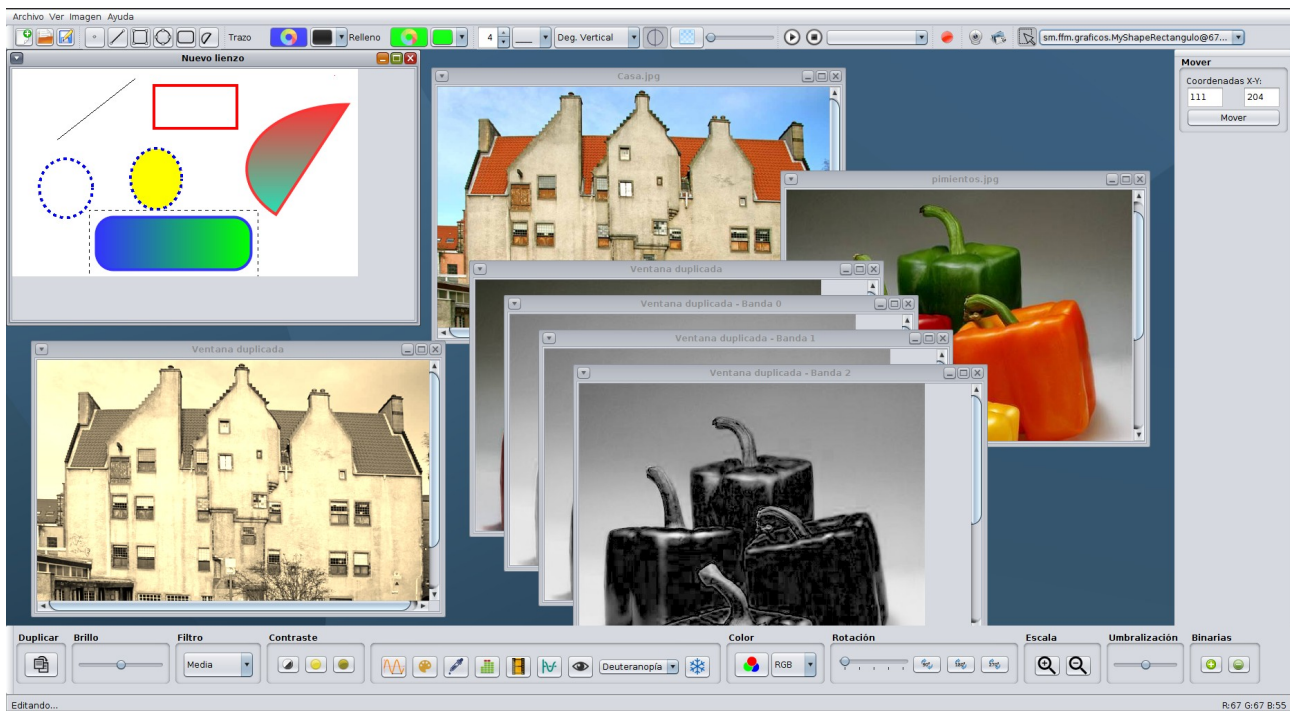


SISTEMAS MULTIMEDIA

PaintBasico2D



Autor: Francisco Fernández Millán

Asignatura: Sistemas Multimedia

Profesor: Jesús Chamorro

Fecha de entrega: 14-06-2019



ugr

Universidad
de **Granada**

1. Introducción

La documentación que se va a desarrollar a continuación, está basada en una aplicación multimedia para la asignatura Sistemas Multimedia, en ella se pretende poder gestionar diferentes tipos de medios como son: **gráficos, imágenes, sonido y video**.

Sobre cada uno de los distintos medios, se podrán realizar acciones que serán definidas más adelante. En resumen abarcarán desde la creación y/o captura hasta la edición de figuras, la reproducción de audio y video y el procesamiento de imágenes.

A continuación, detallaré los requisitos funcionales que he llevado a cabo para la realización de esta aplicación, estarán catalogados según si son requisitos de carácter general, requisitos de dibujo, requisitos de procesamiento de imágenes, requisitos de sonido y requisitos de video.

2. Análisis de requisitos

Como se ha comentado en el apartado anterior, procederé a analizar los requisitos funcionales que son necesarios en cada medio para cumplir los objetivos de esta aplicación.

- **Requisitos de carácter general**

Id. Requisito Funcional	Descripción	Solución
RF01	<i>Nuevo</i> . Permite crear una nueva imagen. El usuario deberá indicar el tamaño de la imagen.	En botón <i>Nuevo</i> tanto de la barra de herramientas como del menú <i>Archivo/Nuevo</i> , creará una nueva imagen. Para indicar el tamaño de la imagen tendremos el menú <i>Imagen/Redimensionar</i> .
RF02	<i>Abrir</i> . Abrir el dialogo “Abrir fichero” y permite seleccionar una imagen, un sonido o un video. Se usará el mismo botón para abrir cualquier medio.	En botón <i>Abrir</i> tanto de la barra de herramientas como del menú <i>Archivo/Abrir</i> , se podrá abrir el fichero deseado. Para poder abrir desde él, cualquier tipo de medio, se realizarán unas condiciones que comprueben la extensión del archivo y según sea, realizar la apertura correspondiente.
RF03	<i>Guardar</i> . Lanza el dialogo “Guardar fichero” y permite guardar la imagen de la ventana que esté seleccionada, incluyendo las figuras dibujadas.	En botón <i>Guardar</i> tanto de la barra de herramientas como del menú <i>Archivo/Guardar</i> , se podrá guardar la imagen deseada. La imagen se guardará con el formado indicado según los disponibles que se mostrarán en los filtros.

- **Requisitos de dibujo**

Id. Requisito Funcional	Descripción	Solución
RF04	Cada figura tendrá sus propios atributos.	Se crearán clases propias que representarán a la formas de dibujo. A su vez, cada una de las formas contendrán sus propios atributos de dibujo.
RF05	Edición de figuras.	A cada figura dibujada se le podrá modificar sus atributos, para ello, se modificará las propiedades de cada figura a editar y no a todas.
RF05.1	Lista desplegable que muestre las figuras dibujadas.	Se creará una lista estándar o una lista desplegable en la que se añadirán las figuras dibujadas mediante un evento propio.
RF05.2	La figura a editar estará seleccionada mediante un boundingbox.	Cuando esté activa la edición y se seleccione una figura de la lista desplegable, a dicha figura se le mostrará a su alrededor un rectángulo discontinuo que indique que es la forma que se va a modificar.
RF05.3	Activar las propiedades de cada figura en la barra de dibujo tras su selección.	Cuando seleccionemos la figura que vamos a editar, obtendremos de ella sus propiedades y activaremos/desactivaremos los atributos que posea en la barra de dibujo.
RF05.4	Editar los atributos de cada figura seleccionada.	A la figura que tengamos seleccionada le podremos modificar cualquiera de sus propiedades sin más que cambiándola en la barra de dibujo.
RF05.5	Cambiar la localización.	Se creará un panel que mostrará las coordenadas de la figura seleccionada en el desplegable. Además, se podrá cambiar su localización introduciendo otras coordenadas.
RF06	Atributo color.	Se podrá elegir tanto el color del trazo como el color del

		relleno con el que se pintarán las formas. Los colores serán mostrados tanto en una lista desplegable para los colores básicos como en un panel de diálogo para la paleta completa de colores.
RF07	Atributo trazo.	Se podrá modificar el grosor del trazo de la figura y su tipo de trazado. En un <i>spinner</i> se aumentará/reducirá el grosor y en una lista desplegable se podrá seleccionar el tipo de trazado.
RF08	Atributo relleno.	Se podrá aplicar tres opciones a la hora de rellenar, estas opciones estarán definidas en una lista desplegable. Por un lado tendremos la opción de no rellenar y por otro, el relleno liso que establecerá su color con la segunda paleta o lista de colores y los distintos tipos de degradado que automáticamente se lanzará un dialogo compuesto por la paleta de colores para la selección del color tanto para el color de frente como el de fondo.
RF09	Atributo alisado de bordes.	Se podrá activar/desactivar la mejora de renderizado correspondiente al alisado de bordes.
RF10	Atributo transparencia.	Se podrá activar/desactivar la transparencia de una forma de dibujo, además de poder establece el nivel de transparencia que se aplicará mediante un deslizador.

- Requisitos de procesamiento de imágenes

Id. Requisito Funcional	Descripción	Solución
RF11	<i>Duplicar</i> , que creará una nueva “ventana imagen” con una copia de la imagen.	El botón <i>duplicar</i> se encargará de a partir de la imagen fuente, obtener su espacio de color y su raster para posteriormente realizar una copia de ella y mostrarla en una nueva ventana.
RF12	Modificar el brillo mediante un deslizador.	A partir de un deslizador podremos modificar el brillo de la figura jugando entre los valores de [0-255] para aumentar o disminuir el brillo. También se hará uso de eventos para cuando se gane o se pierda el foco en el deslizador.
RF13	Filtros de emborronamiento, enfoque y relieve.	Tendremos una lista desplegable en la que podremos seleccionar los distintos filtros. A partir del paquete <i>sm.image</i> proporcionado por el profesor, según el filtro seleccionado se creará un <i>kernel</i> u otro.
RF14	Contraste normal, iluminado y oscurecido.	Tendremos 3 botones donde cada uno de ellos se encargará de un contraste. Según el botón seleccionado se llamará a un <i>LookupTable</i> u otro.
RF15	Negativo.	Tendremos un botón que se encargará de aplicar la operación con el <i>LookupTable</i> correspondiente al negativo.
RF16	Extracción de bandas.	Tendremos un botón que realizará la extracción de las bandas de la imagen recorriendo las bandas obtenidas a través del raster. Crearemos una nueva imagen en una nueva ventana que representará a cada banda.
RF17	Conversión a los espacios <i>RGB</i> , <i>YCC</i> y <i>GRAY</i> .	Tendremos una lista desplegable en la que se mostrará los distintos espacios de color para poder cambiarlo en la imagen.

RF18	Giro libre mediante deslizador.	Tendremos 3 botones que representarán los giros de 90, 180 y 270 grados respectivamente, además de un deslizador que va modificando los grados de la rotación.
RF19	Escalado (aumenta y disminuir).	Tendremos 2 botones que se encargarán del escalado de la imagen. El escalado se realizará según la escala, de esta forma podremos reducir o aumentar.
RF20	Tintado.	Aplicará un efecto de tintado a la imagen, se realizará mediante un método que se encuentra en el paquete sm.image y según el color del trazo seleccionado se aplicará un tintado u otro.
RF21	Ecualización.	Aplicará un efecto de ecualización que se encuentra en el paquete sm.image. Se apreciará como reparten de forma más o menos uniforme los valores del histograma.
RF22	Sepia.	Se trata de una operación componente a componente en la cual se modifica cada uno de los componentes aplicándole ciertos valores a cada componente del pixel para obtener el resultado deseado.
RF23	Umbralización.	Se trata de una operación pixel a pixel en la cual se realiza una media de las tres componentes y si está por debajo de un umbral se modifica a tonos negros y si esta por encima a tonos blancos. Un deslizador se encargará de establecer el umbral.
RF24	Operador <i>LookupOp</i> propio.	Se realizará un operador propio a partir de una función geométrica.
RF25	Operación componente a componente propia.	Se realizará una operación propia aplicada componente a componente.
RF26	Operación pixel a pixel propia.	Se realizará una operación propia aplicada pixel a pixel.

- **Requisitos de sonido**

Id. Requisito Funcional	Descripción	Solución
RF27	Reproducción de sonido.	Para la reproducción de sonido se deberá abrir el audio en cuestión y añadirse a la lista de reproducción, una vez ahí, podremos usar los botones de <i>play</i> y <i>stop</i> para reproducir y pausar respectivamente.
RF28	Grabación de sonido.	Tras pulsar el botón de <i>grabar</i> se lanzará un dialogo de guardado en el que se especificará en nombre del archivo donde posteriormente se guardará el sonido. Una vez finalizada la grabación, pulsaremos de nuevo en el botón y se guardará el contenido en el archivo creado. Además, se añadirá a la lista de reproducción.

- **Requisitos de video**

Id. Requisito Funcional	Descripción	Solución
RF29	Reproducción de video.	Para la reproducción de video se deberá abrir el video en cuestión y añadirse a la lista de reproducción, una vez ahí, podremos usar los botones de <i>play</i> y <i>stop</i> para reproducir y pausar respectivamente.
RF30	Webcam.	Tendremos un botón que se encargará de activar la webcam de nuestro equipo.
RF31	Capturar imágenes	Tendremos un botón que se encargará de capturar imágenes ya sean de la propia webcam o del video que se esté reproduciendo.

3. Diseño de clases propias

En la asignatura, hemos pasado por varias etapas en el desarrollo de gráficos en Java. Comenzamos utilizando la clase ya existente “Graphics” con la que pudimos comprobar en poco tiempo sus grandes limitaciones, entre ellas, las destacadas a la hora de dibujar. Cada figura necesitaba de su propio método “draw”, esto sin hablar de los parámetros necesarios para cada una de ellas.

Cabe mencionar, el problema que se comenta en el tema 5 de la asignatura y es que no se proporcionan clases asociadas a cada una de las figuras, esto implica que no podemos editar la figuras ya pintadas ya que se han pintando con unos ciertos atributos y no se pueden modificar.

Después de encontrarnos con tantos problemas y limitaciones a la hora de desarrollar nuestra práctica, se optó por el uso de la clase “Graphics2D”, en ella se incorpora lo que tanto necesitábamos, clases asociadas a las formas geométricas, de esta forma ya podíamos trabajar independientemente con cada una de ellas.

Con esta clase hemos desarrollado varias prácticas de la asignatura ya que de momento cumplía con todo lo que necesitábamos. La clase “Shape” contiene un gran número de figuras y a su vez a cada una de ellas se le podía modificar muchos atributos.

El problema vino cuando necesitamos que cada una de nuestras figuras tuvieran sus propios atributos, aquí nos encontramos con que tendríamos que crear clases propias ya que si seguíamos usando las propias de “Shape”, al cambiar las propiedades de una figura, se cambiaban en todas.

Por tanto, como se ha mencionado, la solución a esta limitación, es la creación de nuestras clases propias, cada una de ellas poseerá unas propiedades que podrán ser totalmente independiente de las propiedades que tengas las demás. De esta forma podremos dibujar varias figuras cada una con propiedades diferentes unas de otras.

3.1. Idea de diseño

En primer lugar, pensé en un diseño de una super clase “Interface” que heredaría de “Shape” y desde la cual heredaran todas las figuras propias que fuera a dibujar.

Este diseño lo llevé a cabo en su momento ya que tenía pensado implementar todos los métodos definidos de la super clase en cada una de las formas de dibujo, a la vez de que cada forma de dibujo, heredaría de su forma geométrica correspondiente.

De pronto me di cuenta de todo el código redundante que estaba realizando ya que tenía métodos que eran exactamente iguales en todas las figuras. Además, tenía otro problema, los atributos que poseía la super clase, obligatoriamente lo debían de poseer todas las clases que heredaran de ella y por ejemplo, la forma linea no posee la propiedad de relleno.

Esa idea tuve que descartarla y me decidí por crear una super clase abstracta en la que implementar todos los métodos que son comunes a todas las figuras y los no comunes definirlos como abstractos. De esta forma las figuras que hereden de la super clase, tendrán ya definidos todos los métodos comunes y los no comunes será definidos en cada clase propia.

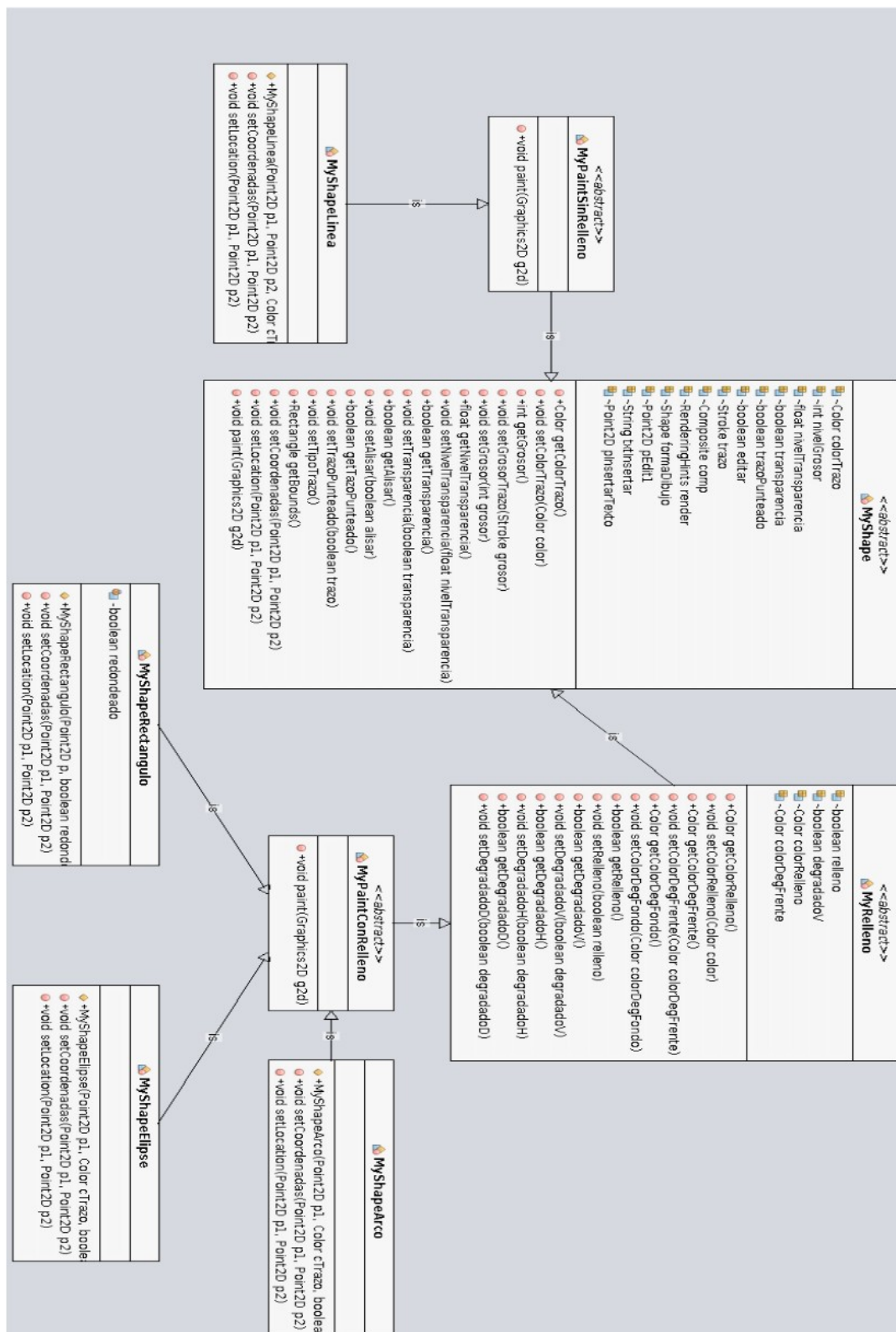
A esta idea había que sumarle la no necesidad de tener que heredar de la clase geométrica correspondiente, esto lo he realizado inicializando en el constructor de cada forma, su correspondiente forma geométrica ya que anteriormente heredábamos de ella y solo hacíamos uso en pocas ocasiones.

Con las propiedades que no son comunes para todas las formas como por ejemplo el relleno, he creado una clase propia intermedia que hereda de la super clase y que se encargará del relleno. De ella heredarán las formas que contengan la propiedad de relleno.

Para finalizar la explicación sobre como he desarrollado el diseño, comprobé que el método paint de cada una de las formas de dibujo, se repetía en todas ellas el mismo código, a diferencia de la linea por no poseer la propiedad del relleno, por tanto decidí de nuevo crear una clase intermedia que heredaría de la clase intermedia relleno su posee la propiedad y de la super clase si no la posee. Entonces para acabar, cada figura heredaría de su clase paint correspondiente.

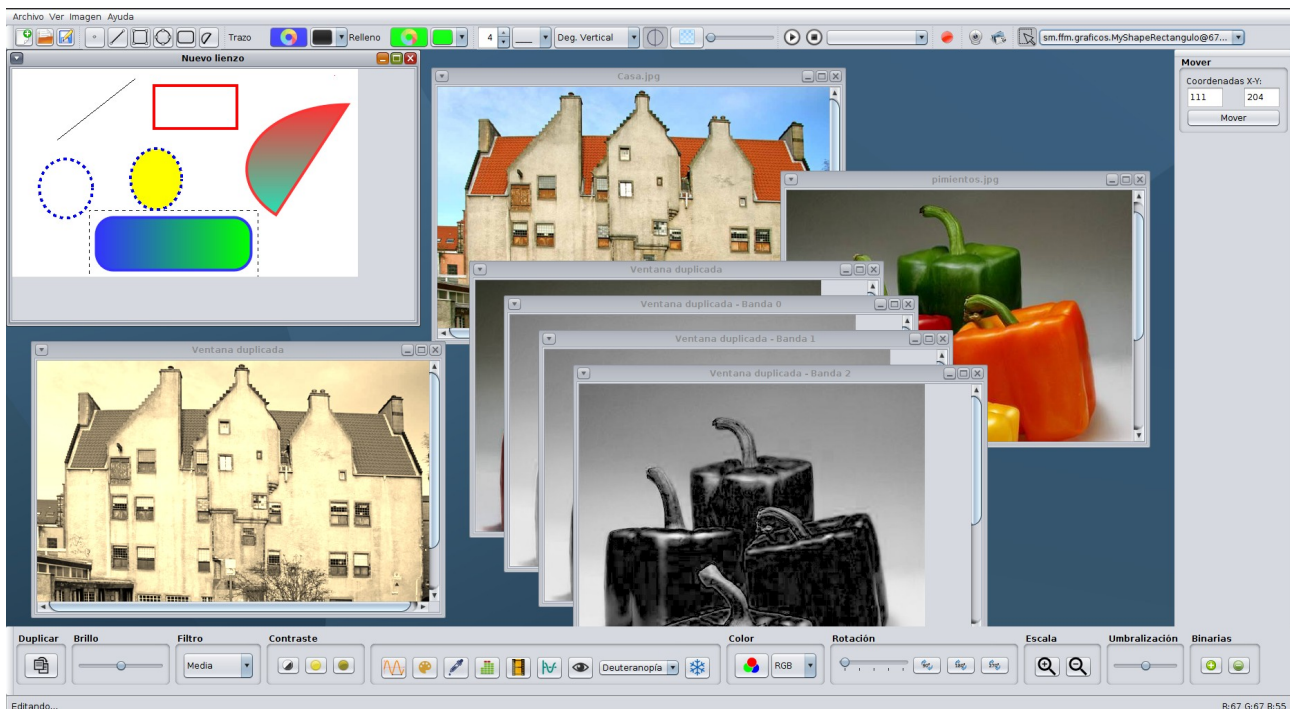
3.2. Jerarquía de clases

En el siguiente diagrama UML se muestra la jerarquía de clases que he llevado a cabo para satisfacer los requisitos de dibujo que se requieren.



4. Interfaz de usuario

A continuación, mostraré como ha quedado la interfaz de usuario para posteriormente analizar en profundidad cada una de las funcionalidades que he implementado.



En la captura de pantalla se muestra un ejemplo de funcionalidades, en este caso se han dibujado distintas formas, cada una de ellas con diferentes atributos, se puede observar como una de ellas está seleccionada para el modo edición y en la esquina superior derecha se muestra el desplegable con la figura seleccionada, en la barra de herramientas de dibujo, se muestran los atributos que posee la figura que se va a editar.

Por otro lado se muestra una imagen abierta y a su duplicado se le ha aplicado la operación sepia.

Para finalizar la breve muestra de funcionamiento, tenemos otra imagen abierta a la cual a su duplicado se le han extraído las bandas.

5. Manual de usuario detallado

En este apartado comentaré cada una de las funcionalidades de mi aplicación. Para ello realizaré capturas de pantalla de la interfaz para facilitar su explicación.

- **Menús**

1. Menú Archivo:



En este menú tendremos las opciones de crear un nuevo lienzo imagen donde poder dibujar nuestras figuras.

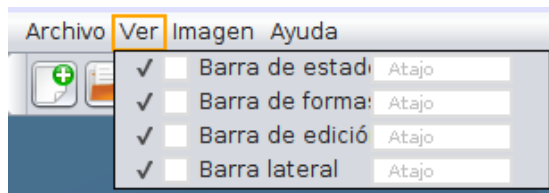
Tenemos la opción de Abrir mediante la cual podremos abrir tanto una imagen, un sonido o un video con el mismo diálogo y filtrando según el tipo de archivo como se muestra en la siguiente imagen.



Por último, la opción de Guardar, que como su nombre indica, se encargará de guardar las figuras dibujadas o la imagen modificada, también contendrá un filtro con los distintos tipos de formato.

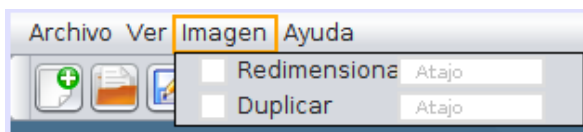


2. Menú Ver:



En el menú Ver tendremos las opciones correspondientes a activar o desactivar ciertas barras de herramientas y paneles, por si queremos una interfaz más sencilla o simplemente ocultar barras de herramientas que no vayamos a utilizar como por ejemplo la encargada de la edición de imágenes si solamente vamos a dibujar en un lienzo.

3. Menú Imagen:

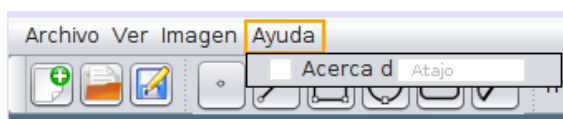


En el menú Imagen tendremos la opción de redimensionar el tamaño de dibujo del lienzo y para el siguiente lienzo nuevo que se abra tendrá el tamaño indicado.



Por último, tenemos la opción duplicar que se encargará de realizar una copia de la imagen que se encuentre abierta, en dicha copia podremos realizar todas las modificaciones que queramos sin afectar a la imagen original.

4. Menú Acerca de:



Para finalizar con los menús, tenemos el menú Acerca de... que contiene la información sobre el nombre de la aplicación, su versión y mi nombre como diseñador de ella :).



- **Barra de herramientas**

A continuación voy a explicar paso a paso cada uno de los componentes de la barra de herramientas y que función tiene. Dividiré la barra en diferentes capturas para facilitar su explicación.



En la parte izquierda tenemos al igual que en el menú Archivo, los botones de Nuevo, Abrir y Guardar. Realizan la misma funcionalidad ya que su método lo único que hace es una llamada a la opción del menú.

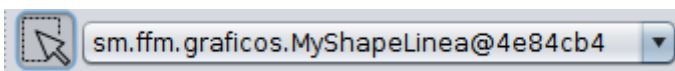
En la parte derecha tenemos las distintas formas de dibujo, tenemos el punto, la línea, el rectángulo, la elipse, el rectángulo con los bordes redondeados y el arco cerrado. Todas ellas pertenecen a un grupo de botones para que solo se pueda seleccionar una de ellas en cada momento.



En este trozo de la barra tenemos el color del trazo y el color del relleno en caso de encontrarse activado. Para cada uno de ellos, tenemos una lista desplegable con los colores más comunes y una paleta de colores con todos los colores existentes por si se quiere un color en especial.

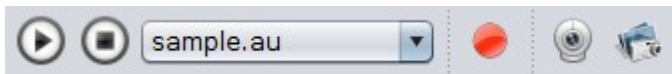


El primer componente es un Spinner y se encarga del grosor del trazo de la figura, el siguiente es un desplegable con las distintas opciones de trazo, ya sea trazo continuo o discontinuo, seguidamente tenemos el desplegable correspondiente a los distintos tipos de relleno. Tenemos la opción sin relleno, el relleno liso, el relleno vertical, horizontal y diagonal, para estos tres últimos, cuando se seleccionan, se abrirá automáticamente la paleta de colores para la selección tanto del color de frente como del color de fondo. El componente siguiente es el encargado de activar/desactivar el alisado de la figura y para finalizar, tenemos el componente encargado de activar/desactivar la transparencia de la figura, seguido de un Slider para modificar el valor de alpha que se aplica, dependiendo del valor que tome, se verá la figura con más o menos transparencia (inicialmente se encuentra a la máxima transparencia, hay que mover el Slider para poder visualizar la figura).

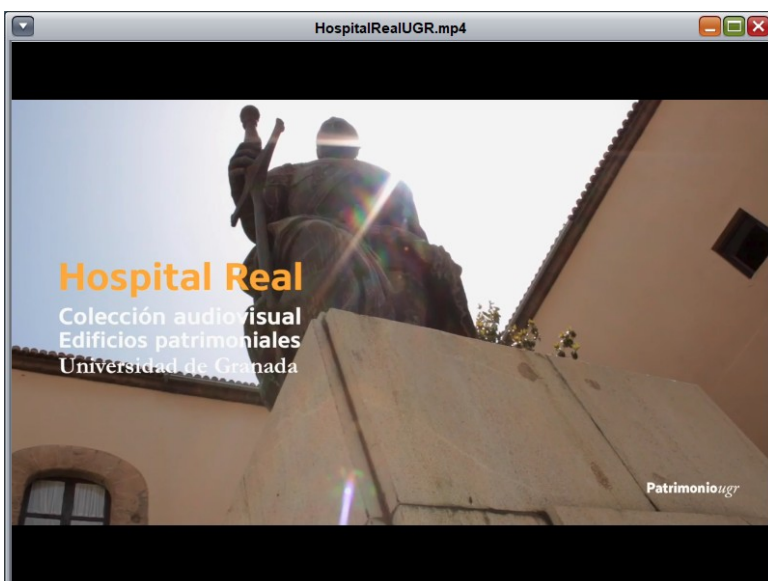


Aquí tenemos la parte de la barra de herramientas encargada de la edición de las figuras, tenemos un botón de edición que es necesario activarlo para después seleccionar la figura que queramos modificar. En la lista desplegable se irán añadiendo las figuras que se van dibujando en el lienzo, esto es posible mediante eventos. En el Javadoc proporcionado se encuentra documentado en cada clase lo que se realiza, de forma resumida, se crea una clase que representará un evento lanzado por un objeto lienzo, creamos una clase "listener" para los manejadores en la que definimos un método que se encargue de activar cada vez que se pinte una nueva figura. Cada vez que se llama a este método en la creación de la figura, se añadirá a la lista desplegable.

Cuando queramos modificar una figura, seleccionamos el botón editar y elegimos la figura, a continuación, se identificará la figura seleccionada en el lienzo con un marco que la encierra. Después de esto ya podemos cambiar todos los atributos que queramos a esa figura, para terminar la edición solamente tenemos que pulsar una de las herramientas y se desactivará el modo edición.

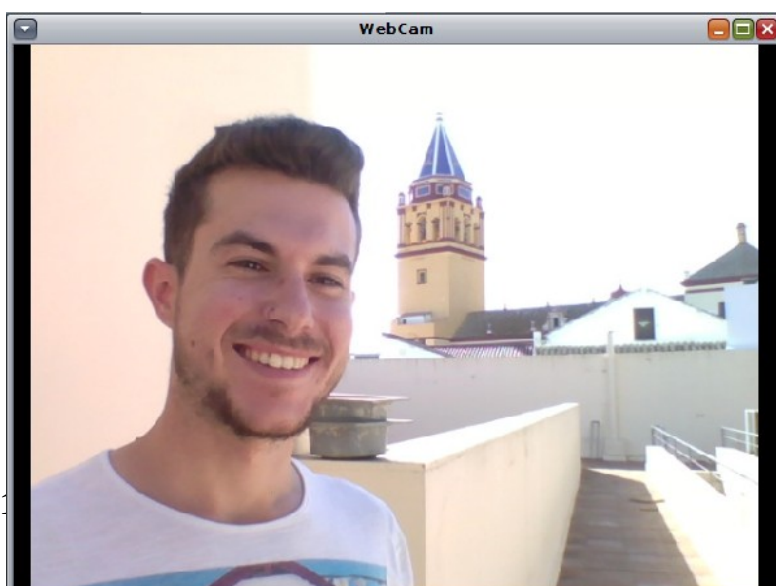


Por último, tenemos la parte de la barra de herramientas que se encarga de la reproducción de audio y video, en ambos casos se utilizan los mismos botones de play y stop y la lista desplegable que mostrará la lista de reproducción, seleccionando en ella el archivo a reproducir, se reproducirá el audio o el video. En caso de ser video se mostrará la ventana interna encargada de la reproducción.



Por otro lado tenemos el botón de grabación que tras pulsarlo mostrará una ventana al igual que en la opción de guardar para especificar el archivo donde se guardará el audio grabado. Tras finalizar la grabación, pulsaremos de nuevo en el botón, se almacenará el audio en el fichero seleccionado y se añadirá automáticamente a la lista de reproducción para su posterior reproducción.

Para finalizar, tenemos el botón de activación de la Webcam, que mostrará una ventana interna con el contenido capturado desde la Webcam de nuestro equipo.

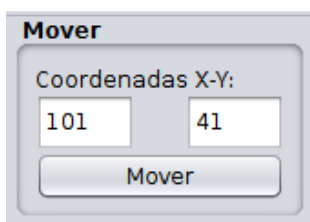


También tenemos como último botón, el encargado de realizar capturas como imagen de la Webcam o de la reproducción de video.



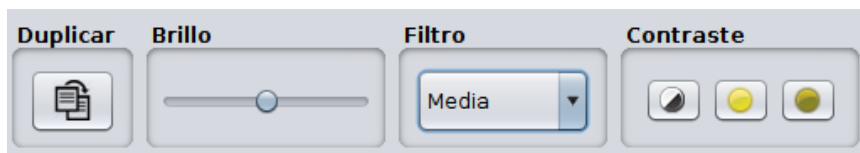
- **Barra lateral**

En la barra lateral unicamente tendremos un panel encargado del cambio de localización de las figuras.



En él, cuando seleccionemos una figura en el modo de edición, se mostrarán las coordenadas X e Y de dicha figura para posteriormente poder cambiar sus valores y cambiar su localización en el lienzo.

- **Barra de edición**



Como primer botón tenemos la opción duplicar que al igual que el menú Imagen/Duplicar, realizará una copia de la ventana imagen seleccionada y se podrá modificar manteniendo la original intacta.

Tenemos también un Slider que se encargará de aplicar la operación de brillo a la imagen, dicho componente se moverá entre los valores 0 y 255, siendo 0 el menor brillo posible (negro) y 255 el mayor brillo posible (blanco).

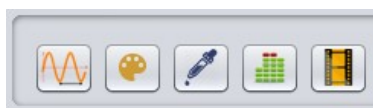
Tenemos una lista desplegable con los distintos tipos de filtros que podemos aplicar a la imagen. Por un lado tenemos el filtro Media, Binomial, Enfoque, Relieve y Laplaciano que son los tipos de “KernelProducer” que se encuentran definidos en el paquete “sm.image” proporcionado por el profesor.

Por otro lado tenemos Filtro norte, Filtro este y Filtro gauss que han sido añadidos por mí creando un nuevo kernel y pasándole como parámetro su matriz de convolución.

Filtro Norte	Filtro Este	Filtro de tipo Gauss
$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 1 & 1 \\ 2 & 7 & 11 & 7 & 2 \\ 3 & 11 & 17 & 11 & 3 \\ 2 & 7 & 11 & 7 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$

Por último, tenemos el panel encargado del contraste en el que tenemos un botón para el contraste que dentro del paquete “sm.image” llamará a la función TYPE_SFUNCION de la clase LookupTableProducer. El botón de iluminación que llamará a la función TYPE_LOGARITHM y el botón oscurecer que llamará a la función TYPE_POWER.

Como sus nombres indican, el contraste se encargará de contrastar la gama de colores de la imagen, el botón iluminar y oscurecer pues meterán más o menos iluminación a la imagen respectivamente.



En esta imagen tenemos algunas de las operaciones que se pueden realizar sobre la imagen.

Comenzamos de derecha a izquierda explicando el funcionamiento de cada una de ellas.

-Primero tenemos el botón encargado de la función Seno, se trata de una función “LookupOp” basada en el seno, esta función es calculada mediante un máximo K que será el mayor valor posible que tomará la función multiplicado por el cálculo del seno en función del valor del bucle y el parámetro correspondiente a la velocidad angular.

-Como segundo botón tenemos la operación Sepia, esta operación se encarga de recorrer pixel a pixel la imagen obteniendo mediante raster los componentes del pixel ya que en el resultado final influirán todos los componentes y son necesarios los valores de cada uno de ellos en cada componente.

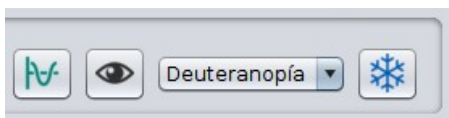
Se basa en una matriz de tonos marrones que irán modificando pixel a pixel su tonalidad para obtener un resultado final de un filtro como si de una imagen vieja se tratara.

```
sepiaR = min(255 , 0.393 ·R + 0.769 ·G + 0.189 ·B)
sepiaG = min(255, 0.349 ·R + 0.686 ·G + 0.168 ·B)
sepiaB = min(255, 0.272 ·R + 0.534 ·G + 0.131 ·B)
```

-El tercer botón se encarga de la operación de tintado sobre la imagen, el tintado se aplicará dependiendo del color que tengamos seleccionado en el trazo. Esta operación “TintOp” está incluida en el paquete “sm.image”.

-El cuarto botón se corresponde con la ecualización de la imagen, que consiste en obtener para una imagen un histograma con una distribución uniforme. Esta operación “EqualizationOp” se encuentra incluida en el paquete “sm.image”.

-Por último, el quinto botón que representa la operación de negativo sobre la imagen, esta operación hará que zonas de la imagen que estén iluminadas tiendan a apagarse y zonas que estén apagadas tiendan a iluminarse creando así el efecto de los negativos de las cámaras fotográficas antes de revelar. Esta operación llama a la función TYPE_NEGATIVE de la clase “LookupTableProducer” también proporcionada en el paquete “sm.image”.



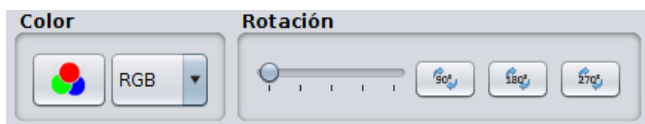
Para finalizar la parte de las operaciones, tenemos las de diseño propio, en un apartado siguiente se analizarán con más detalles.

Por ahora de forma más resumida:

-Tenemos un botón encargado de la operación “LookupOp” propia, esta operación se basa en la función $x^2/\exp(w)$, siendo w la velocidad angular. En esta operación, se obtiene el valor máximo en 127.0 en la escala de 255.0, por tanto todos los colores superiores a él los saturará aumentando su intensidad. Más adelante se mostrará la gráfica obtenida.

-Tenemos un botón encargado de la operación de diseño propio para la realización componente a componente en la que en función del tipo de disfunción visual se reducirán el componente de color correspondiente obteniendo de esta forma diferentes tipos de daltonismo.

-Por último, el tercer botón encargado de la operación de diseño propio para la realización pixel a pixel en la que a partir de una matriz de convolución de tonos grisáceos se aplicará un efecto como de “congelación” de la imagen.

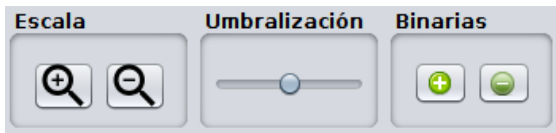


En estos dos paneles tenemos las siguientes funcionalidades:

-Por un lado la extracción de las bandas de la imagen según el espacio de color seleccionado en el desplegable, en el tenemos la opción de seleccionar el espacio de color RGB que extraería las bandas correspondientes a los colores primarios de la imagen, variando en cada una de ella el brillo que representaría la intensidad, podremos observar como aumenta o disminuye la intensidad del color correspondiente. El espacio de color YCC corresponde a la intensidad de la luz (Y) y dos crominancias (C) con informaciones del color. El espacio de color GREY con escala de grises en la que solo se extrae la banda correspondiente a los niveles de gris.

Y por último, YcbCr en el que el parámetro (Y) representa la luminancia, los parámetros (Cb y Cr) indican el tono del color, (Cb) para una escala entre el azul y el amarillo y (Cr) para una escala entre el rojo y el verde. Es utilizado para realizar correctamente la ecualización de una imagen ya que separa la intensidad de la parte cromática y la acromática.

-En el segundo panel tenemos las funciones correspondientes a la rotación de las imágenes, por un lado tenemos un Spinner que se encargará de variar los grados de rotación libremente de la imagen y luego tres botones que representan grados ya predefinidos que son los más usados normalmente.



La última parte de la barra de herramientas posee las siguientes funcionalidades:

- Por un lado a opción de poder modificar el escalado de la imagen, de esta forma podremos aumentar o disminuir su contenido con un factor de escala predefinido.

- El panel de umbralización, se compone de un Slider que establecerá el umbral que se requiere como parámetro en la función de `UmbralizacionOp`, en ella se realiza una umbralización en niveles de gris basados en el umbral establecido. Consiste en recorrer componente a componente la imagen obteniendo el valor RGB de cada uno, a partir del valor obtenido, se crea un nuevo color para posteriormente calcular la media a partir de cada una de sus componentes de color. Según el umbral establecido, para valores de media superiores al umbral, se modificará el color a tonos blancos y para valores por debajo del umbral, con tonos negros.

- Para finalizar, tenemos el panel encargado de la suma y resta binaria, esta operación se encargará de realizar la mezcla entre dos imágenes, pudiendo mezclar la primera con la segunda y viceversa, de forma que la segunda se adapte a la primera o la primera a la segunda. Esto se puede realizar mediante llamadas a la función `BlendOp` para la suma y `SubtractionOp` para la resta, ambas se encuentran incluidas en el paquete `sm.image`.

Editando...

R:145 G:134 B:88

Como último panel que comentar de la interfaz, tenemos la barra de estado, en ella en la parte izquierda se mostrará información de si estamos en modo edición o información de la herramienta de dibujo que tengamos seleccionada. En el lado derecho tendremos las tres componentes del color que darán información sobre los colores del pixel en el que nos encontremos con el ratón.

6. Explicaciones detalladas sobre operaciones propias

- Operador LookupOp basado en una función propia

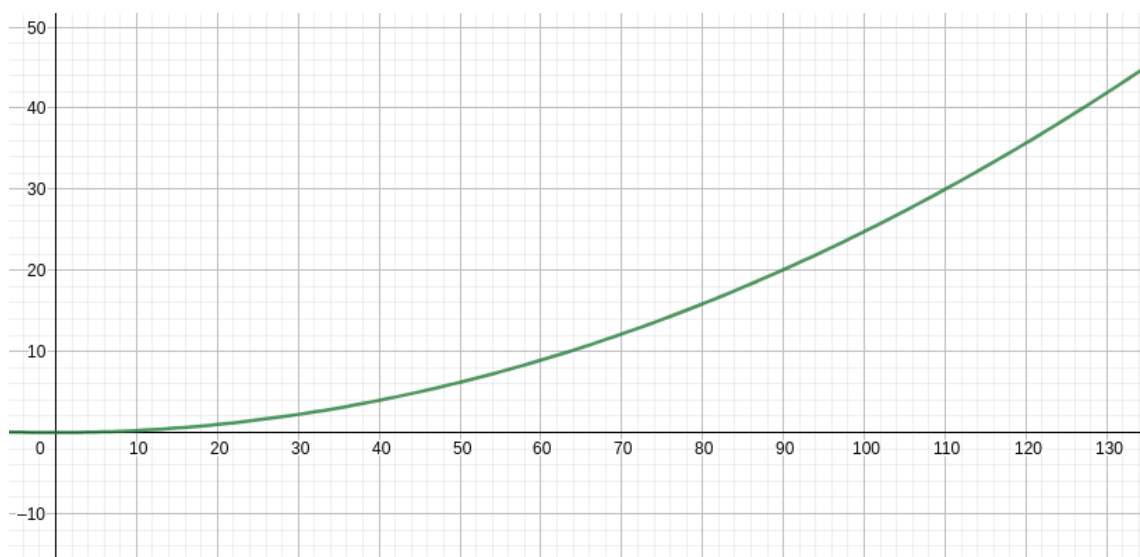
Para llevar a cabo este operador, he hecho uso de la página <https://www.geogebra.org/graphing?lang=es> en la que especificando la función que se desea implementar se mostraría una visualización de ella en una gráfica.

La función que se usó en prácticas anteriores fue la función seno por tanto no quise usar la función coseno ya que tendría similitudes. Luego probé a realizar una función en la que se usara el logaritmo neperiano pero me encontré con el problema de que el valor máximo que me tomaba la función era 5 por tanto no iba a tener mucho margen de modificación.

Por último me decidí por la función $x^2 / \exp(w)$ siendo w la velocidad angular que se establecerá en la llamada a la función, en mi caso he utilizado el valor 6 ya que conforme aumentaba ese valor más plana se volvía la recta hasta llegar al valor 10 que sería totalmente lineal en el eje X.

Como no sabía que valor tomaba como máximo, tuve que hacer en la función un primer recorrido para almacenar el máximo que se consigue y a partir de ahí ya poder calcular el valor de K que multiplicará en cada iteración al valor de la función. El máximo que he obtenido es 127.

A continuación muestro la gráfica de representación.



Como podemos observar, en los colores que corresponden a los valores mínimos de X como serían tonalidades de negros y colores fríos, apenas les realiza ningún cambio, conforme vamos avanzando se va modificando el valor de intensidad para colores más cercanos al punto máximo.



Se puede observar como en la imagen de Fry, para tonos fríos y azulados que se encontrarán en niveles bajos, apenas se aprecia la diferencia solamente que se vuelven un poco más oscuros, en cambio para colores anaranjados, se los lleva a colores verdosos y ya si hablamos de tonos blancos, se los lleva prácticamente a tonos negros.

En esta escala de colores podemos observar como en función de la posición del color se pasa según la modificación a otras tonalidades.



- Operación de diseño propio aplicada componente a componente

Para la operación de diseño propio aplicado componente a componente me inspiré de una viñeta vista en la siguiente página <https://www.cuantarazon.com/1033976/asi-es-como-ven-el-mundo-las-personas-con-distintos-tipos-de-daltonismo>, tras esto me documenté un poco en una página más profesional de una clínica óptica <https://www.clinicarementeria.es/patologias/daltonismo>, y para más información, consulté a mi compañera de piso que está en su último año de el Grado en Óptica y Optometría.

Después de toda esta información que nunca viene mal aprender cosas nuevas, decidí implementarlo como función propia.

En ella según el tipo de disfunción visual seleccionada en la lista desplegable que se proporciona, se establecerá la reducción del componente de color correspondiente.



A continuación mostraré los resultados de aplicar cada una de las disfunciones visuales a una fotografía.



La imagen superior izquierda sería la imagen original, en la superior derecha se aplica Deuteranopía y lo que hace es reducir las tonalidades de verdes dando lugar a tonos más marrones. En la inferior izquierda, tenemos Protanopía, que reduce las tonalidades de rojos, en esta fotografía no se aprecia del todo pero si puede verse como se aumentan los colores verdosos y azulados. Y por ultimo la imagen inferior derecha en la que se ha aplicado Tritanopía, que reduce los tonos azules, como podemos ver el cielo que en la imagen original tiene tonos un poco azulados, con esta disfunción se vuelven más amarillentos.


Obviamente se apreciarán más o menos los resultados según el grado de disfunción visual que posee la persona, en este ejemplo solo he querido mostrar una reducción del 20% de cada uno.

- Operación de diseño propio aplicada pixel a pixel

Para la operación de diseño propio aplicado pixel a pixel quise proporcionar la idea de congelar la imagen añadiéndole tonalidades de grises y azules simulando tonos fríos.

Me he basado en la matriz de convolución encontrada en el siguiente artículo sobre las matemáticas en el retoque digital de imágenes <https://es.slideshare.net/jorquera/retoquedigital-23519186>.

En él he utilizado la siguiente matriz:



$$\begin{pmatrix} (151, 198, 255), & (167, 202, 250), & (178, 207, 249) \\ (176, 220, 255), & (190, 223, 254), & (197, 220, 253) \\ (209, 224, 245), & (216, 229, 247), & (217, 228, 246) \end{pmatrix}$$

A partir de ella he adaptado los valores para obtener el resultado que esperaba reduciendo en cada pixel los componentes de rojo, verde y azul, siendo este último el que menos se reduzca y así resaltar más las tonalidades de colores fríos.

```
int congR = (int) Math.min(255, 0.54*pixelComp[0] + 0.43*pixelComp[1] + 0.50*pixelComp[2]);
int congG = (int) Math.min(255, 0.69*pixelComp[0] + 0.65*pixelComp[1] + 0.55*pixelComp[2]);
int congB = (int) Math.min(255, 0.93*pixelComp[0] + 0.94*pixelComp[1] + 0.92*pixelComp[2]);
```



En la imagen de Fry se puede observar fácilmente como según el color del pixel le aplica mayor o menor carga de niveles azulados obteniendo así el efecto deseado.

Bibliografía

1. <https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html>
2. <https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html>
3. <http://www.manualweb.net/java/>
4. <https://www.geeksforgeeks.org/java-robot-class-get-pixel-color-given-point/>
5. <https://www.freepng.es/>
6. <https://www.clinicarementeria.es/patologias/daltonismo>
7. <https://es.slideshare.net/jorquera/retoquedigital-23519186>
8. <http://www.jc-mouse.net/ingenieria-de-sistemas/uml-java-easyuml-plugins-para-netbeans>
9. <https://docs.oracle.com/javase/7/docs/api/javax/swing/ListCellRenderer.html>
10. <https://github.com/sarxos/webcam-capture>