



UNIVERSIDAD DE GRANADA

SISTEMAS MULTIMEDIA
GRADO EN INGENIERÍA INFORMÁTICA

Manual de usuario

SMM 2018/2019

Autora
Nazaret Román Guerrero



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

CURSO 2018-2019

Índice

1. Ventanas internas	2
1.1. Análisis de requisitos	2
1.2. Explicación de cada requisito	2
1.3. Codificación	2
2. Gráficos	4
2.1. Análisis de requisitos	4
2.2. Explicación de cada requisito	5
2.3. Codificación	6
2.4. Funcionamiento del sistema	7
3. Imagen	14
3.1. Análisis de requisitos	14
3.2. Explicación de cada requisito	15
3.3. Codificación	16
3.3.1. Brillo	16
3.3.2. Emborronamiento	17
3.3.3. Enfoque	17
3.3.4. Relieve	17
3.3.5. Contraste	18
3.3.6. Contraste con iluminación	18
3.3.7. Contraste con oscurecimiento	19
3.3.8. Negativo	19
3.3.9. Extracción de bandas y espacios de color	19
3.3.10. Rotación	21
3.3.11. Zoom-in	21
3.3.12. Zoom-out	21
3.3.13. Tintado (mediante botón)	22
3.3.14. Ecualización	22
3.3.15. Sepia	23
3.3.16. Umbralización	23
3.3.17. Función propia	24
3.3.18. Termal	25
3.3.19. Rayos X	26
3.3.20. Suma de imágenes	27
3.3.21. Resta de imágenes	28
3.3.22. Tintado (mediante deslizador)	28
4. Sonido	30
4.1. Análisis de requisitos	30
4.2. Explicación de cada requisito	30
4.3. Codificación	30

4.3.1. Reproducción	31
4.3.2. Grabación y temporizador	31
4.3.3. Lista de archivos	32
5. Vídeo	33
5.1. Análisis de requisitos	33
5.2. Explicación de cada requisito	33
5.3. Codificación	33
5.3.1. Reproducción	33
5.3.2. Uso de la cámara y capturas	34
6. Bibliografía y fuentes de código	36

1. Ventanas internas

1.1. Análisis de requisitos

Para poder gestionar gráficos, imágenes y vídeo en la misma aplicación, debe existir una jerarquía de ventanas internas que se encarguen de las distintas partes:

- RF1. Ventana interna encargada de la gestión de gráficos.
- RF2. Ventana interna encargada de la gestión de imágenes y gráficos.
- RF3. Ventana interna encargada de la gestión de la webcam del dispositivo.
- RF4. Ventana interna encargada de la gestión de reproducción de vídeo.

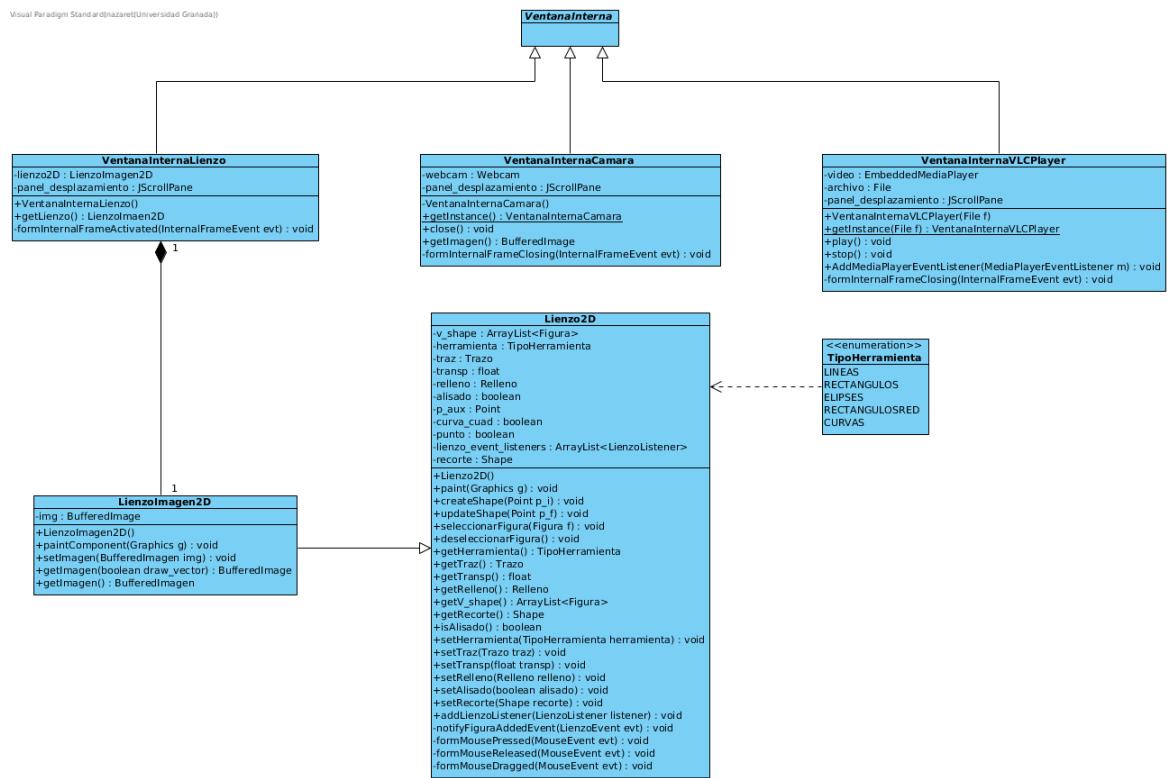
1.2. Explicación de cada requisito

Para cada uno de los requisitos anteriores, se pide:

- RF1. Se abrirá una ventana interna que gestione los gráficos, manteniendo todas las figuras en el lienzo, y cada una con sus propiedades.
- RF2. Se abrirá una ventana interna que gestione el procesamiento de imágenes y además, gráficos, manteniendo todas las figuras en el lienzo, cada una con sus propiedades.
- RF3. Se abrirá una ventana interna que gestione la cámara del dispositivo y que mostrará en tiempo real lo que capte la webcam.
- RF4. Se abrirá una ventana interna que gestione el vídeo, reproduciendo el que el usuario elija.

1.3. Codificación

Para que esto haya sido posible, se ha creado una jerarquía de ventanas internas que se puede observar en el siguiente diagrama de clases:



Como se puede observar, existe una superclase abstracta, *VentanaInterna*, de la cual heredan las demás. Esta clase solo se utiliza para hacer el casting correspondiente y evitar errores en el código.

La VENTANAINTERNACAMARA es la encargada de ejecutar la webcam; la clase VENTANAINTERNALIENZO es la encargada de ejecutar el vídeo; y por último, la clase VENTANAINTERNALIENZO es la encargada de gestionar gráficos e imágenes. Para ello, la clase tiene un atributo de tipo LIENZOLIMAGEN2D que permite el procesado de imágenes; ésta a su vez hereda de LIENZO2D, capaz de gestionar los gráficos creados por el usuario en el lienzo.

Este diagrama será incluido con el nombre de VI.PNG para que sea más legible puesto que en el documento puede ser difícil de leer.

2. Gráficos

2.1. Análisis de requisitos

En lo referente a gráficos, la aplicación debe ser capaz de:

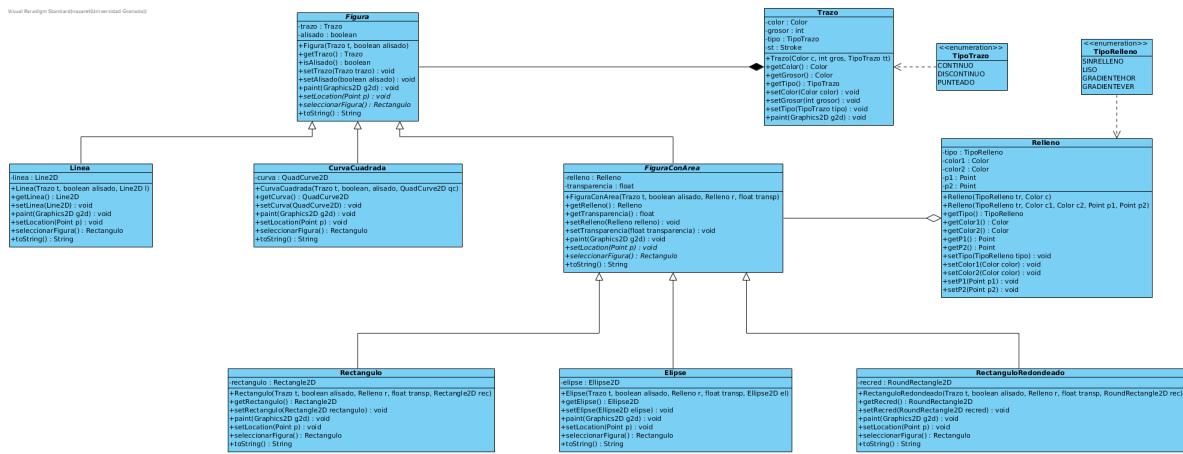
- RF1. El lienzo mantendrá todas las figura a medida que se dibujen.
- RF2. Cada figura tendrá sus propios atributos:
 - Color del trazo.
 - Grosor del trazo.
 - Alisado.
 - Tipo de relleno.
 - Color/colores del relleno.
 - Transparencia.
- RF3. Habrá una lista de figuras a través de la cuál seleccionar la figura que se deseé.
- RF4. La figura que se seleccione deberá estar enmarcada en un rectángulo rojo discontinuo.
- RF5. Al seleccionar la figura, se marcarán sus atributos.
- RF6. Se podrán editar todos los atributos de la figura seleccionada al cambiarlos en la barra de herramientas.
- RF7. Se podrá cambiar la localización de la figura.
- RF8. Habrá diferentes herramientas para dibujar.
- RF9. Se podrá elegir el color del trazo y el color (o colores) del relleno de forma independiente uno de otro.
- RF10. Se podrá elegir el color, grosor y discontinuidad del trazo.
- RF11. Se podrá elegir el color (o colores) y el tipo de relleno.
- RF12. Habrá tres tipos de relleno: liso, gradiente vertical y gradiente horizontal.
- RF13. Se podrá elegir alisar los bordes de la figura.
- RF14. Se podrá establecer la transparencia de la figura mediante un deslizador.
- RF15. Cuando se cambie de lienzo, se actualizarán las propiedades en la barra de herramientas.

2.2. Explicación de cada requisito

- RF1. En el lienzo se crearan y mantendrán todas las figuras que el usuario dibuje.
- RF2. Los atributos de cada figura serán independientes por lo que cada figura puede tener un color, grosor, relleno, etc. diferentes.
- RF3. Se creará una lista desplegable con las figuras de lienzo, a través de la cuál se podrá seleccionar la figura que se deseé.
- RF4. Una vez se elija la figura, esta se resaltará mediante un rectángulo a su alrededor, de color rojo y trazo discontinuo.
- RF5. Cada vez que se seleccione una figura del lienzo, los atributos de ésta se verán reflejados en la barra de herramientas.
- RF6. En una figura seleccionada, se podrá cambiar el atributo que se deseé cambiando las propiedades en la barra de herramientas. Los cambios se verán automáticamente.
- RF7. Habrá un botón para mover la figura, que creará cuadros de diálogo en pantalla para indicar el desplazamiento en las coordenadas x e y.
- RF8. Se podrán dibujar líneas, rectángulos, elipses, rectángulos con esquinas redondeadas y curvas con un punto de control.
- RF9. Habrá que elegir el color del trazo y el color del relleno (o colores, ya que habrá que elegir dos en el caso de que sea degradado).
- RF10. Podrá variar el color, el grosor y el tipo de discontinuidad del trazo. Entre los tipos de discontinuidad se incluirán tres: continuo, discontinuo y punteado.
- RF11. Se elegirá el color del relleno en el caso de que sea liso, o los colores (dos) en el caso de que sea degradado.
- RF12. Existirán cuatro posibilidades de relleno: sin relleno, relleno liso (con un solo color) y relleno degradado en vertical u horizontal (con dos colores).
- RF13. Se podrá elegir si alisar los bordes de cada figura o no alisarlos.
- RF14. Se podrá modificar la transparencia de cada figura mediante un deslizador. Variará desde totalmente transparente a opaco.
- RF15. Cuando haya varios lienzos disponibles y se cambie entre ellos, habrá que seleccionar las propiedades del lienzo seleccionado en la barra de herramientas.

2.3. Codificación

Para llevar a cabo todos los requisitos anteriores, se ha creado una jerarquía de clases que contienen las distintas figuras, cada una con sus propiedades. La jerarquía definida es:



Al igual que ocurre con el diagrama anterior, será incluido con el nombre de FIGURAS.PNG para que pueda verse con más facilidad y que se lean bien los atributos y métodos de cada clase. Además, se incluye el archivo GENERAL.PNG que contiene la unión de ambos diagramas de forma que se pueda ver como funciona la aplicación a grandes rasgos.

Como se puede observar en la imagen, se ha definido una jerarquía que consta de una clase abstracta FIGURA, que contiene un objeto de tipo TRAZO con el que se pintarán las figuras.

Tanto la línea como la curva implementan directamente de esta clase, ya que no necesitan atributos extra; ni una línea ni una curva tienen relleno, y por tanto, tampoco pueden ser transparentes ya que entonces desaparecerían. Por ese motivo, en mi jerarquía, ni una línea ni una curva pueden tener algún grado de transparencia distinto de 1.0, es decir, opaco.

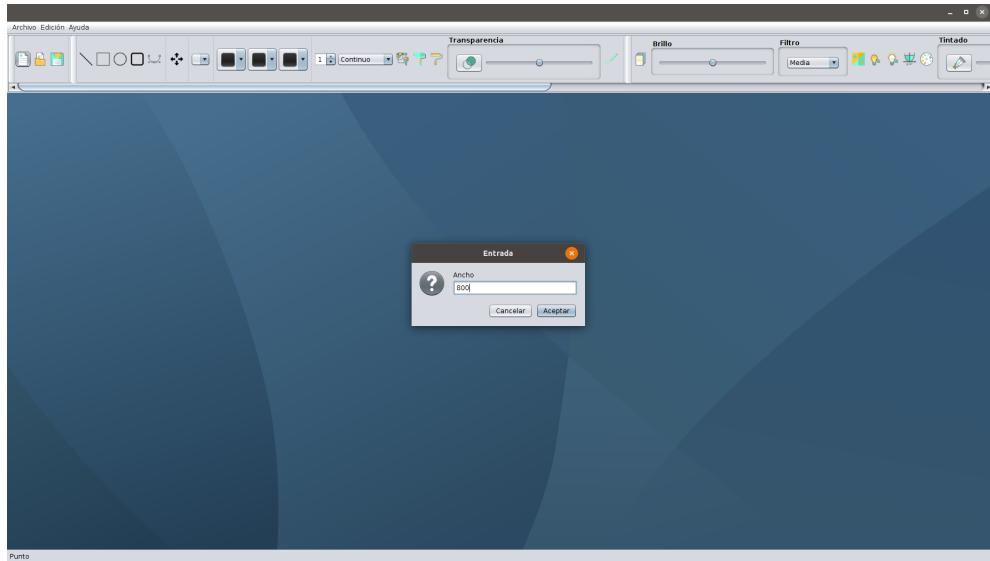
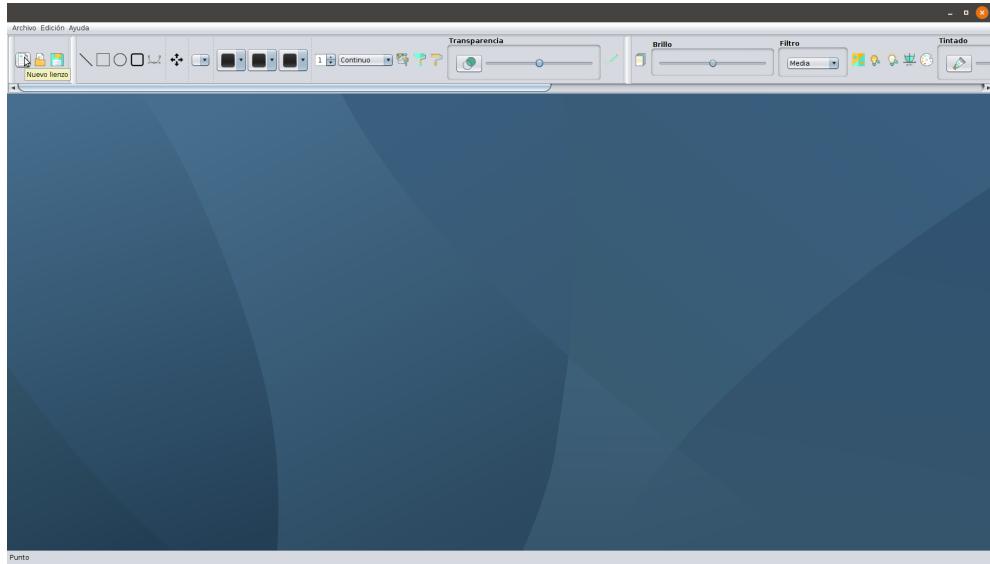
Por otro lado, tenemos la clase FIGURAConAREA que de nuevo es abstracta, y que hereda directamente de FIGURA. Esta clase está pensada para figuras que tienen superficie o área y que por tanto, pueden tener un relleno y una transparencia. Por ello esta clase tiene un objeto de tipo RELLENO, que contiene todas las características que puede tener, como el color y el tipo de degradado.

Por último, nos encontramos con las clases que representan las figuras del rectángulo, la elipse y el rectángulo con las esquinas redondeadas. Estas figuras tienen área por lo que heredan de FIGURA CON AREA.

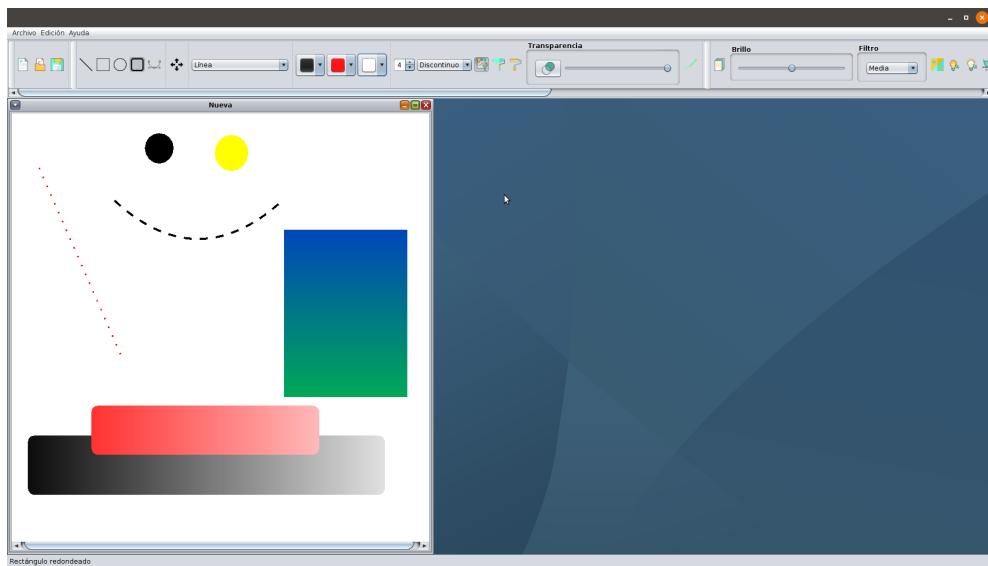
2.4. Funcionamiento del sistema

Los requisitos anteriores se van a ir mostrando ahora paso a paso para demostrar el funcionamiento esperado:

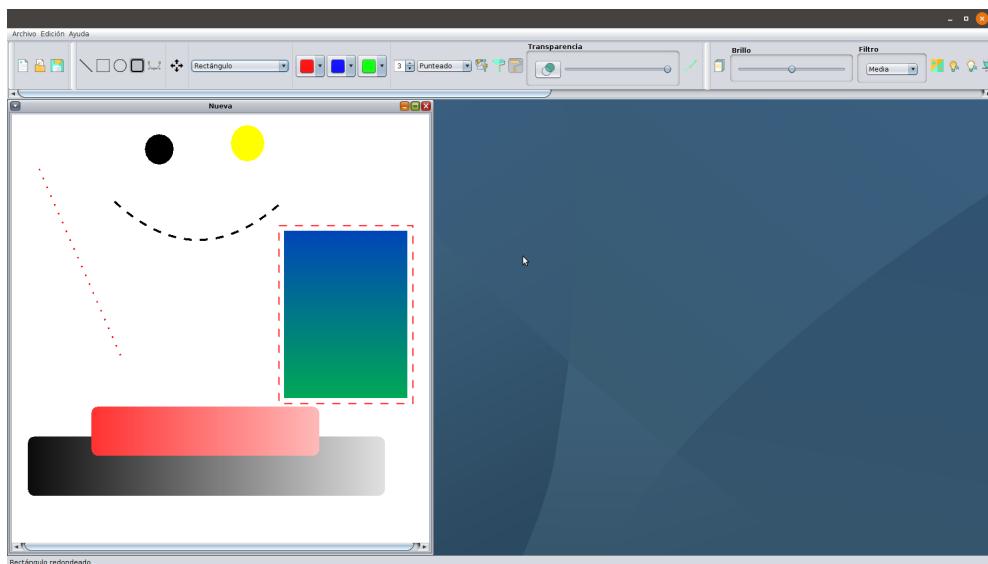
- Creación de un nuevo lienzo. El usuario elige las medidas que desea para el lienzo.

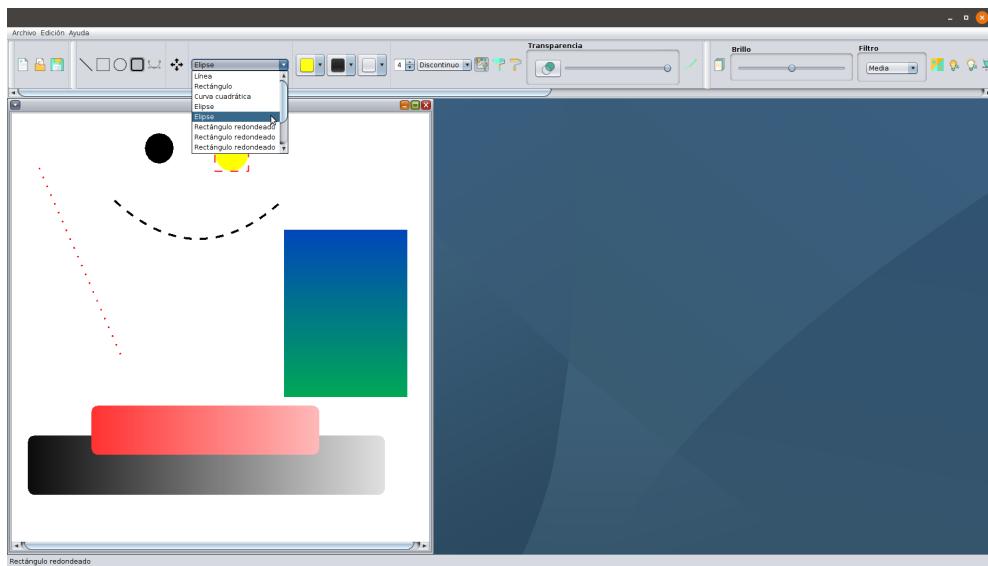


- Se pintan diversas figuras en el lienzo, cada una con sus propiedades. Se muestran todas las formas posibles, el relleno liso, el relleno con gradiente horizontal y el relleno con gradiente vertical.

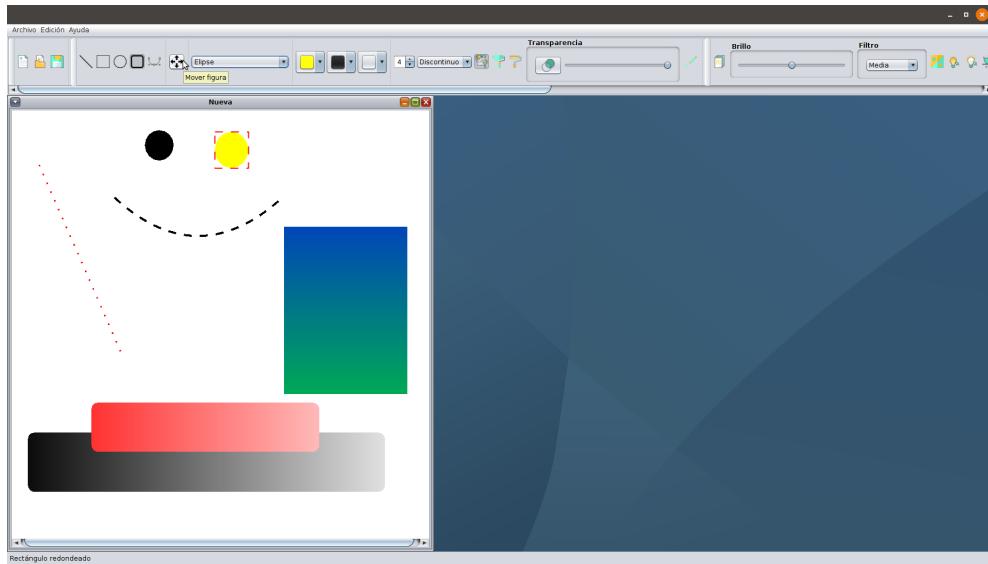


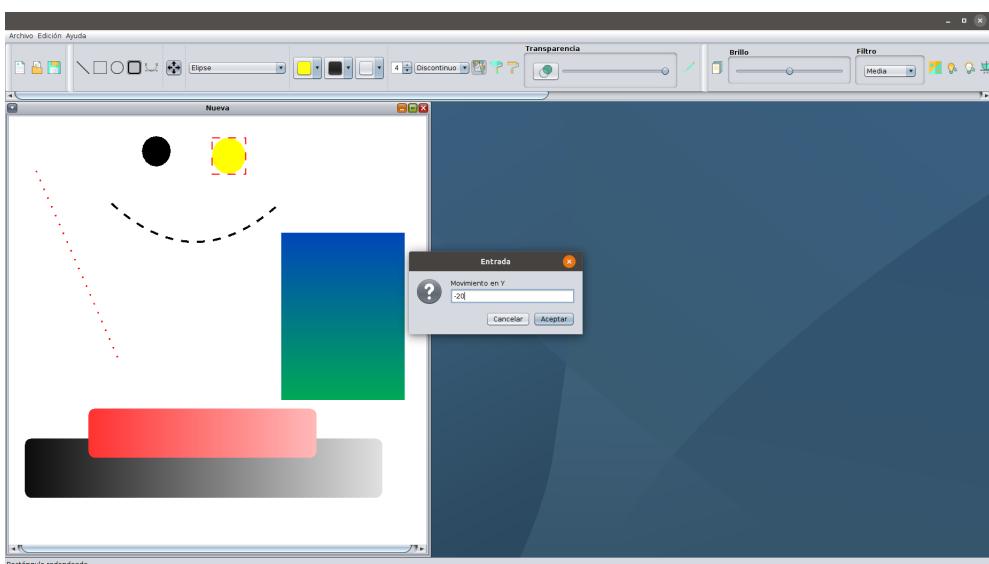
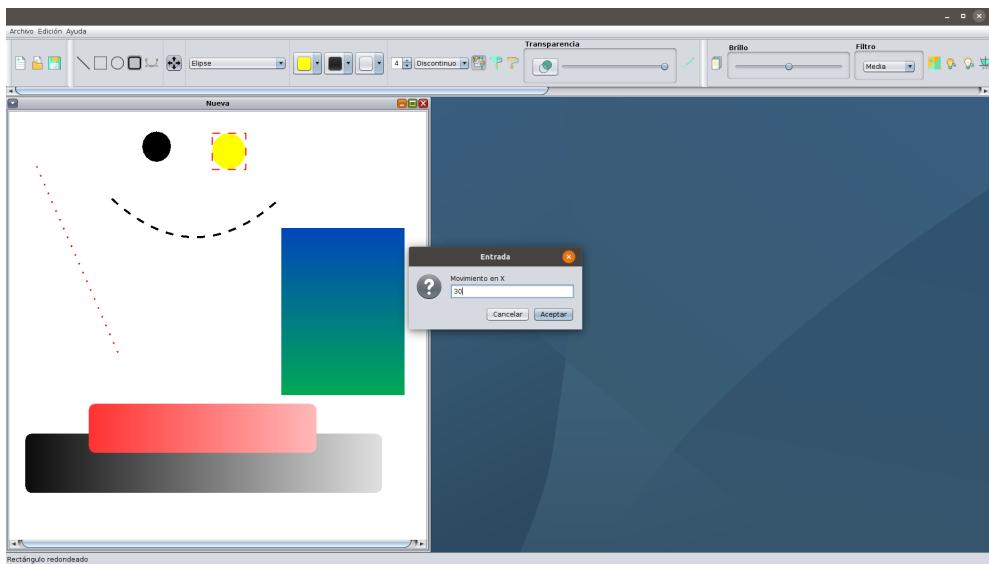
- Seleccionamos dos figuras, primero un rectángulo con degradado y después otra figura, la elipse amarilla. Como se puede observar, al seleccionarlas a través de la lista desplegable, todos los botones y desplegables (excepto el botón de herramienta o aquellos elementos que no tengan valor para la figura seleccionada, como son en este caso, por ejemplo, el segundo color del relleno en la elipse, o el grosor y tipo de trazo) se actualizan en la barra de herramientas, tomando los valores de la figura que hemos seleccionado. Además, se remarca con un rectángulo rojo discontinuo para que sea más sencillo identificar qué figura es.

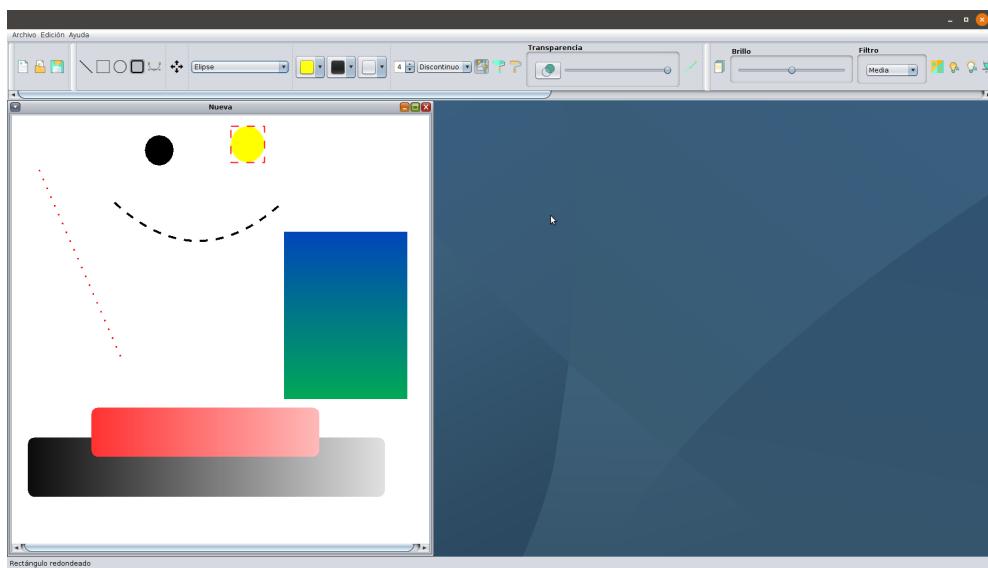




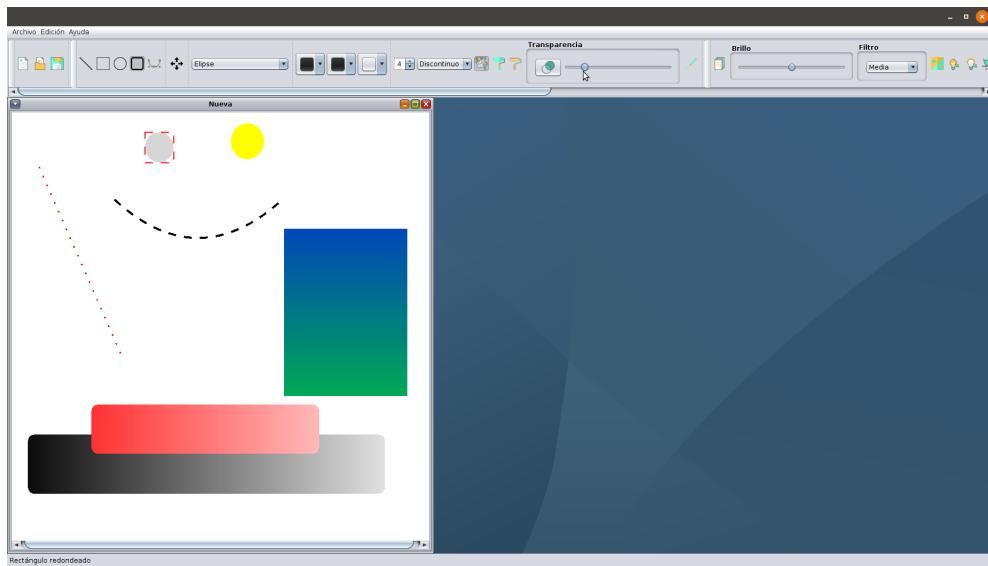
- Ahora vamos a mover la figura seleccionada. Para ello, pulsamos el botón de mover; se abrirán dos diálogos en los que el usuario introducirá el desplazamiento que desea. Éste puede ser negativo en el caso de que se quiera mover hacia arriba o hacia la izquierda y positivo si se quiere mover hacia abajo o hacia la derecha.





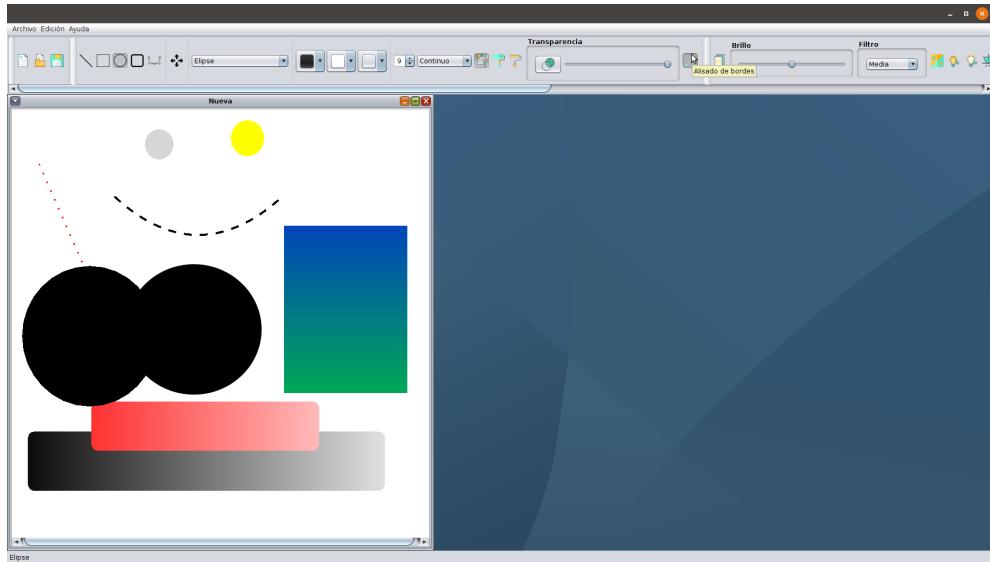


- Ahora modificaremos alguna propiedad de una figura, como por ejemplo la transparencia de la elipse negra. Para ello, en este caso he utilizado el deslizador que varía de totalmente transparente a opaca. No obstante, también está incluido el botón de al lado del deslizador que ofrece la posibilidad de la semitransparencia, es decir, la transparencia con una potencia de 0.5.

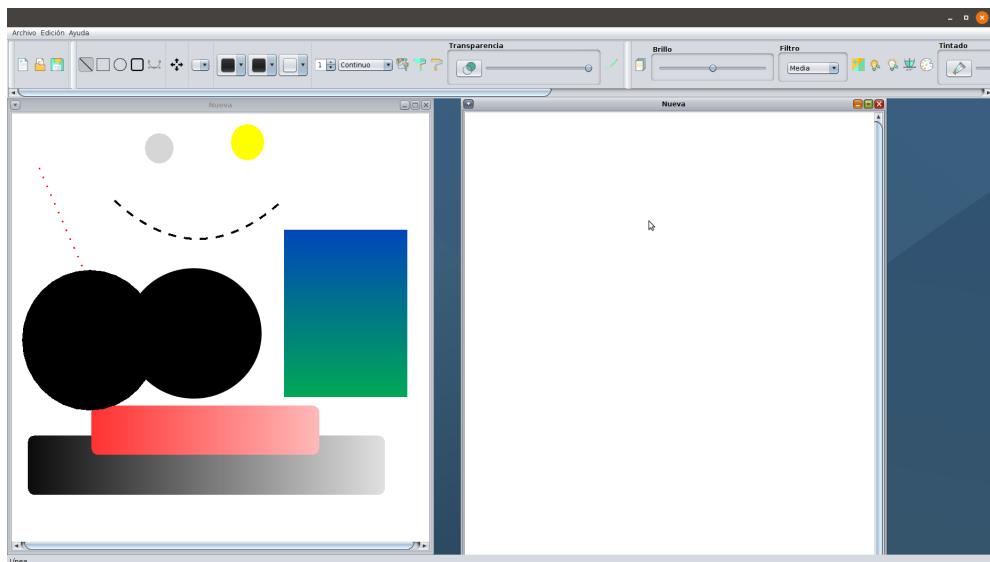


- Como última propiedad que queda por mostrar de las figuras, se muestra el alisado de los bordes, el botón que hay a la derecha del deslizador de transparencia. Para que se vea el efecto, hay dos elipses negras dibujadas, la de la izquierda sin alisado y la de la derecha con alisado. Como se puede observar, el borde de la elipse de la derecha es

mucho más suave, sin “escalones”.

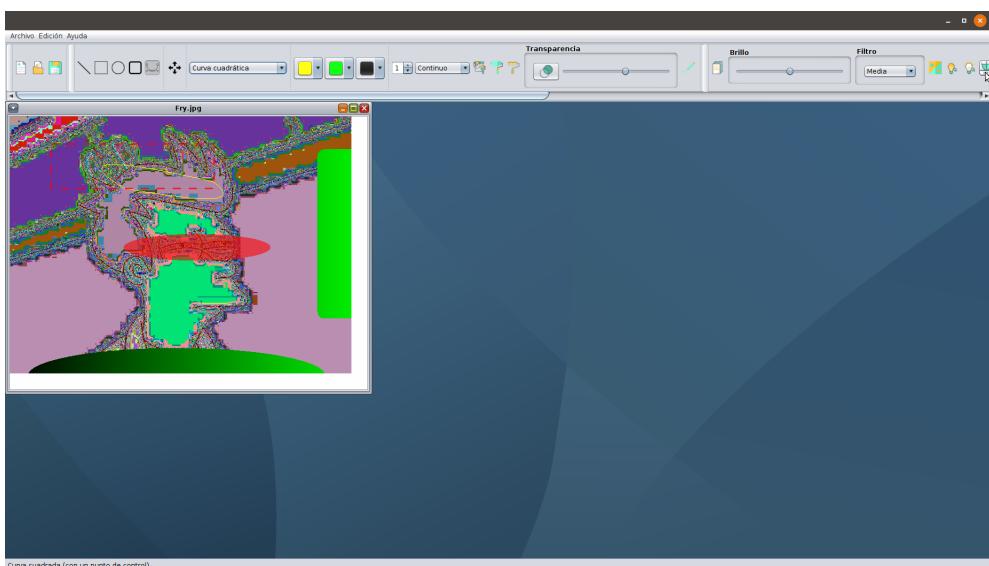
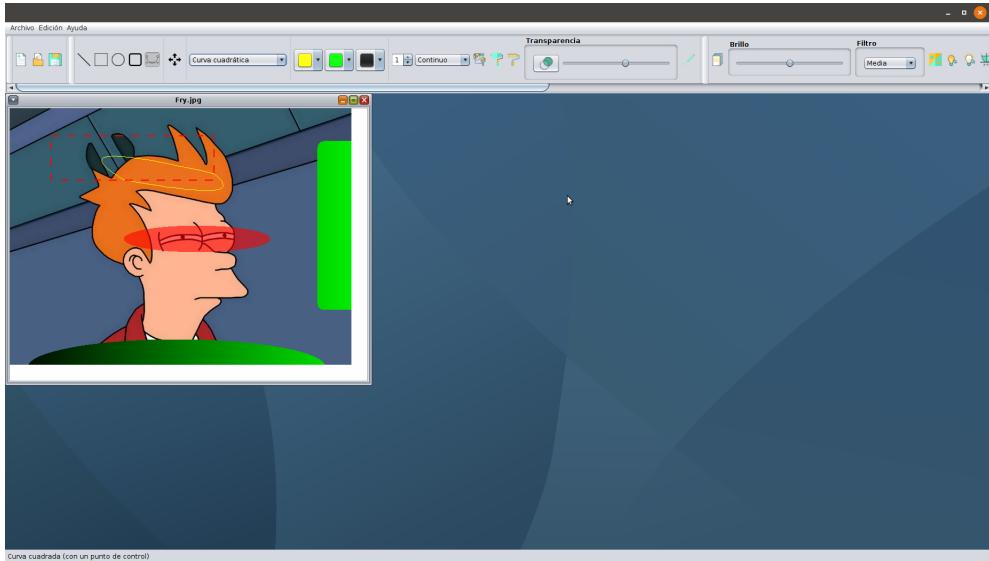


- Ahora, abriremos un nuevo lienzo y observaremos como las propiedades en la barra de herramientas cambian, en este caso a las que tiene un lienzo por defecto.



- Por último, se muestra como se puede dibujar sobre lienzos con imágenes y también procesar dichas imágenes. Las figuras que se dibujen se mantendrán solo dentro de la propia imagen, es decir, las figuras no se pintarán sin sobrepasan los límites de la imagen, como se puede observar en la primera imagen. Además, se puede procesar la fotografía poniendo filtros sobre esta, rotándola... aunque estos efectos no se verán

reflejados sobre las figuras.



3. Imagen

3.1. Análisis de requisitos

En la parte de procesamiento de imágenes, los requisitos funcionales que se le piden al sistema son:

- RF1. Duplicado de imágenes.
- RF2. Modificación del brillo mediante un deslizador.
- RF3. Filtro de emborronamiento.
- RF4. Filtro de enfoque.
- RF5. Filtro de relieve.
- RF6. Contraste.
- RF7. Contraste con iluminación.
- RF8. Contraste con oscurecimiento.
- RF9. Efecto negativo.
- RF10. Extracción de bandas de la imagen.
- RF11. Conversión a distintos espacios de color (RGB, YCC, GRAY).
- RF12. Rotación de la imagen mediante un deslizador.
- RF13. Zoom-in.
- RF14. Zoom-out.
- RF15. Efecto tintado con el color seleccionado.
- RF16. Ecualización de la imagen.
- RF17. Filtro sepia.
- RF18. Umbralización en niveles de gris mediante un deslizador.
- RF19. Filtro basado en la función $f(x) = |x^3 + |\cos(xw)| + 1| \% 255$.
- RF20. Filtro termal.
- RF21. Filtro rayos-x.
- RF22. Suma de imágenes.
- RF23. Resta de imágenes.
- RF24. Efecto tintado con el color seleccionado mediante un deslizador.

3.2. Explicación de cada requisito

Para cada requisito mencionado previamente, se pide al sistema que efectúe lo siguiente:

- RF1. Se debe crear una copia de la imagen independiente. Una vez creada, su título será el título de la imagen original unido a la palabra “copia”.
- RF2. El deslizador indicará el brillo que se le aplicará a la imagen que se está procesando.
- RF3. Se llevará a cabo el emborronamiento de la imagen cuando se seleccione esta opción.
- RF4. La imagen se hará más nítida cuando esta opción se seleccione.
- RF5. Se resaltará el relieve de la imagen que se está procesando.
- RF6. Cuando se elija esta opción, se contrastará la imagen, es decir, los tonos oscuros se oscurecerán más y los tonos claros se aclararán.
- RF7. Esta operación implica que los tonos claros se mantienen igual mientras que los tonos oscuros se aclararán.
- RF8. En este caso, los tonos oscuros se mantendrán igual, y los tonos claros se oscurecerán.
- RF9. Se generará una imagen donde los colores estén invertidos.
- RF10. Según el espacio de color en el que esté la imagen, se generarán las bandas correspondientes y se crearán nuevas imágenes en escalas de gris, indicando a qué banda pertenece cada una.
- RF11. Se cambiará el espacio de color en el que está la imagen, generando otra imagen distinta, indicando el nuevo espacio de color.
- RF12. La imagen que se esté procesando se rotará respecto al centro de ésta, siendo el deslizador el que indique los grados.
- RF13. Se aumentará el zoom de la imagen.
- RF14. Se disminuirá el zoom de la imagen.
- RF15. Se tintará la imagen con una potencia de 0.5 con el color que haya seleccionado.
- RF16. Se modificará el histograma de la imagen para hacer más fácil el reconocimiento de los elementos que en ella figuran.
- RF17. Se aplicará el filtro sepia a la imagen, que dará lugar a una imagen con un aspecto antiguo.

- RF18. Se generará una imagen con sus elementos en blanco o negro según el valor que se le dé al umbral, que está definido por el valor del deslizador.
- RF19. Se aplicará la función $f(x)$ sobre la imagen.
- RF20. Este filtro dará lugar a una imagen donde los pixeles que poseen un valor rojo mayor que 128 se quedarán igual, mientras que los demás cambiarán a escala de gris. En este caso, el filtro termal se aplica sobre el color **rojo**.
- RF21. Se generará una imagen que imite una radiografía, es decir, los colores se cambian, dependiendo de los valores definidos por los umbrales.
- RF22. Se superpondrá una imagen sobre otra, generando una nueva que tendrá como título la suma de los nombres de las imágenes originales.
- RF23. Se generará una nueva imagen donde se restarán dos imágenes.
- RF24. Similar al RF15, pero en este caso, la potencia del tintado vendrá definida por el valor del deslizador.

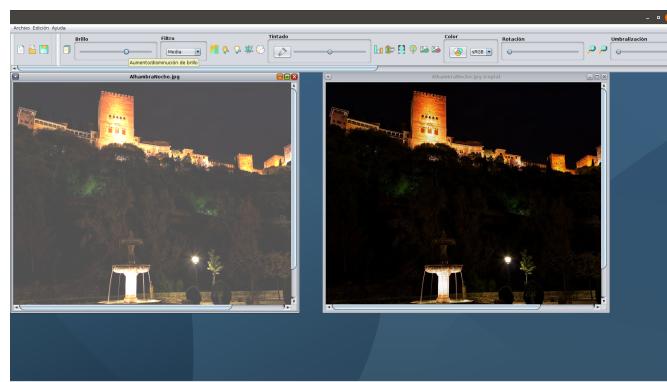
3.3. Codificación

En esta parte me centraré especialmente en las operaciones nuevas (no aplicadas en alguna práctica), las que han necesitado de la creación de nuevas clases, o en las propias, referentes a los requisitos RF9, RF17, RF19, RF20, RF21 y RF24. Toda la documentación para estos requisitos está disponible con *javadoc*. A pesar de ello, pondré ejemplos del funcionamiento de todas las operaciones (a excepción del duplicado, que mostraré su uso cuando aplique otras operaciones, por lo que no habrá una sección dedicada exclusivamente a esto).

La explicación de cada operación se hará en el mismo orden en el que se definieron los requisitos funcionales.

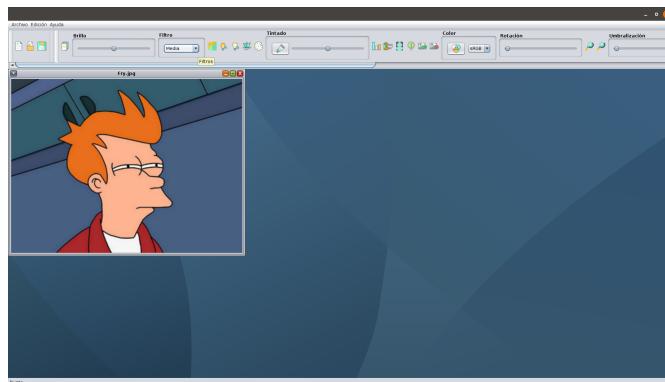
3.3.1. Brillo

El brillo de todos los pixeles aumenta, aclarando la imagen.



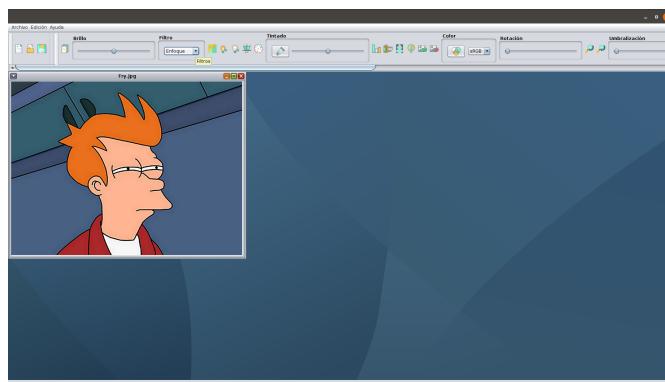
3.3.2. Emborronamiento

En este caso, se aplica el filtro media 3x3 que emborra la imagen de manera suave.



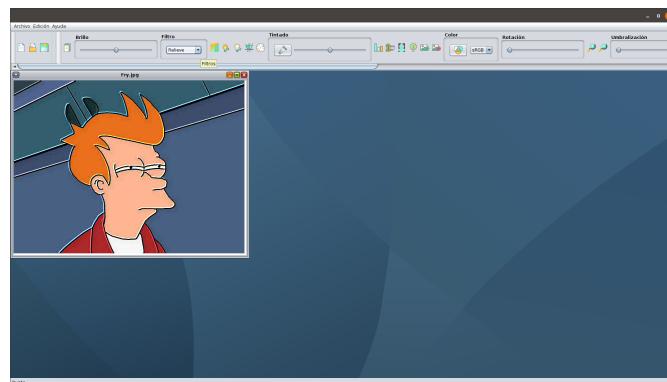
3.3.3. Enfoque

Al contrario que el emborronamiento, este filtro permite definir mejor los bordes de la imagen, haciendo que ésta se vea más nítida.



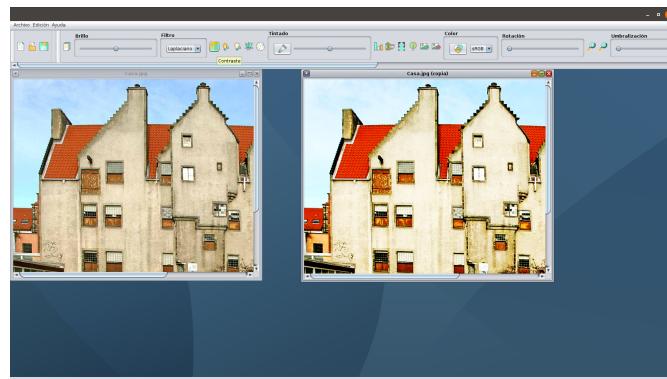
3.3.4. Relieve

En este caso, los bordes presentes en la imagen se verán más definidos, dando la impresión de que la imagen está en 3D.



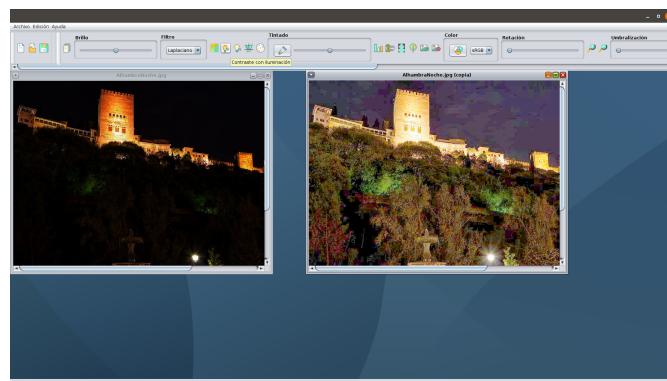
3.3.5. Contraste

Los pixeles más oscuros acentúan su oscurecimiento y los más claros, se aclaran más.



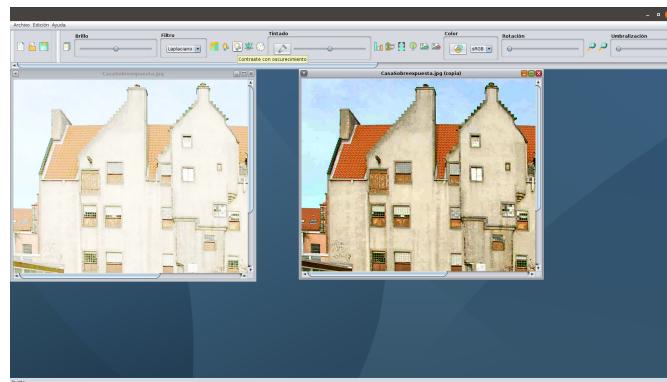
3.3.6. Contraste con iluminación

Las partes oscuras se aclaran, las claras se mantienen igual, dando un aspecto de mayor luminosidad. Filtro adecuado para imágenes oscuras.



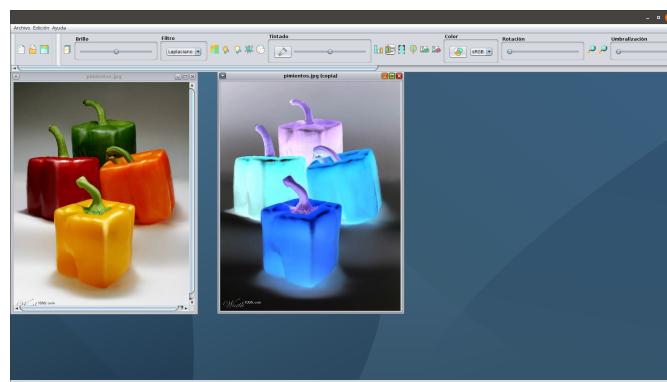
3.3.7. Contraste con oscurecimiento

A la inversa que el contraste con iluminación; las partes claras se oscurecen y las oscuras se mantienen. Adecuado para imágenes sobreexpuestas.



3.3.8. Negativo

Los colores de la imagen se invierten. En este caso, he utilizado la función ya implementada en el paquete SM.IMAGE, de forma que solo he hecho la aplicación de dicha función, no la manera de tratar los pixeles de la imagen para conseguir el efecto.



3.3.9. Extracción de bandas y espacios de color

Se muestran dos funcionalidades en una: se convierten las imágenes a distintos espacios de color y, si es posible, se extraen sus bandas:

- Conversión a RGB y extracción de bandas.



- Conversión a YCC y extracción de bandas.

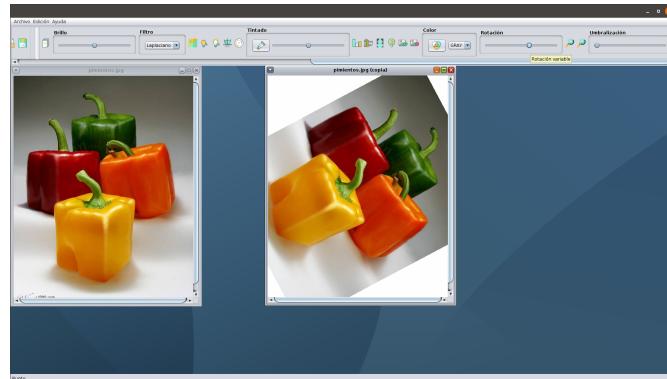


- Conversión a GRAY.



3.3.10. Rotación

La imagen rota mediante un deslizador en 360°.



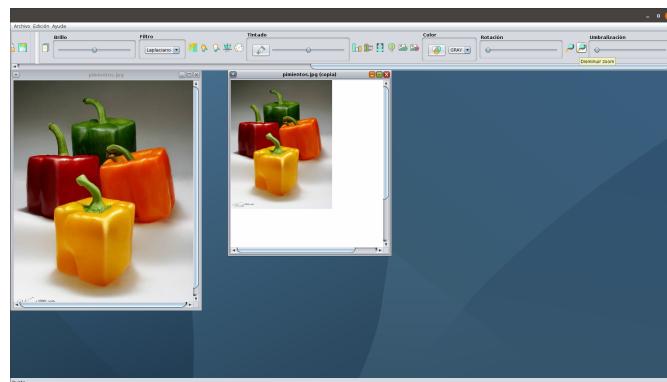
3.3.11. Zoom-in

Se aumenta el tamaño de la imagen pero sin producir un recorte para ajustarse a la ventana. La imagen se mantiene íntegra.



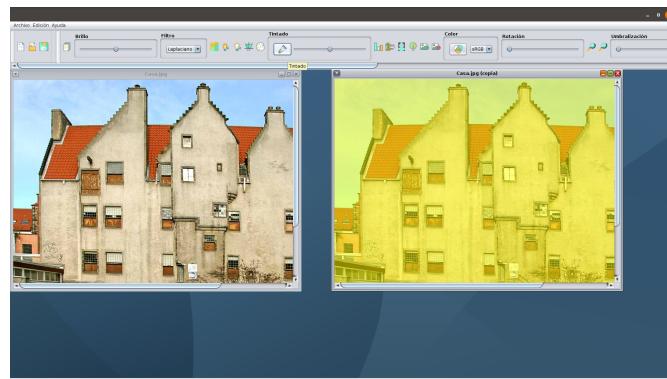
3.3.12. Zoom-out

Se disminuye la imagen pero sin añadir marco o borde a la imagen original.



3.3.13. Tintado (mediante botón)

Se aplica el tinte del color elegido con una potencia de 0,5.



3.3.14. Ecualización

Se intenta aclarar la imagen de forma que sea más nítida y se reconozcan mejor las figuras.



3.3.15. Sepia

Filtro que da aspecto de imagen antigua. Para ello, se ha definido la clase `SEPIAOP` en el paquete `SM.NGR.IMAGE`. Es una operación pixel a pixel, puesto que hacen falta las tres componentes de la imagen para dar lugar al color final de cada pixel, siguiendo la regla:

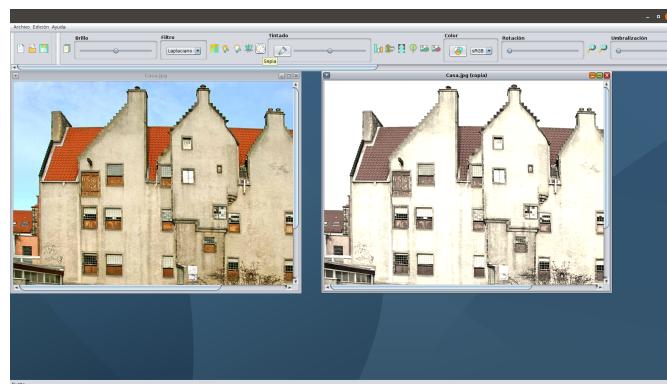
```

1  pixel_comp[0] = (int)(pixel[r]*0.393 + pixel[g]*0.769 + pixel[b]*0.189);
2  pixel_comp[1] = (int)(pixel[r]*0.349 + pixel[g]*0.686 + pixel[b]*0.168);
3  pixel_comp[2] = (int)(pixel[r]*0.272 + pixel[g]*0.534 + pixel[b]*0.131);

```

donde `R` es la componente roja del pixel, `G` la verde y `B` la azul.

Un ejemplo de funcionamiento es el siguiente:

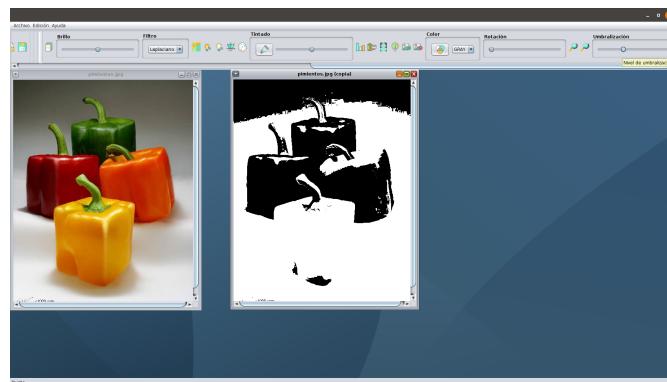


3.3.16. Umbralización

En este caso, la imagen de salida es una imagen binaria, donde solo hay blanco o negro, dependiendo del umbral que se imponga. Dicho umbral estará definido por el usuario mediante el deslizador y varía entre 0 y 255.

Para poder generar la imagen, se calcula la media de la suma de las tres componentes de cada pixel; si está por debajo del umbral, esa zona queda negra. Si está por encima, en blanco.

Un ejemplo de uso sería el siguiente:



3.3.17. Función propia

En este caso, se pide implementar una función propia. La función implementada cumple con la siguiente fórmula explícita:

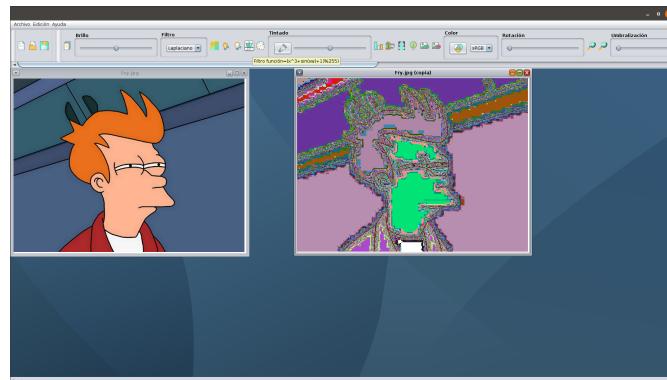
$$f(x) = |x^3 + |\cos(xw)| + 1| \% 255$$

donde w representa la velocidad angular. La representación gráfica de la función es la siguiente:



A pesar de que en el gráfico no sale, la función está definida entre 0 y 255.

El aspecto que le da a la imagen es poco convencional y difícil de entender, pues ni yo misma entiendo qué pasa con los colores que se le asignan a los pixeles. La imagen final es un tanto bizarra, pero a la vez divertida, puesto que los colores no se invierten exactamente pero distan mucho de los originales. Un ejemplo de uso es este:



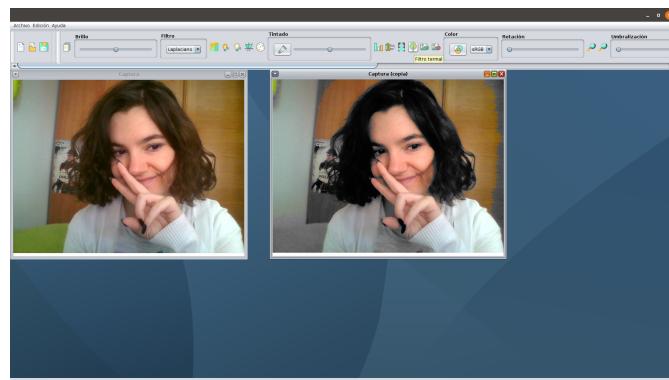
3.3.18. Termal

El filtro crea algo similar a un filtro de fronteras pero que trata los colores, es decir, en este caso, busca aquellos pixeles con tonos de rojo intenso (por encima de 128 si hablamos entre 0 y 255) y los deja igual, mientras que los demás los pone en escala de gris. De esta forma, se resaltan las partes rojas entre lo demás (por eso lo he llamado termal, ya que el rojo se asigna siempre a un color cálido).

Para ello, se ha definido una clase propia, `TermalOp`, que calcula el valor de cada pixel. Es una operación componente a componente, ya que el valor de cada componente del pixel no depende de los demás.

Un ejemplo de uso es el siguiente (la primera es una imagen normal, la segunda una tomada con la webcam):





Como se puede comprobar, las partes con más rojas o anaranjadas son las que se mantienen mientras que el resto pasan a escala de gris.

3.3.19. Rayos X

La idea de este filtro es conseguir que las imágenes tengan los colores de una radiografía como la siguiente:



Para ello, se ha creado una clase `RayosXOp`, que aplica esta operación pixel a pixel; el valor final de un pixel se calcula como la media de los tres componentes: si está por debajo del umbral inferior, es decir, si es una parte oscura, se destacará igual que ocurre con los huesos en la radiografía; si está entre el umbral inferior y el superior, tendrá una tonalidad intermedia, como se ven los músculos y piel en las radiografías; y si está por encima del umbral superior, esa zona quedará oscura, ya que no es lo que nos interesa y se tomará como información despreciable (como ocurren con el fondo de la sala en las radiografías).

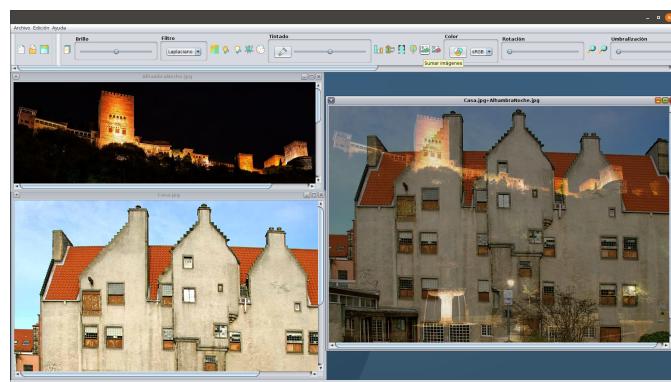
Un ejemplo de su uso con las mismas imágenes que en el filtro termal es el siguiente:



3.3.20. Suma de imágenes

En este caso, se unen dos imágenes en una, superponiendo la segunda encima de la primera, con un grado de transparencia de 0,5. La imagen generada es una nueva, que conserva las dimensiones de la primera imagen. Para poder aplicarlo, se ha utilizado la operación BLENDOP del paquete SM.IMAGE.

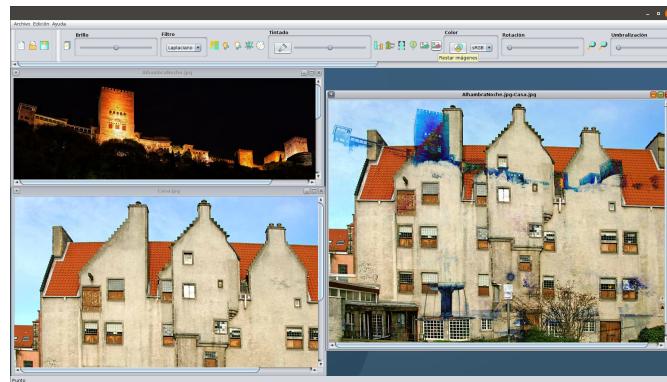
Un ejemplo de uso es el siguiente:



3.3.21. Resta de imágenes

Al contrario que en el caso anterior, en la resta la idea es de nuevo superponer dos imágenes, pero estando en negativo aquella que se superpone. Es decir, las partes negras pasarían a blanco al superponerla. Para que el efecto se muestre, se ha usado SUBTRACTIONOP, también del paquete SM.IMAGE.

Un ejemplo de uso es el que sigue:

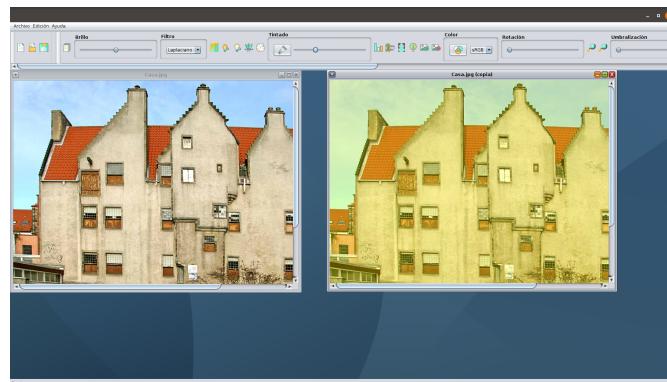


3.3.22. Tintado (mediante deslizador)

Por último, la operación de tinte aplicada al botón se aplicará también con un deslizador. La diferencia fundamental es que, en este caso, la potencia que se le da no es de 0,5 sino que es el usuario el que la elige mediante el deslizador, que va de 0 a 1, donde 0 significa que no hay tintado y 1 significa que se ha tintado la imagen completamente.

Para ello, se utilizará el color seleccionado, en este ejemplo (al igual que en el del botón), se ha usado el color amarillo.

Un ejemplo de su uso, en el que se muestra la diferencia con la imagen utilizada en la explicación del tinte con botón es la siguiente:



En este caso, al poder elegir, la imagen se ve mucho menos amarilla que en el caso del botón, ya que la potencia no supera el 30 %.

4. Sonido

4.1. Análisis de requisitos

Respecto a la parte de sonido, el sistema cuenta con los siguientes requisitos:

- RF1. Reproducir sonido.
- RF2. Grabar sonido.
- RF3. Incluir un temporizador de grabación.
- RF4. Mantener disponibles los archivos que se abran o graben en una lista.

4.2. Explicación de cada requisito

La explicación para cada requisito de sonido es la siguiente:

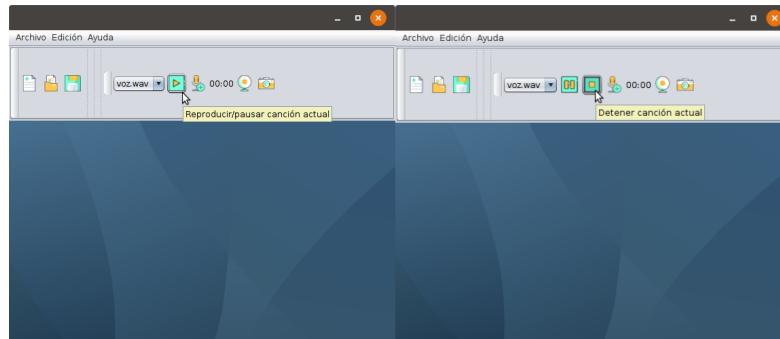
- RF1. El sistema reproducirá los sonidos en los formatos .AU y .WAV. Para ello, se incluirá un botón de reproducción que empezará a reproducir el audio cuando se pulse; también incluirá un botón de stop que parará la canción y la reiniciará, de forma que al volver a pulsar el botón de reproducción, comenzará desde el principio. Adicionalmente, se añadirá un botón de pausa que detenga la canción pero no la rebobine, es decir, que cuando se vuelva a reproducir, lo haga desde donde se quedó.
- RF2. Se añadirá un grabador de voz que utilice el micrófono disponible del dispositivo para grabar sonido. Para ello, se incluirá un botón de inicio de grabación, y un botón de parada. Cuando se pulse éste último, la grabación se detendrá y se abrirá un diálogo para guardar el archivo. Dicha grabación se añadirá a la lista de archivos reproducibles.
- RF3. Al lado de los botones de grabación de sonido, habrá un temporizador que indicará cuánto tiempo está durando la grabación.
- RF4. Habrá una lista que incluya todos los archivos disponibles, ya sea porque se han abierto o porque se han grabado. Seleccionando uno de los incluidos en la lista, se podrá reproducir o parar como un archivo más.

4.3. Codificación

Aquí explicaré brevemente el funcionamiento de cada requisito con imágenes de la aplicación.

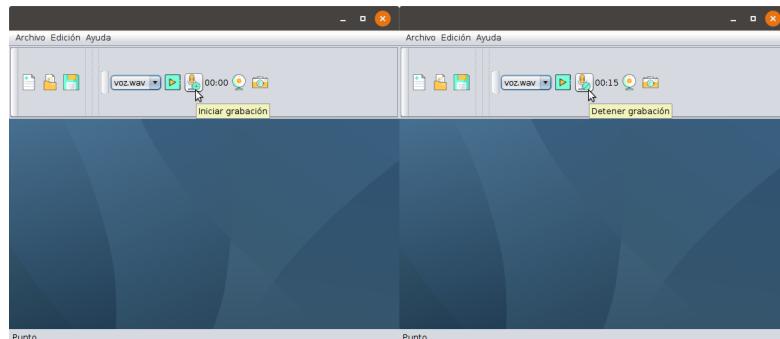
4.3.1. Reproducción

Como se ha explicado previamente, se empezará a reproducir (o se detendrá) cuando se pulse el componente diseñado para este fin.

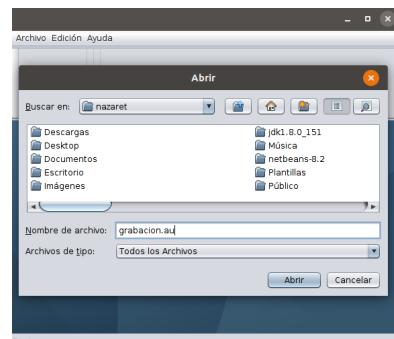


4.3.2. Grabación y temporizador

Referente a los requisitos RF2 y RF3, se grabará y detendrá la grabación tal y como se muestra en la imagen. Como se puede observar, mientras se está grabando, el temporizador marca los minutos y segundos que se han grabado.

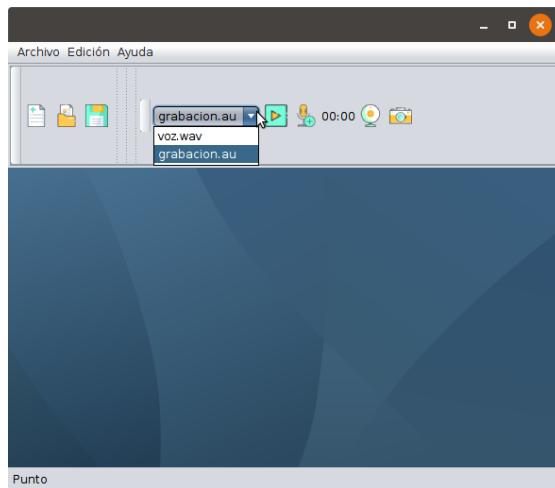


Una vez detenida la grabación, se abrirá un diálogo para guardar el archivo.



4.3.3. Lista de archivos

Por último, se mostrará una lista (en este caso un desplegable) con los archivos que se han abierto o que se han grabado. Se podrá seleccionar el que se deseé para reproducirlo.



La lista se hará tan larga como archivos se hayan añadido.

5. Vídeo

5.1. Análisis de requisitos

Por último, los requisitos respecto a vídeo que pide la evaluación son los que siguen:

- RF1. Reproducir pistas de vídeo.
- RF2. Apertura de la webcam.
- RF3. Captura de imágenes de la webcam o de vídeo.

5.2. Explicación de cada requisito

- RF1. Se abrirán pistas de audio en todos los formatos soportados por VLC¹.
- RF2. Se podrá activar la cámara conectada al dispositivo y ver en tiempo real lo que sucede.
- RF3. Con un botón se podrá tomar captura del vídeo (o de la cámara, según la ventana que haya seleccionada en ese momento) y se podrá procesar la imagen como si fuera una imagen cualquiera. Todas las operaciones disponibles para el resto de imágenes lo estarán para cada captura que tomemos.

5.3. Codificación

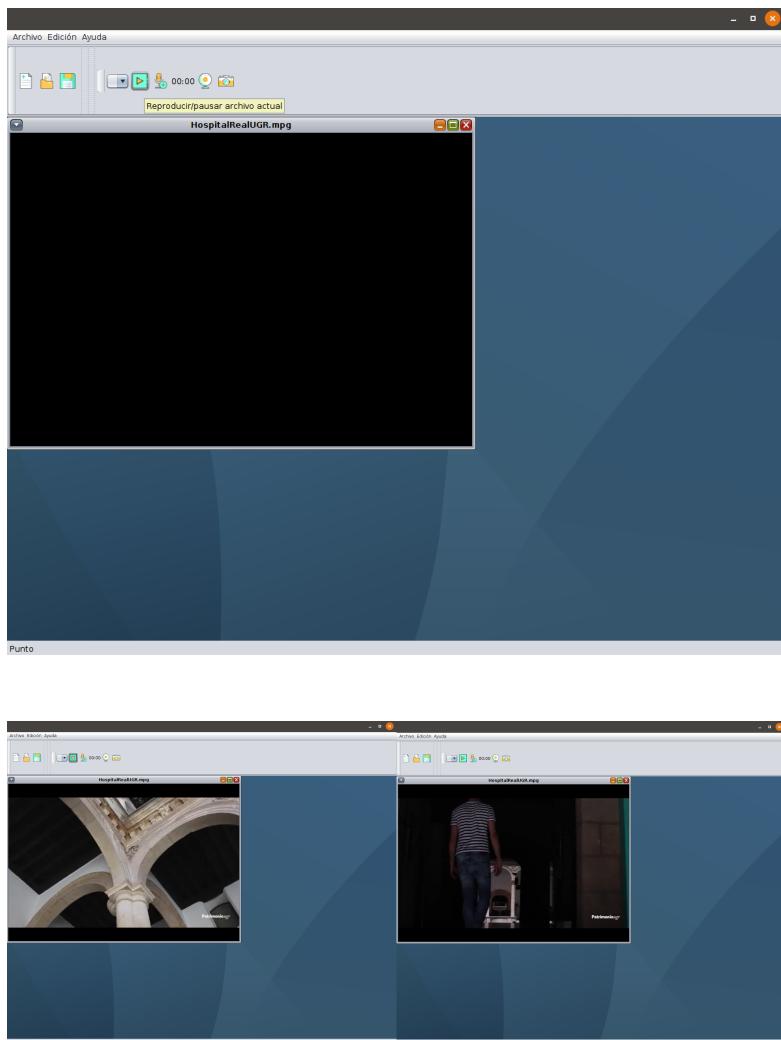
En este apartado explicaré el funcionamiento mediante capturas y parte de la implementación que he llevado a cabo.

5.3.1. Reproducción

Para poder reproducir vídeo, ha sido necesario crear un nuevo tipo de ventana interna al panel, que se dedique solo a reproducir vídeo y donde no se puedan llevar a cabo acciones como dibujar o procesar imágenes.

Este tipo de ventana solo se utilizará para esto, y consta de un área de visualización dada por el propio gestor de vídeo que se usa, perteneciente a la librería VLCJ.

Visualmente, la reproducción será como la que sigue: se abrirá una ventana que mostrará el vídeo (primera imagen), y habrá que darle a reproducir cuando se quiera comenzar a verlo y a parar cuando se desee dejar de reproducirlo (segunda imagen).



5.3.2. Uso de la cámara y capturas

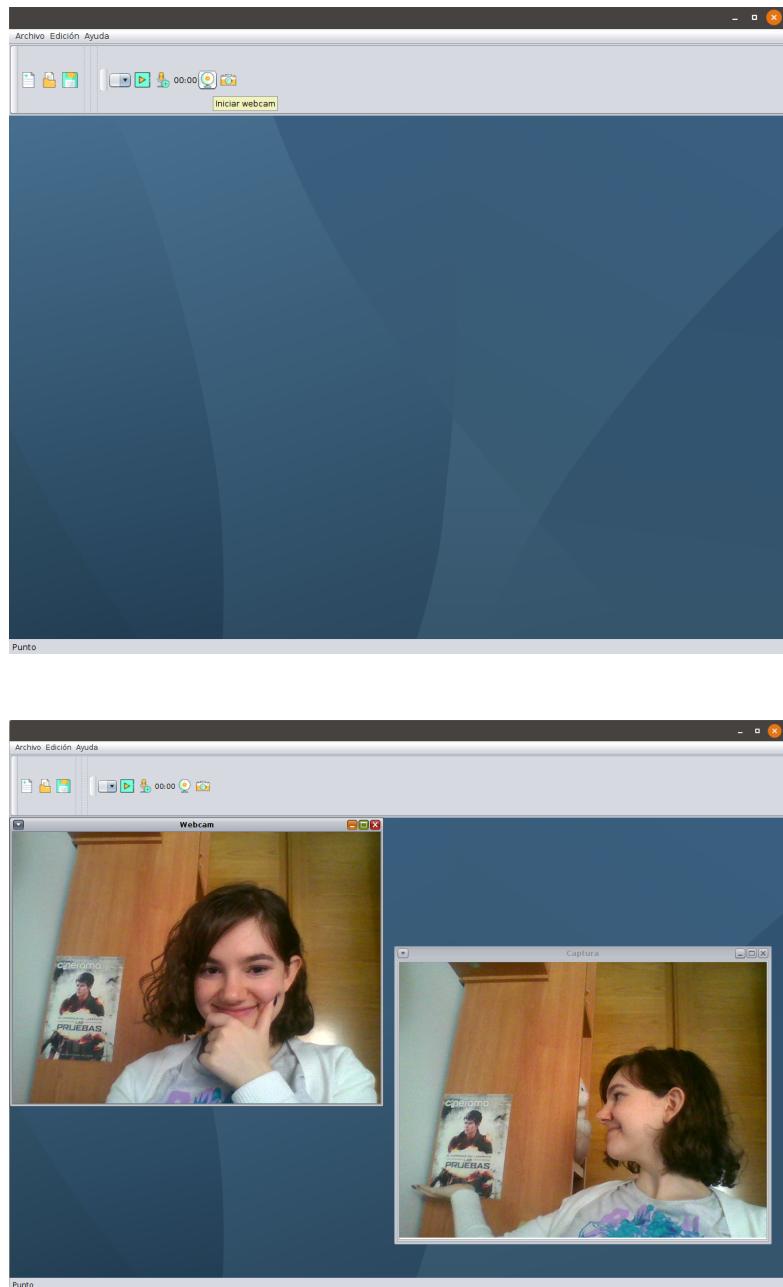
De nuevo, al igual que en el caso anterior hace falta una ventana específica para la cámara del dispositivo, ya que en ese caso, además de restringirse el dibujado de formas o el procesado de imágenes, tampoco se puede parar y reanudar un archivo, ya que lo que capta la cámara es en tiempo real.

Para ello, se crea una ventana interna en cuyo interior se encuentra el área visual donde se mostrará la secuencia que capte el dispositivo.

¹Imprescindible tener instalado el reproductor y las librerías de VLC puesto que sin ellas, la parte de vídeo no funciona.

Una vez se visualice el contenido, se podrán tomar tantas capturas como se deseen, que se guardarán en una ventana interna donde, esta vez sí, se podrá dibuja o procesar la imagen con los distintos filtros.

Su funcionamiento es el siguiente:



6. Bibliografía y fuentes de código

- Transparencias de teoría y prácticas de la asignatura Sistemas Multimedia.
- <https://docs.oracle.com/javase/7/docs/api/java/>
- <https://stackoverflow.com/>
- <https://www.flaticon.com/>