Sistemas Multimedia: Práctica de Evaluación

Laura Gómez Garrido

Descripción del problema

Se pretende realizar una aplicación multimedia que permita gestionar de forma integral diferentes tipos de medios: gráficos, imágenes, sonido y vídeo. Sobre cada uno de dichos medios se podrán llevar a cabo diferentes tareas que, en función del medio, abarcarán desde la creación y/o captura hasta la edición, reproducción y procesamiento. Para ello, la aplicación contará con un conjunto de menús y barras de herramientas que permitirán llevar a cabo dichas tareas.

Para una descripción del problema más detallada, consultar el guión de la práctica de evaluación del curso 2018/2019.

Para una mayor legibilidad del problema, dividiremos su descripción de acuerdo a sus grandes partes (carácter general, gráficos, imágenes, sonido y video) puesto que son relativamente independientes y haremos el análisis para cada una de ellas.

Carácter general: Interfaz de Usuario.

Nuestra aplicación nos permitirá trabajar, de forma integrada, con todos los medios estudiados a lo largo del curso: gráficos, imágenes, sonido y vídeo. Para ellos tendremos un escritorio central donde podrán alojarse diferentes ventanas internas de diferentes tipos en función del medio. Así, nuestros requisitos funcionales serán:

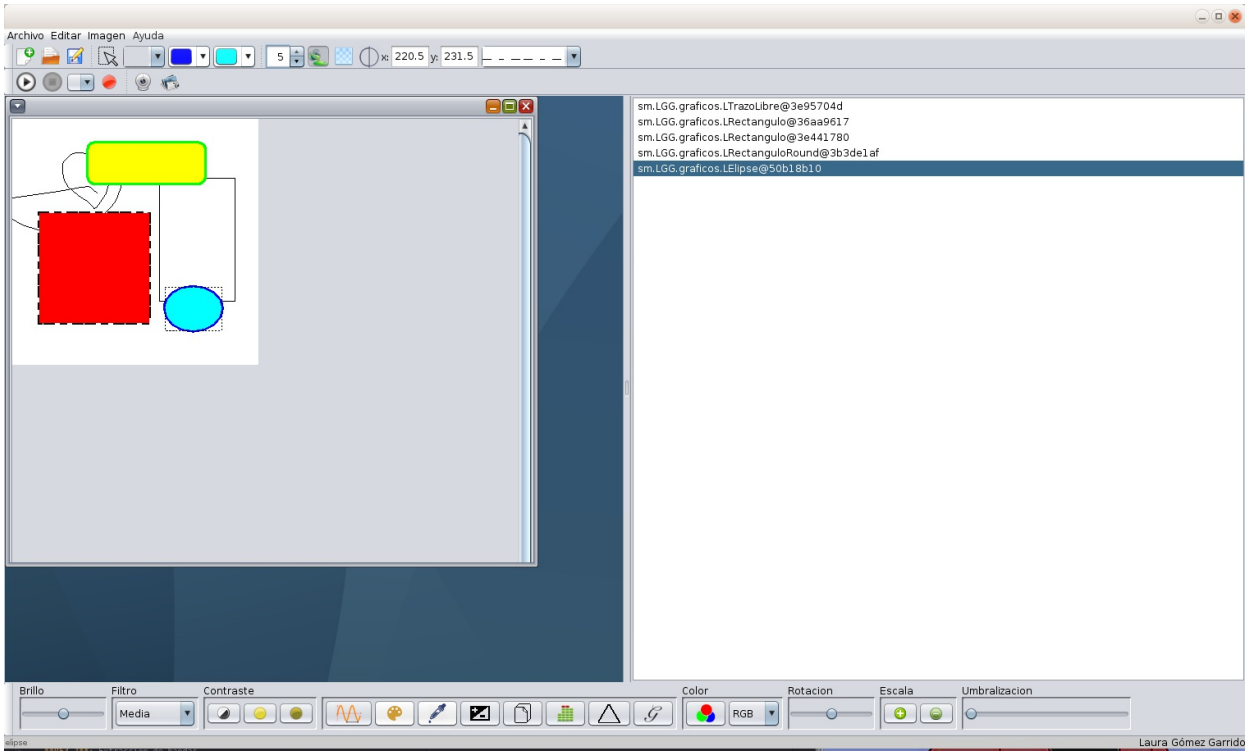
Requisitos Funcionales:

- **RF1.1**: Habrá un escritorio central donde podremos trabajar con todas nuestras funcionalidades al mismo tiempo.
- **RF1.2**: Dentro del escritorio tendremos ventanas internas con las que se trabajará de acuerdo a la funcionalidad. Pese a que en el guión recomiendan heredar de una ventana interna, al final se ha optado porque cada ventana sea independiente y puesto que ya todas

heredan de JInternalFrame.

- **RF1.3**: Deberá tener un botón "Nuevo" que permitirá crear una nueva imagen, con la posiblidad de indicar el tamaño de la imagen. Además, este botón tendrá un icono y un *ToolTipText*.
- **RF1.4**: Deberá tener un botón "Abrir" que permitirá abrir tanto un fichero de imagen, como uno de video como de sonido. Los formatos reconocidos serán los estándares manejados por Java. Si se produce un error, se lanzará un diálogo informando del problema y asociado al diálogo de abrir definiremos filtros para que sólo muestre extensiones correspondientes a ficheros de formatos admitidos.
- **RF1.5**: Deberá tener un botón "Guardar" que lanzará un diálogo que permitirá guardar la imagen de la ventana que este seleccionada, incluyendo las figuras dibujadas. De la misma forma que con "Abrir", definiremos filtros para que sólo muestre las extensiones de formatos admitidos.
- **RF1.6**: En la barra de menú, pondremos un menú "Archivo" que volverá a tener las opciones "Nuevo", "Abrir" y "Guardar" indicadas anteriormente.
- **RF1.7**: En la barra de menú, pondremos un menú "Ver" que nos permitirá ocultar o visualizar las diferentes barras de herramientas.
- **RF1.8**: En la barra de menú, pondremos un menú "Ayuda" que tendrá la opción "Acerca de", la cual os mostrará un diálogo con el nombre del programa, versión y autor.

Desarrollo



Aquí destacaremos la implementación de varias clases auxiliares en SM.LGG.iu donde muchas de ellas sirven de gran utilidad a la hora de desarrollar el entorno visual de esta aplicación,

haciendo referencia por ejemplo a la gran variedad de renders implementados o al panel de dialogo que hemos utilizado para la transparencia, por ejemplo.

Si bien, la interfaz de usuario no es todo lo minimalista que debería de ser debido en gran parte a la cantidad de elementos y al tratar de utilizar una JList junto con un split que ocupan gran parte del espacio de nuestro programa.

Gráficos

Pese a estar muy ligados los gráficos con las imágenes, ya que utilizaremos la misma ventana interna para ambos y será sobre las imágenes donde podremos dibujar nuestros gráficos, se ha decidido diferenciar los requisitos funcionales de ambos al ser relativamente independientes en su mayoría. Por ello, antes de iniciar con los requisitos funcionales de los gráficos, recalcaremos algunos detalles.

Las operaciones que realicemos sobre nuestros gráficos sólo afectarán a los gráficos que más adelante definiremos y de la misma forma las operaciones de las imágenes sólo afectarán a las imágenes y no lo a los gráficos. Un ejemplo concreto donde se podría ver esta funcionalidad es, por ejemplo, al rotar una imagen. Si tenemos una imagen con un rectángulo dibujado en ella y la rotamos, la imagen se rotaría pero el rectángulo se mantendría en la posición original. Sin embargo, si guardamos la imagen con el rectángulo y la abrimos, si a continuación la rotásemos el rectángulo se rotaría junto a la imagen puesto que ya no sería más un gráfico sino que ahora formaría parte de la propia imagen.

Requisitos Funcionales:

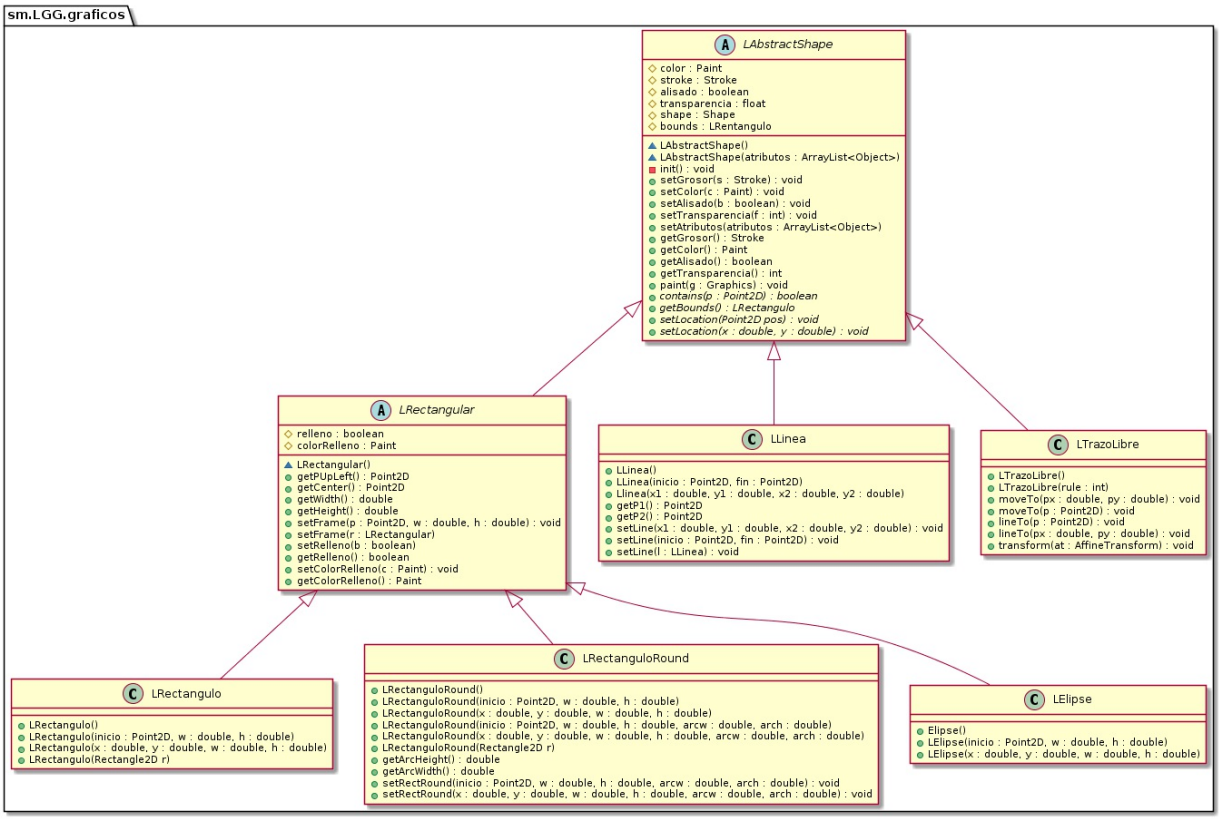
- **RF2.1**: La aplicación permitirá dibujar diferentes formas geométricas(lineas, rectángulos, etc.) con diferentes atributos sobre una imagen. Para ello, se incorporará una barra de herramientas que dé acceso a todos los elementos necesarios para dibujar con sus correspondientes *ToolTipText* asociados.
- **RF2.2**: Nuestra ventana interna tendrá en su interior un lienzo, que será donde dibujaremos nuestros gráficos. El lienzo mantendrá todas las figuras que se vayan dibujando.
- **RF2.3**: Cada figura tendrá sus propios atributos, independientes del resto de formas. A la hora de crearlas, se crearán con los valores del lienzo pero, es importante remarcar, que se dibujarán con los valores de la figura independientemente de qué atributos tenga activos el lienzo en ese momento.
- **RF2.4**: Deberá existir un panel en el que se muestre la lista de figuras que hay en el lienzo activo, la cual se actualizará cada vez que cambiemos de ventana o añadamos una nueva figura al lienzo activo.
- **RF2.5**: El usuario podrá seleccionar cualquier figura a través del panel de figuras. La figura

seleccionada deberá identificarse mediante un rectángulo discontinuo a su alrededor.

- **RF2.6**: Al seleccionar cualquier figura, deberá activarse sus propiedades en la barra de dibujo así como deseleccionarse la forma dibujo que estuviese activa. De la misma forma, si se seleccionar una forma de dibujo deberá deseleccionarse la figura seleccionada si la hubiese.
- **RF2.7**: Para la figura seleccionada, se pondrán editar sus atributos a través de la barra de herramientas.
- **RF2.8**: Se podrá cambiar la localización de la figura seleccionada, para ello aparecerá en la barra de dibujo las coordenadas en la que se encuentra localizada la figura y el usuario podrá modificar dicha información.
- **RF2.9**: De forma *OPCIONAL* se podrá cambiar la localización también mediante arrastras y soltar.
- **RF2.10**: De forma *OPCIONAL* se podrá redimensionar la figura seleccionada.
- **RF2.11**: La aplicación permitriá dibujar, al menos, la siguientes formas geométricas: Línea recta, Rectángulo y Elipse. Además, de forma opcional, se podrán dibujar también las siguientes figuras: Rectángulo redondeado, Arco, Curva con un punto de control, Curva con dos puntos de control, Trazo Libre, Polígono, Formas personalizadas, Texto formateado.
- **RF2.12**: El usuario deberá poder elegir los atributos con las que se pintarán las formas. Más adelante se indicarán de forma clara los diferentes atributos de acuerdo a qué figura nos estemos refiriendo.

Clases propias

El primer problema que debemos abordar respecto a los gráficos se trata de definir un conjunto de figuras que puedan ser dibujadas cada una con sus propiedades. Para ello, hemos utilizado el siguiente diseño de clases:



De esta forma, tenemos una clase abstracta *LAbstractShape* que contiene todos los elementos que, entendemos en este caso, que debe de tener como mínimo un gráfico. Así, definimos todos los métodos que podemos definir en caso genérico e indicamos como métodos abstractos todos los métodos que deberán tener nuestros gráficos y a los cuales no podamos darle una definición genérica.

Para facilitar nuestro diseño de clases, nos aprovechamos de las *Shape* ya definidas por Java, siendo que cada figura tendrá su equivalente *Shape* como una variable más.

Finalmente, para cada nueva gráfico que queramos definir únicamente tendríamos que heredar de esta clase abstracta.

Sin embargo, entre todas nuestras clases nos damos cuenta de que podemos realizar un agrupamiento más y este es el de las figuras rectangulares. Para ello, definimos la clase abstracta *LRectangular* de la que colgarán todas las clases que se puedan, por decirlo de algún modo, encuadrar dentro de un rectángulo. Realmente, lo que hemos hecho ha sido que las figuras que heredan de *RectangularShape* tengan su gráfico heredando de nuestro *LRectangular*.

Además, nos damos cuenta de que estas figuras tienen en común que se pueden rellenar de acuerdo a algunos parámetros. Por ello, definimos nuevas variables que nos permitan configurar esta nueva propiedad.

Imágenes

Recordamos el hecho de que pese a estar muy ligados tanto los gráficos como las imágenes al aplicarse en la misma ventana interna y poder ser dibujados los gráficos sobre las imágenes, se tratan de funcionalidades esencialmente distintas al no poder aplicarse las opciones de imágenes sobre gráficos y viceversa.

Requisitos Funcionales:

- **RF3.1**: La aplicación permitirá aplicar diferentes operaciones sobre cualquier imagen. Para ello, se incorporará una barra de herramientas que dé acceso a todos los elementos necesarios con sus correspondientes *ToolTipText* asociados.
- **RF3.2**: Duplicar una imagen.
- **RF3.3**: Modificar brillo mediante un deslizador.
- **RF3.4**: Filtros de emborronamiento, enfoque y relieve.
- **RF3.5**: Contraste normal, iluminado y oscurecido.
- **RF3.6**: Filtro de negativo.
- **RF3.7**: Extracción de bandas.
- **RF3.8**: Conversión de espacios RGB, YCC y GRAY.
- **RF3.9**: Giro libre mediante un deslizador.
- **RF3.10**: Escalado (aumentar y disminuir).
- **RF3.11**: Tintado con el color de frente seleccionado en el momento.
- **RF3.12**: Ecualización.
- **RF3.13**: Filtro sepia.
- **RF3.14**: Umbralización basada en niveles de gris con deslizador para modificar el umbral.
- **RF3.15**: Un operador "*LookupOp*" basado en una función propia.
- **RF3.16**: Una operación de diseño propio aplicada componente a componente.
- **RF3.17**: Una operación de diseño propio aplicada pixel a pixel.
- **RF3.18**: *OTROS OPERADORES OPCIONALES**

Desarrollo

Las imágenes podrán ser abiertas, editadas y guardadas en cualquier formato. No sólo eso sino que también podremos dibujar sobre ella los mismos gráficos de la sección anterior y estos se guardarán junto con la imagen. Será nuestro evento de abrir el encargado de ver si queremos abrir una imagen o cualquier otra cosa, creando una *VentanaInternaImagen* en caso de encontrarnos en esta situación.

Dicho esto, debido a la gran cantidad de filtros implementados, procederemos a explicar únicamente los propios que son los que tienen algún interés nuevo ya que todos los demás han

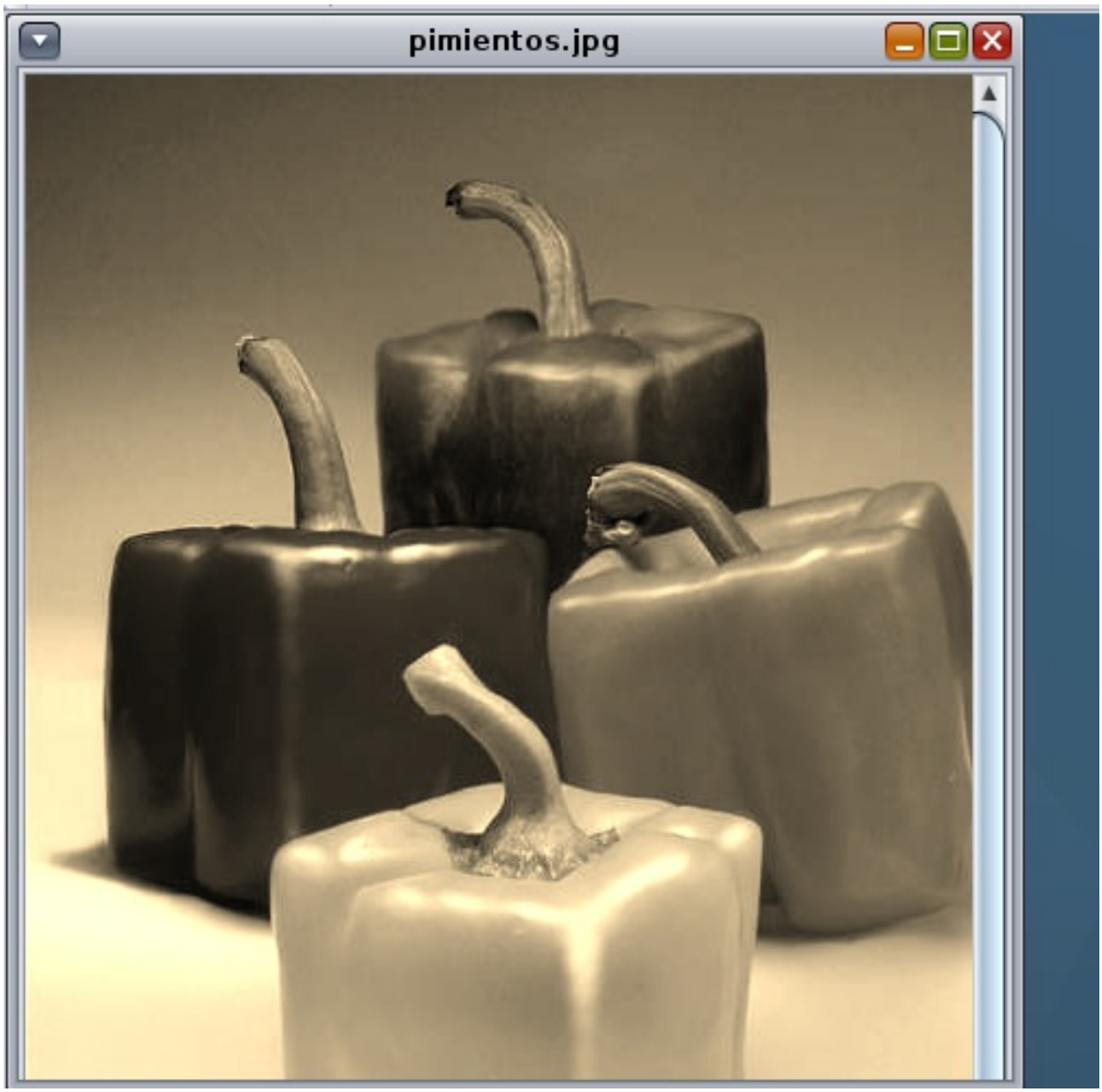
sido probados y vistos en las prácticas realizadas durante el curso y con guiones que facilitaban su implementación o con clases proporcionadas que ya directamente lo implementaban.

Umbralización:

Nos piden hacer una umbralización basada en niveles de gris. Para ello, definimos el constructor al que le pasamos un entero que representará nuestro umbral. Como nuestra imagen de salida será del mismo espacio de color y de las mismas dimensiones que nuestra imagen orginal, únicamente tendremos que sobrecargar el método filter. En este, recorremos la imagen pixel a pixel, haciendo la media de los valores de los 3 componentes y devolviendo el color original si superamos el umbral y un color negro en caso de no superarlo.

Sepia:

Aquí utilizaremos el constructor por defecto y nuevamente será necesario recargar únicamente el método filter. Recorreremos la imagen pixel a pixel y a cada componente le asignaremos los colores RGB que vienen dados en el guión de la práctica 12.

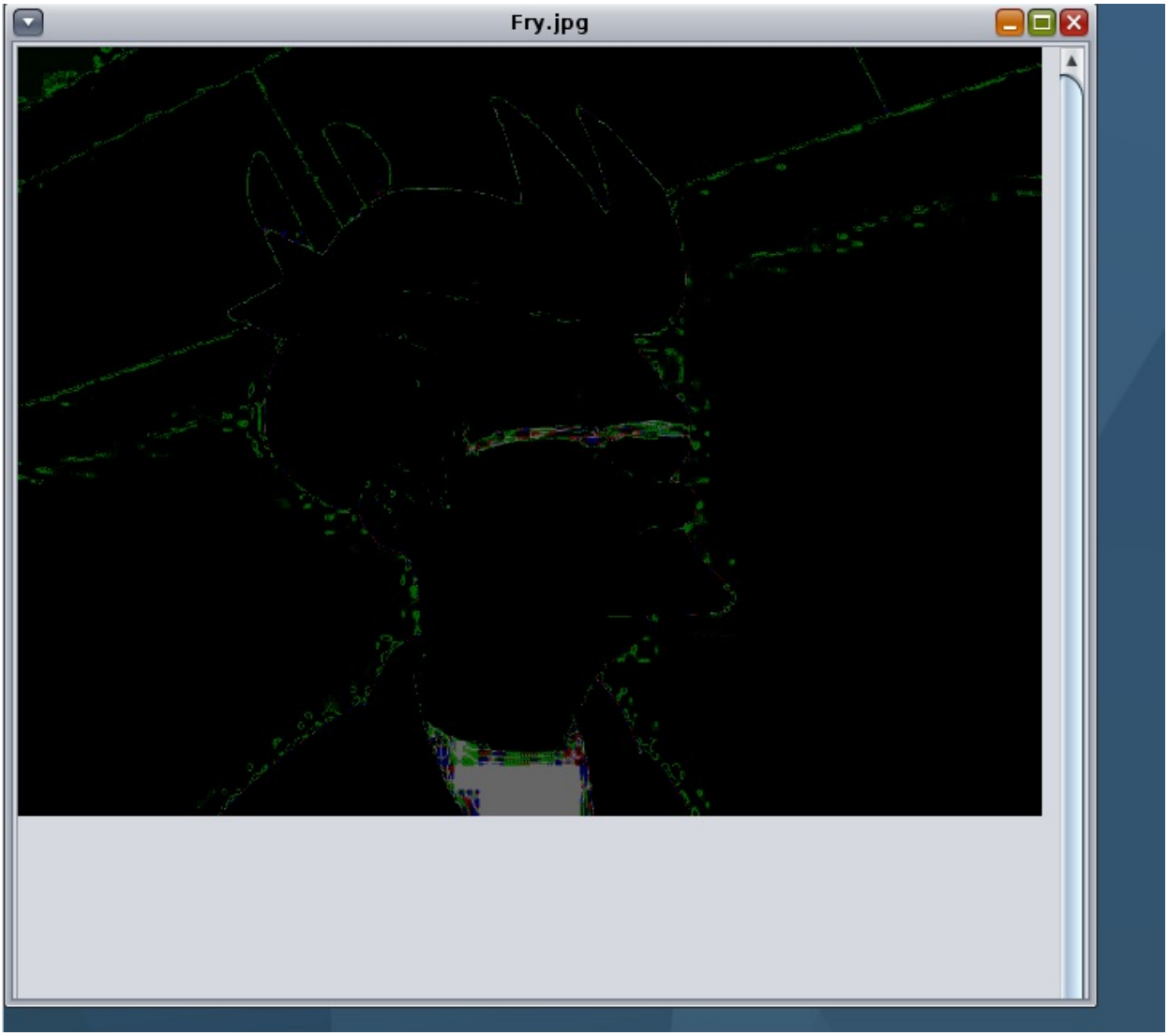


Gauss:

Se trata de una función que utiliza el constructor por defecto y en la que únicamente es necesario sobrecargar el método filter. Estamos implementado un caso particular de la función densidad de la gaussiana. Recorremos la imagen pixel a pixel, donde a cada componente le asignamos el valor de la siguiente fórmula:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Donde nuestra varianza será una constante de valor uno y nuestra media será la media del valor de los tres componentes en ese pixel.



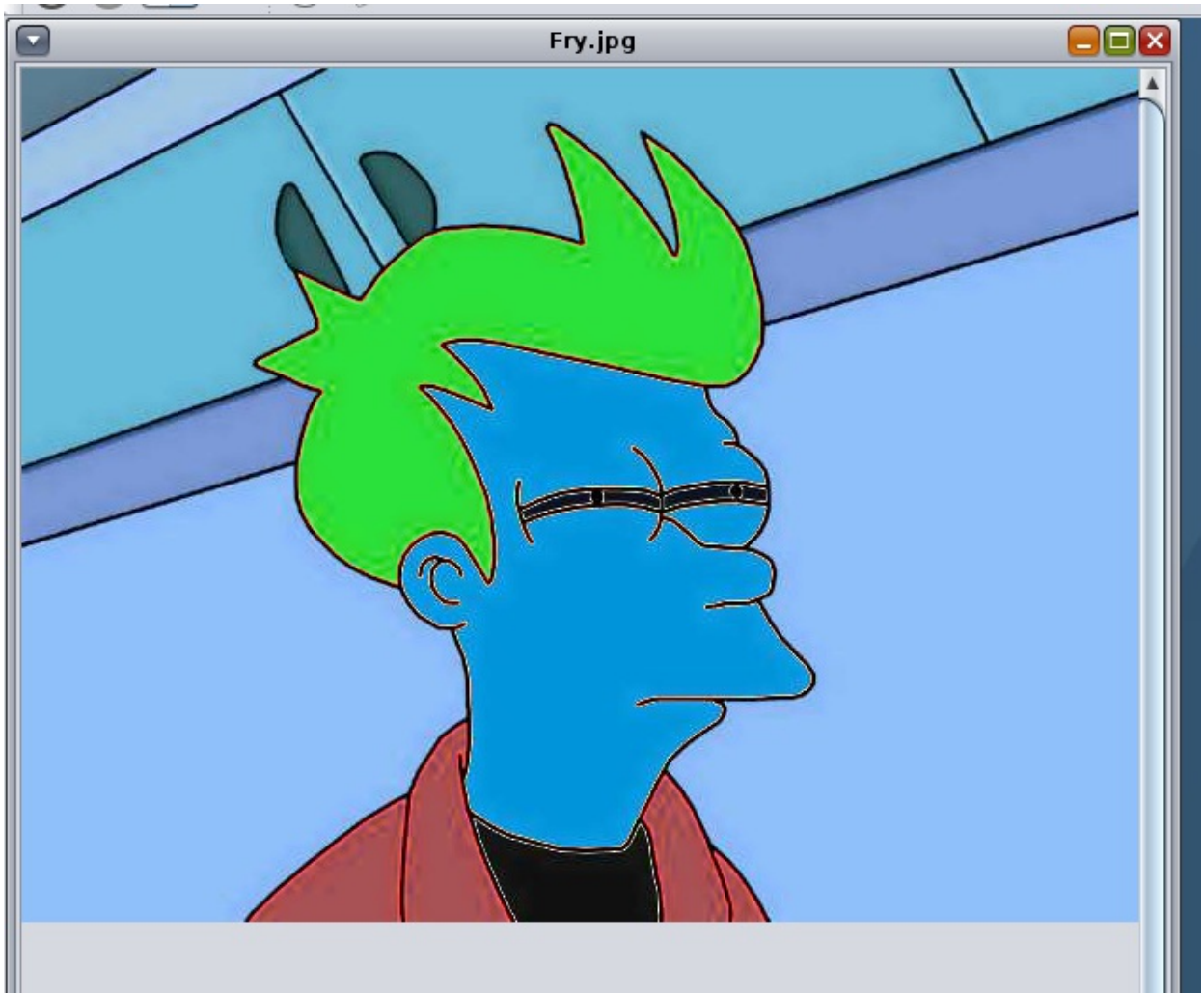
Apreciamos que este resultado es esperado debido a que hemos puesto como varianza la constante 1 y mientras que nuestros valores pueden llegar a variar mucho de la media y esto no lo tenemos en cuenta.

Triángulo:

Está función sigue la misma dinámica que todas las anteriores sólo que en este caso recorreremos la imagen componente a componente. Nuestra función elevará los valores centrales de cada componente mientras que disminuirá los valores de los extremos. De esta forma su gráfica simulará un triángulo:

$$f(x) = 2x \leftarrow x \leqslant 127$$

$$f(x) = 2(255 - x) \leftarrow x > 127$$



Sonido

Requisitos Funcionales:

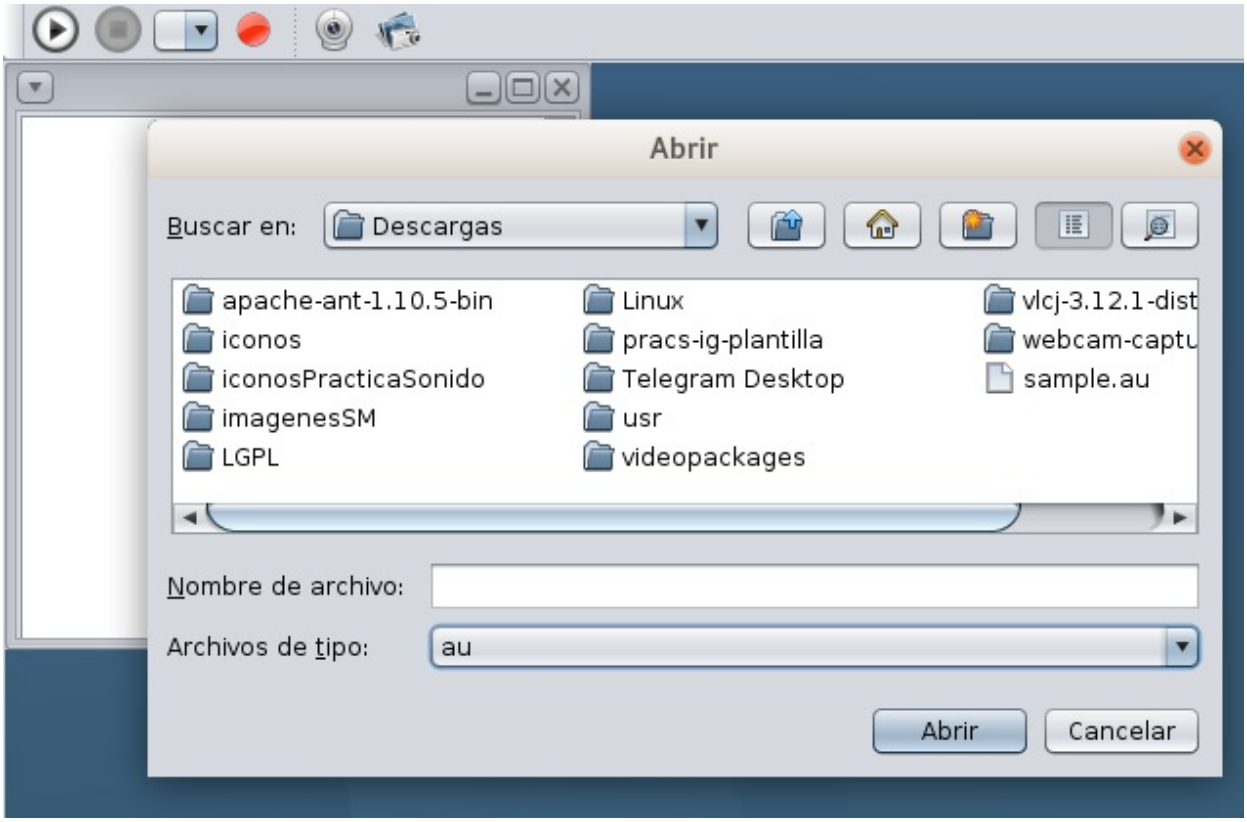
- **RF4.1**: La aplicación deberá permitir tanto reproducir como grabar sonidos. Para ello, se incorporará una barra de herramientas que dé acceso a todos los elementos necesarios con sus correspondientes *ToolTipText* asociados.
- **RF4.2**: La reproducción se aplicará sobre los formatos y códecs reconocidos por Java mientras que en la grabación se fijarán los parámetros de digitalización.
- **RF4.3**: *OPCIONAL* Permitir la selección de los parámetros de digitalización y formato de fichero al grabar.

Desarrollo

La gran parte de este apartado ha sido implementado siguiendo, fundamentalmente, el guión de la práctica 13. De esta forma, realmente no hay demasiado código propio a comentar. Aún así, procederemos a comentar algunos detalles.

A la hora de comenzar a grabar, hemos optado por el camino sencillo en el que primero indicamos dónde, con qué nombre y con qué formato de fichero comenzamos la grabación. Así, a la hora de finalizar la grabación pulsaremos el mismo botón que hemos utilizado para iniciarla que ahora tendrá otro icono. Tras finalizar la grabación, esta será añadida al combobox de sonidos y videos donde podrá reproducirse como cualquier otro.

Nótese, que al abrir un nuevo sonido podremos seleccionar el formato de sonidos que queremos abrir además de que nuestro programa detecta si es sonido, imagen o video dependiendo de qué formato tenga. De esta forma, al abrir el fichero este aparecerá en nuestro combobox y para reproducirlo únicamente tendremos que seleccionarlo y darle a reproducir.



Además, no sólo hemos implementado el play y el stop, puesto que también hemos añadido la opción de pausa. Esta aparecerá en el propio botón de play, tras haberlo pulsado y al mantener el cursos sobre este. Para esto, hemos añadido una variable de tipo *File* a nuestra *VentanaPrincipal* y junto con esto y los eventos de reproducción controlamos si queremos que nuestro botonPlayPause inicie una nueva reproducción, continúe por dónde se quedó o haga pausa. No sólo eso, sino que también con los eventos de estado de la reproducción controlamos si el botón de stop está o no habilitado.

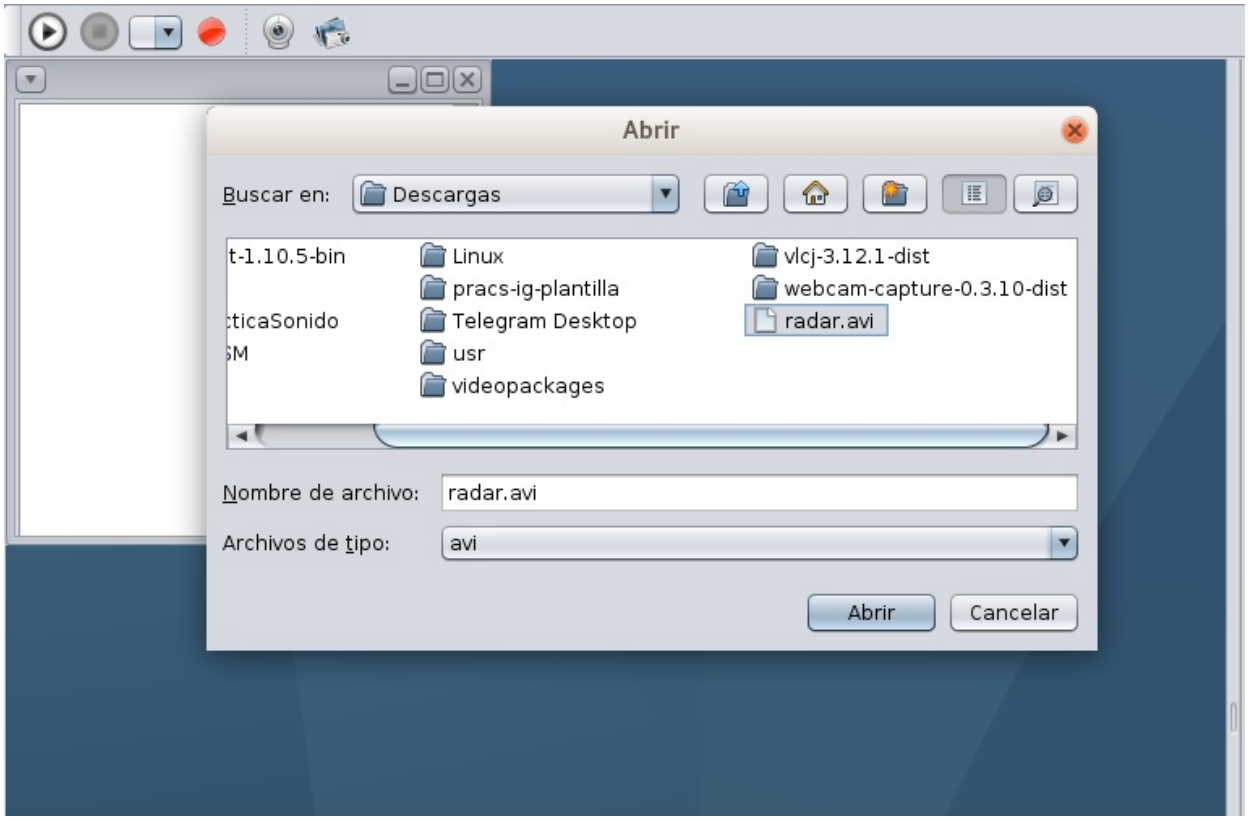
Vídeo y webcam

Requisitos Funcionales:

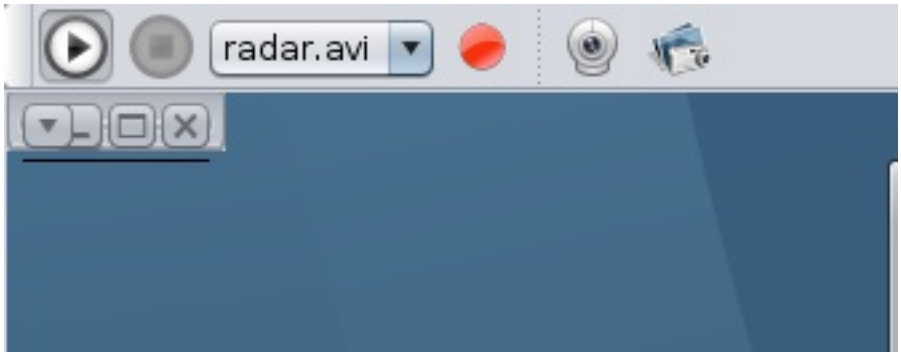
- **RF5.1**: La aplicación permitirá tanto reproducir vídeo como mostrar la secuencia que esté captando la webcam. Para ello, se incorporará una barra de herramientas que dé acceso a todos los elementos necesarios con sus correspondientes *ToolTipText* asociados.
  OPCIONAL: Algunos botones de sonido y de vídeo pueden compartir funcionalidades.
- **RF5.2**: La reproducción se aplicará sobre los formatos y códecs reconocidos por Java.
- **RF5.3**: Se permitirá al usuario capturar imágenes de la cámara o del vídeo que se esté

reproduciendo, concretamente de la ventana que este activa siempre y cuando esta sea de tipo vídeo o webcam. Esta nueva imagen se mostrará en una nueva ventana interna.

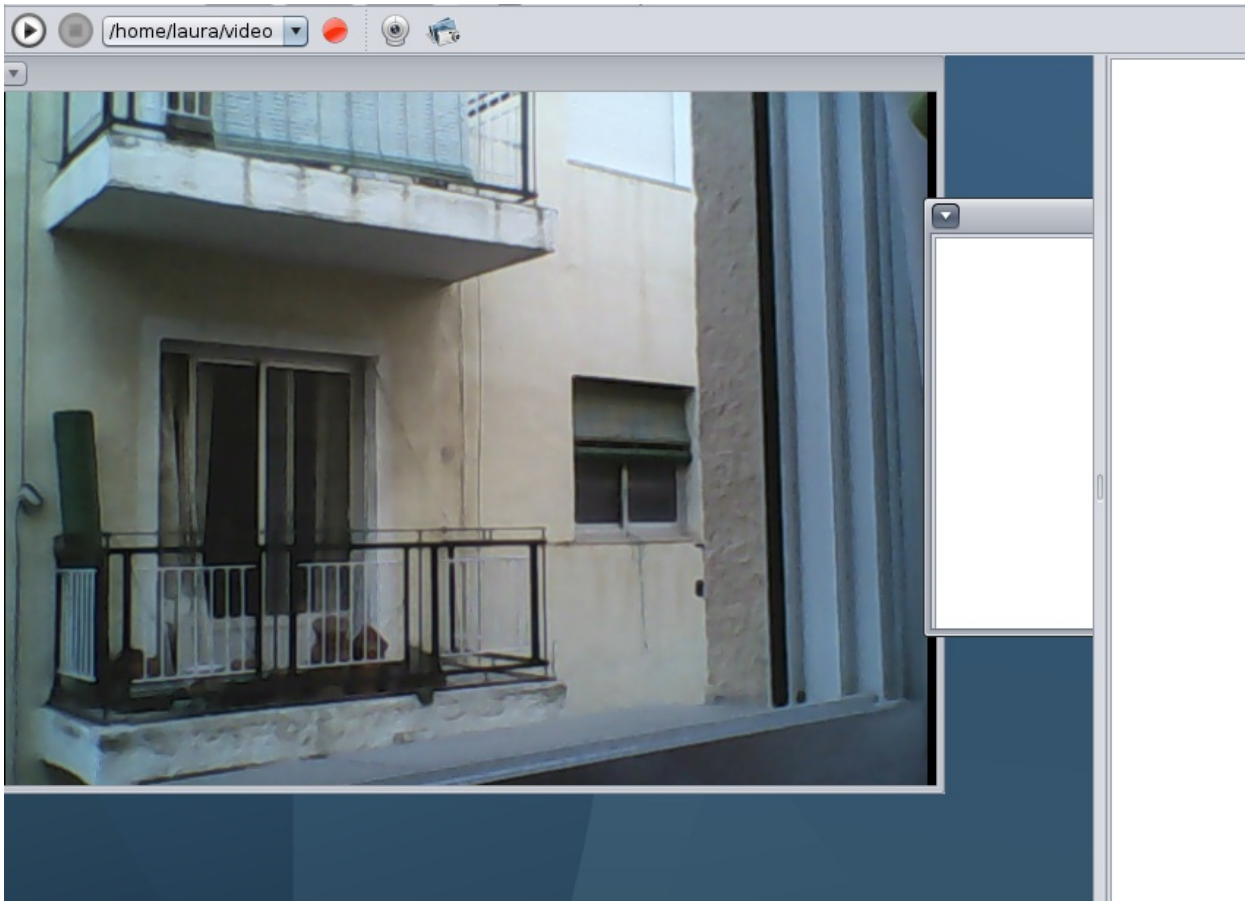
Desarrollo



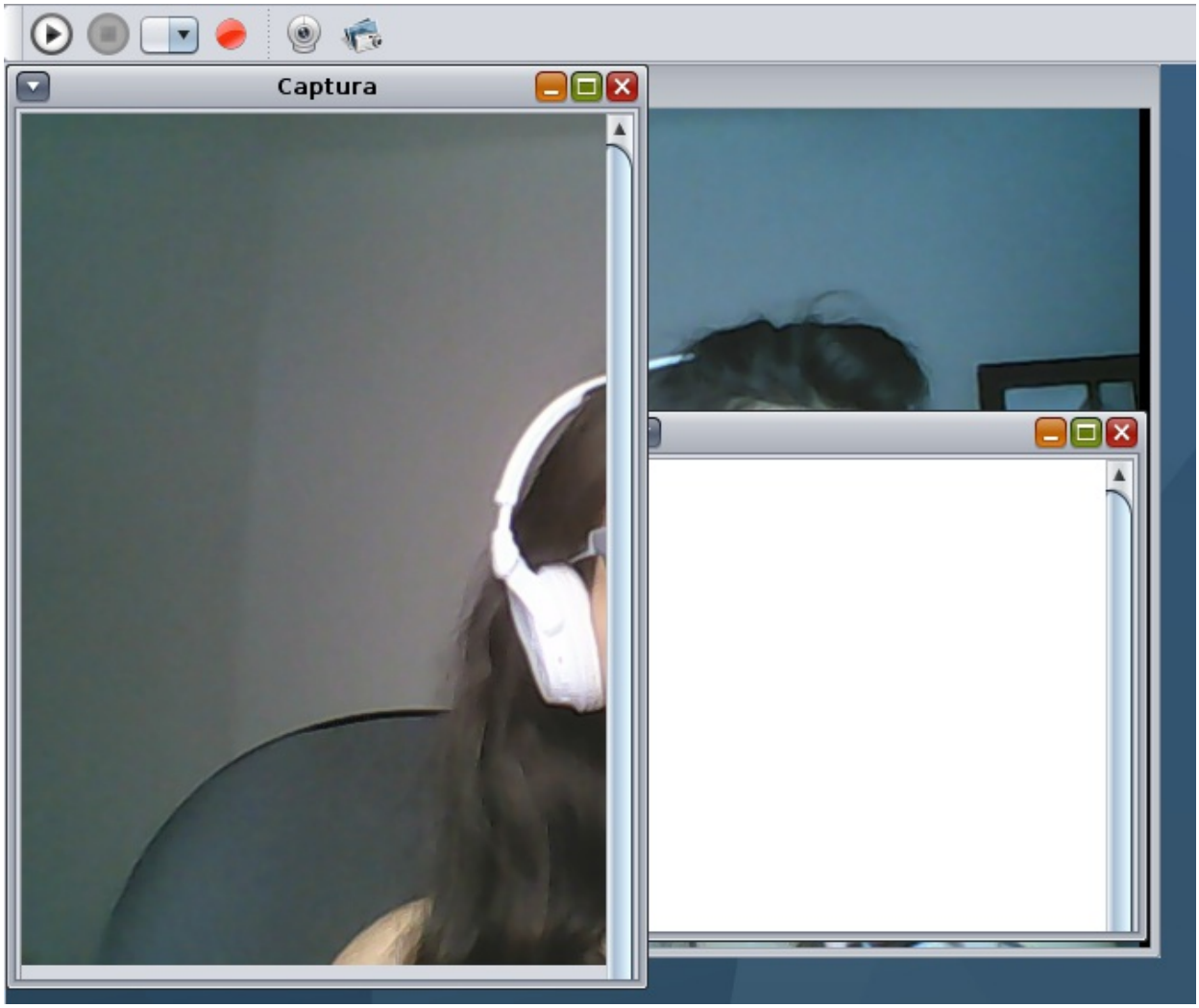
De la misma forma que con el sonido, de al utilizar el botón de abrir automáticamente nuestro programa detecta si es o no un vídeo, añadiéndolo al combobox de reproducción de sonidos y vídeos. Para poder reproducir un vídeo, habría que seleccionarse en dicho combobox y después darle un par de veces al botón que utilizamos para dar play y pause, para crear la ventana de reproducción y para que esta inicie. Sin embargo, hay que tener en cuenta que la ventana se crea muy pequeña y que puede parecer que está escondida detrás del lienzo imagen que aparece nada más iniciar el programa, deberemos de ser nosotros mismos quienes redimensionemos dicha ventana a nuestro gusto para poder ver el vídeo cómodamente.



Por otro lado, en este apartado también tenemos la opción de grabar con la webcam y de mostrar lo grabado en una ventana interna de nuestro escritorio. Para esto, pulsaremos el pequeño icono de la webcam:



Junto, con, además, la opción de hacer una captura de lo que se esté reproduciendo en la webcam y la cual será mostrada en una ventana interna de imágenes.


Captura