0, pixelCompDest.length);
}
```

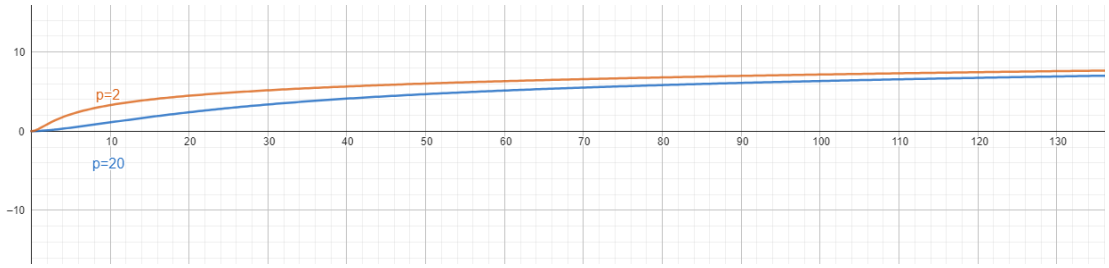


4.3.4. LookupOp Propia

Como requería el guión, hemos creado un filtro Lookup con un parámetro definido en [2,20]. Como en anteriores casos hemos calculado el máximo valor que alcanza la función y por consiguiente, la constante de normalización.

Podemos ver como se comporta la función para los valores extremos del parámetro p. En nuestro caso la función es parecida a la logarítmica, salvo que la adición de la arcotangente consigue suavizar la curva al inicio, pudiendo tener valores en todo el intervalo de definición [0,255].

$$f(x) = \ln(1+x) + \arctan(\frac{x}{p})$$




```
double Max = Math.log(1 + 255f) * Math.atan(255f / p);
double K = 255.0 / Max;
byte lt[] = new byte[256];
for (int l = 0; l < 256; l++) {
   lt[l] = (byte) (K * (Math.log(1 + l) * Math.atan(l / p)));
}
```

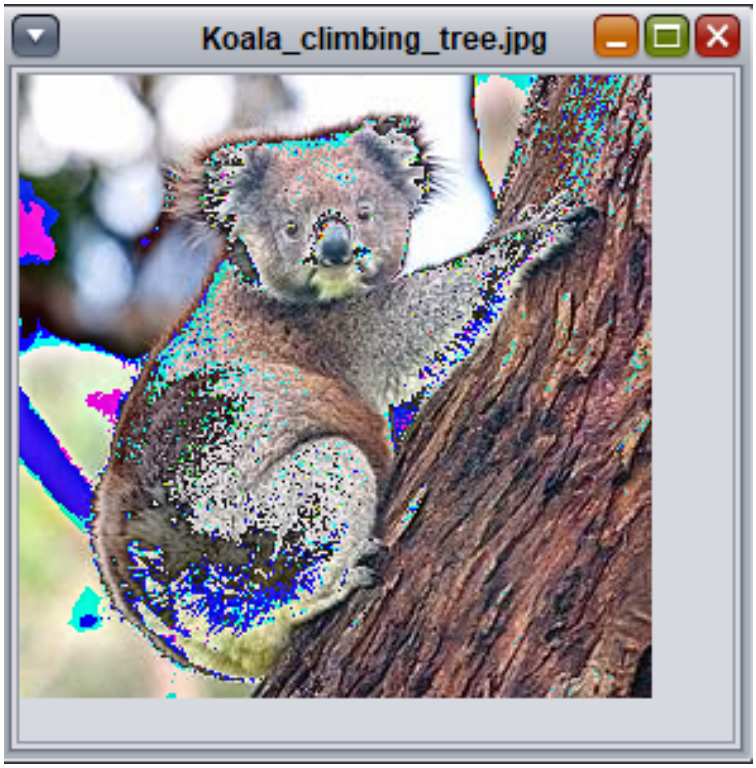
4.3.5. OperadorOp Propia

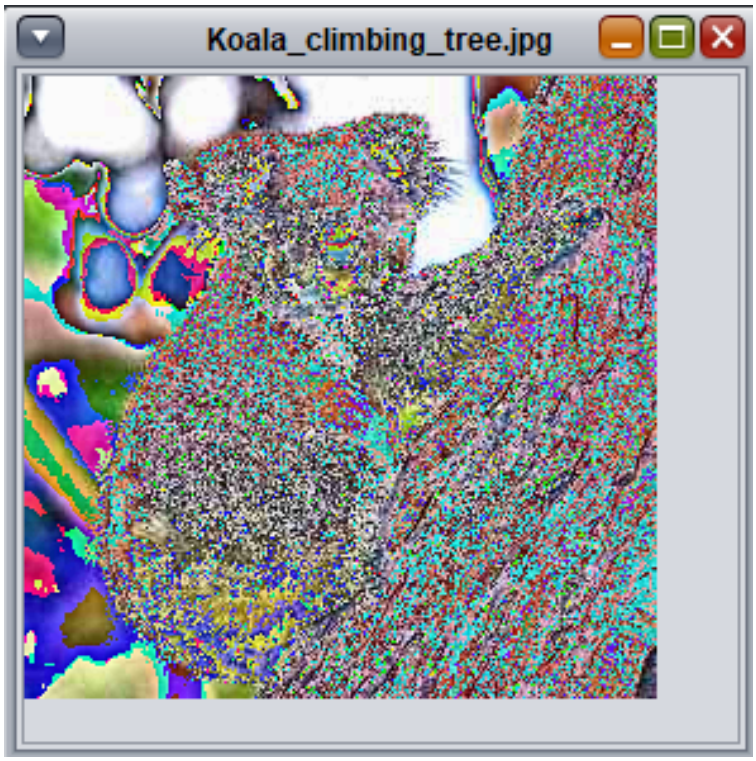
Implementamos un operador propio que actúa pixel a pixel. En este caso haciendo uso de lo aprendido en clases de prácticas, hemos creado una clase OperadorOpPropia heredada de BufferedImageOpAdapter. En ella tenemos como variable de instancia el parámetro que como en la anterior función obtendremos de un deslizador. Hemos sobrecargado también el método filter en el que se modifica el raster multiplicando cada banda de cada pixel por param/2. Vemos en los ejemplos para los extremos del intervalo, p = 2 como aún se pueden diferenciar los colores, mientras que cuando p = 10 apenas se puede diferenciar la silueta.

$$f(x) = pR_f/2, pG_f/2, pB_f/2$$

```
pixelCompDest[0] = param * (pixelComp[0]) / 2;
pixelCompDest[1] = param * (pixelComp[1]) / 2;
```

```
pixelCompDest[2] = param * (pixelComp[2]) / 2;
```





5. Sonido

5.1. Requisitos Funcionales

RF 4.1 La aplicación permitirá tanto la reproducción como la grabación de audio. Tendrá una lista de reproducción con una lista desplegable asociada. Cuando abramos un archivo de audio, este se añadirá a dicha lista.

RF 4.2 La aplicación deberá incorporar un botón para reproducir sonido. Al pulsar el botón, se reproduciría el audio que esté seleccionado en ese momento en la lista de reproducción.

RF 4.3 La aplicación deberá incorporar un botón para parar la reproducción o la grabación.

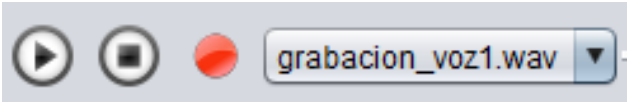
RF 4.4 La aplicación deberá incorporar un botón para grabar audio, de forma que al pulsarlo se iniciará el proceso de grabación, que terminará cuando se pulse el botón de parada. El sonido se grabará en el fichero indicado por el usuario y se añadirá a la lista de reproducción.

5.2. Análisis

Para esta sección hemos seguido el guión de la practica 13 en cuanto a diseño de botones y gestión de eventos se refiere.

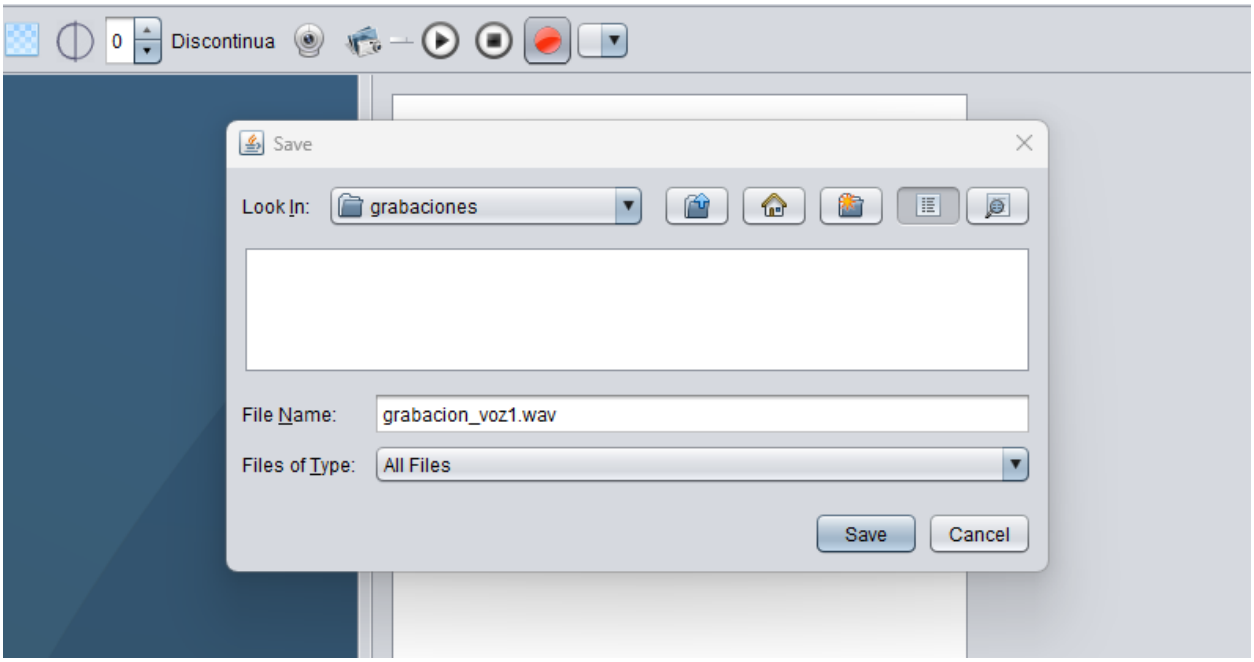
5.3. Diseño y Codificación

En la barra superior de herramientas hemos creado un botón de reproducción, que servirá tanto para vídeo como para audio, este se deshabilitará cuando el medio de reproducción este activo, así solo podremos pulsar el botón de parada. Este botón de parada también nos sirve para video como audio.



Como hemos comentado, para abrir un archivo de audio utilizaremos el botón de abrir ficheros, y este se añadirá a la lista desplegable, desde la que podremos seleccionar el audio a reproducir.

Tenemos también un botón de grabación el cual nos abrirá un dialogo para poder guardar el archivo de audio en la ubicación y con el nombre que deseemos, automáticamente se comenzará a grabar y deberemos pulsar el botón de parada para finalizar la grabación. Este archivo se incluirá en la lista desplegable y se seleccionará por defecto.



6. Vídeo

6.1. Requisitos Funcionales

RF 5.1 La aplicación permitirá tanto la reproducción de vídeo como la captura de la cámara.

RF 5.2 La aplicación deberá incorporar un botón para reproducir vídeo, que sea el mismo que para la reproducción de sonido.

RF 5.3 La aplicación deberá incorporar un botón para parar la reproducción, que sea el mismo que para la reproducción de sonido.

RF 5.4 La aplicación deberá incorporar un botón para reproducir la imagen que nos ofrece la webcam.

RF 5.5 La aplicación deberá incorporar un botón para realizar una captura de imagen de la webcam o del vídeo.

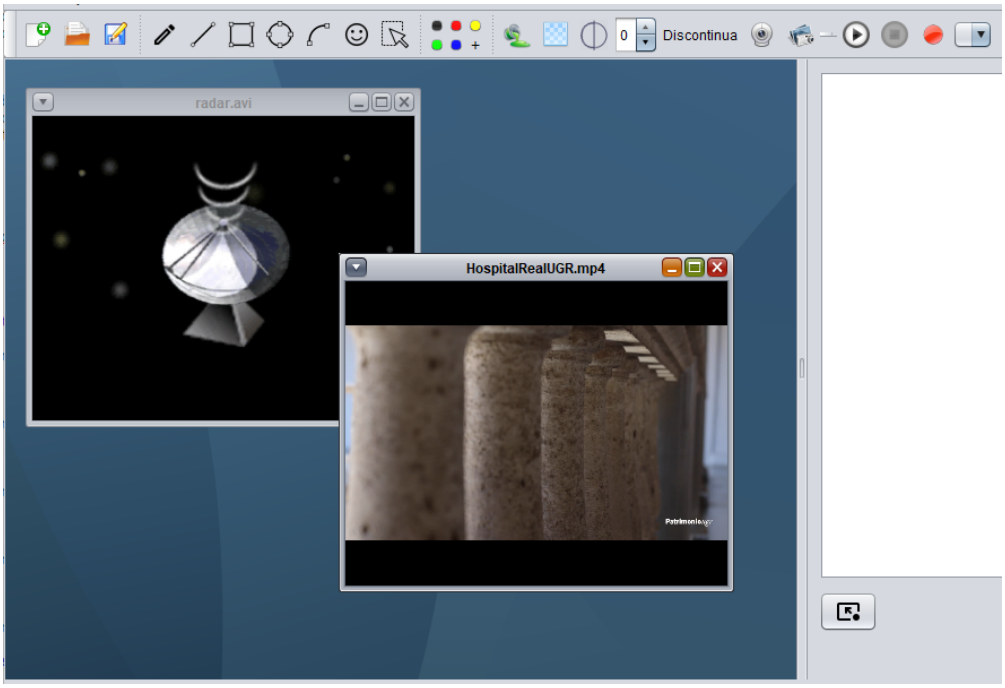
6.2. Análisis

Para esta sección hemos seguido el guión de la practica 14 en cuanto a diseño de botones, reproducción usando VLCj y gestión de eventos se refiere.

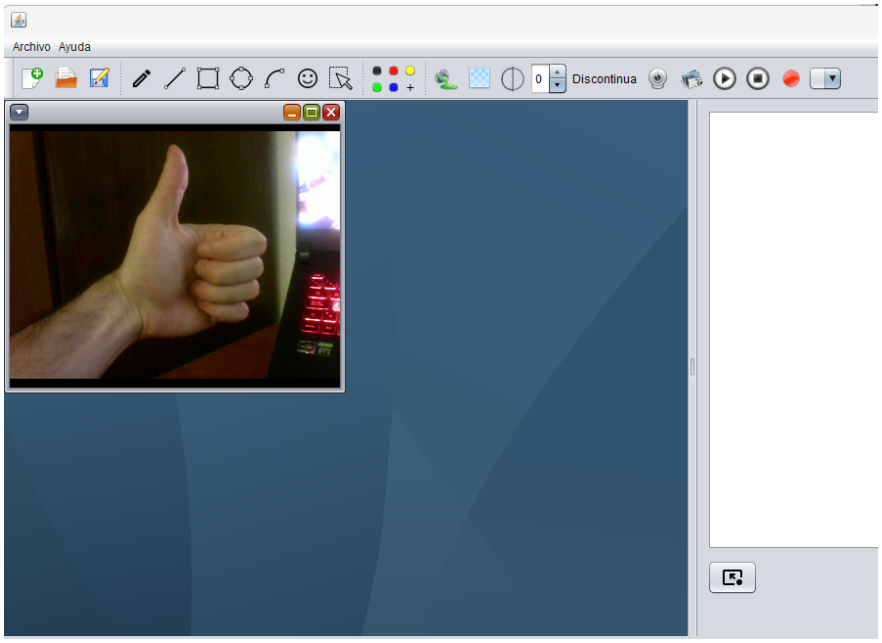
6.3. Diseño y Codificación

Al igual que con el sonido, la reproducción de vídeo, en la barra superior de herramientas tenemos los botones de reproducción y pausa, así como un botón para reproducir la imagen que nos ofrece la webcam y un botón para realizar una captura del vídeo o webcam que tengamos.

Para la gestión del vídeo hemos creado una clase VentanaInternaVideo heredada de VentanaInternaSM, en el que tenemos el reproductor vlcplayer y el archivo que abrimos con el botón anteriormente explicado. En dicha clase se han implementado métodos para poder reproducir y parar el vídeo, así como un listener para la ventana.



Para la gestión de la webcam hemos creado otra clase VentanaInternaCamara también heredad de VentanaInternaSM, en la que tenemos como variable la webcam de nuestro equipo y métodos sobrecargados para poder acceder a su imagen desde la clase superior.



Incluimos un botón que independiente la ventana sea VentanaInternaVideo o VentanaInterna-Camara, esta realiza una captura a la imagen de la ventana y la muestra en una nueva ventana de imagen. Dichas funciones de captura están definidas en cada clase heredada.