Multimedia Suite.

Documentación del desarrollo

Requisitos

Para el desarrollo de este software se nos pedía realizar una “Suite” de herramientas multimedia que pudiera manejar tanto imágenes como gráficos, reproducción y grabación de sonidos, reproducción de vídeo, captura de webcam y capturas de pantalla.

Mas concretamente dentro de cada uno de los apartados anteriores se nos pedían unos mínimos a desarrollar que son:

- Dentro de la creación y edición de figuras gráficas :
  - Al menos la creación de las siguientes figuras:
    - Punto.
    - Línea recta.
    - Rectángulo.
    - Elipse.
    - Trazo libre.
  - Dentro de los atributos de estas figuras:
    - Color.
    - Trazo.
    - Relleno.
- Dentro de la edición de imágenes:
  - Modificación del brillo de manera manual.
  - Modificación del contraste de manera manual.
  - Modificación del contraste mediante 3 tipos preestablecidos:
    - Contraste normal.
    - Contraste iluminado.
    - Contraste oscurecido.
  - Filtros:
    - Emborronamiento.
    - Enfoque.
    - Relieve.
    - Fronteras.
  - Conversión de imágenes a negativo.
  - Conversión de imágenes a gama de grises.
  - Giro de imágenes de forma manual.
  - Giro de imágenes de forma preestablecida:
    - Giro 90º.
    - Giro 180º.
    - Giro 270º.
  - Escalado de imágenes.
- Dentro de la reproducción / grabación de audio:
  - Reproducción de archivos de audio sin codificar.
  - Grabación de archivos de audio sin codificar.
- Dentro de la reproducción de vídeos:
  - Reproducir al menos archivos de tipo AVI.
- Dentro de las capturas de webcam:
  - Mostrar por pantalla lo que captura la webcam.
- Dentro de las capturas de pantalla:
  - Capturar fotogramas de una reproducción de vídeo.

- Capturar fotogramas de una captura webcam.

Análisis

Analizando los anteriores requisitos llegamos a la conclusión de que el desarrollo de la aplicación a de llevarse a cabo mediante una estructuras de clases con POO(programación orientada a objetos) y para ello el mejor lenguaje que podemos utilizar es Java ya que desde sus orígenes fue creado para realizar este paradigma de programación.

Una primera aproximación a la estructura de clases que tendrá el software podría ser llevar una clase de gestión para cada “herramienta” multimedia que deseamos desarrollar, lo cuál nos dejaría con seis clases, que serían: Gráficos, Imagen,Reproducción de vídeo, Reproducción de audio, grabación de audio, visualización de webcam, pero como tanto las herramientas de dibujo como las de imagen tienen que estar juntas ya que se debe poder pintar sobre imágenes quedarían 5 clases de herramienta, dentro de cada cuál habrá que llevar distintas clases de representación de objetos de manejo como pueden en imagen-gráficos clases de representación de figuras, que posean los atributos necesarios de cada figura, para la reproducción de video y audio y para la webcam llevar controladores de esos medios.

Una visión abstracta de estas clases sería:

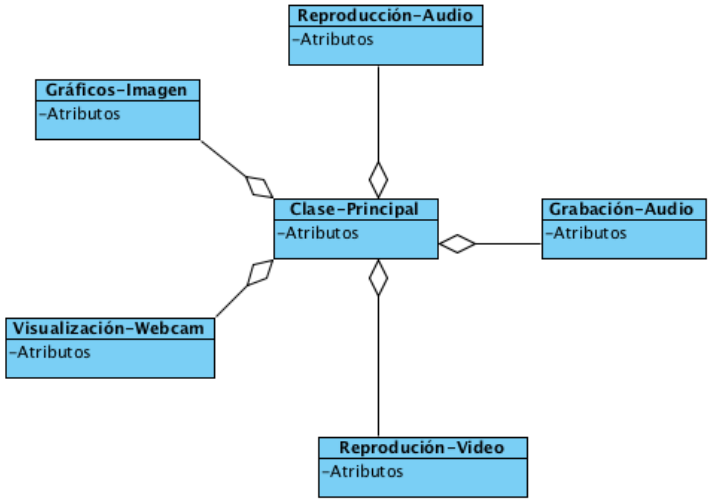

Figura 1: Una visión abstracta de las clases.

Como podemos observar no es para nada específico pero si nos da una idea de las primeras divisiones que se pueden hacer para llegar a la versión final que vendrá en el apartado de diseño. Podemos apreciar que hay que llevar una clase, en este caso llamada Clase-Principal que se encargará de la gestión de los objetos de las distintas clases. No he creado una clase para las capturas de pantalla puesto que es una operación que puede llevarse desde la clase-Principal sin problema.

Diseño

A la hora de enfrentarnos al diseño del software como es evidente al tratarse de una Suite de aplicaciones multimedia tenemos que desarrollar un entorno de ventanas o interfaz gráfico donde visualizar, editar, reproducir, etc los distintos medios que se nos piden como requisitos. Por esto el siguiente paso es adaptar el diagrama de clases creado en el apartado anterior a las “necesidades” de nuestro interfaz. Para ello la estructura de clases resultante es:

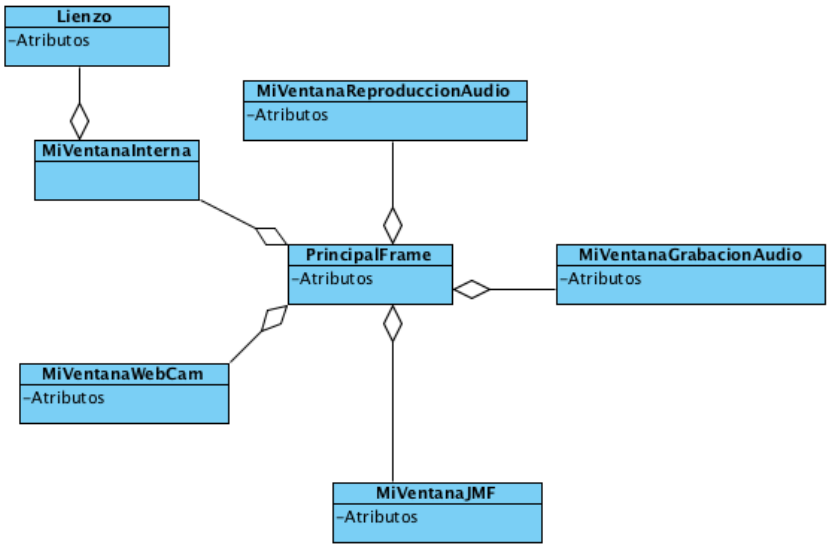

Figura 2: Diagrama de clases incluyendo interfaz gráfico.

A primera vista podemos observar que he añadido una nueva clase y he renombrado las demás, la clase añadida es Lienzo, la cuál se va a encargar de todo lo relativo a los gráficos y las imágenes, he querido separarla de la parte de interfaz porque son muchas operaciones las relativas a ésto y es mas escalable y re utilizable la clase Lienzo.

Aún después de esto todavía nos queda ver que objetos manejan estas clases, cuales ya existen y podemos utilizar de las bibliotecas de Java, cuales podemos mejorar y cuales debemos crear para que se puede llevar a cabo la aplicación.

Para las clases de reproducción y grabación de audio y video y para la de webcam voy a utilizar las clases proporcionadas en clase SMPlayer, SMRecorder, SMSoundPlayer, SMSoundRecorder y la API JMF de Java que se puede descargar aquí http://www.oracle.com/technetwork/java/javase/download-142937.html, estas clases no las voy a documentar puesto que no he sido yo el desarrollador. Sin embargo para la parte de dibujo si he creado clases propias para representar las figuras que se pueden dibujar en la clase Lienzo, estas se pueden ver en el siguiente diagrama:

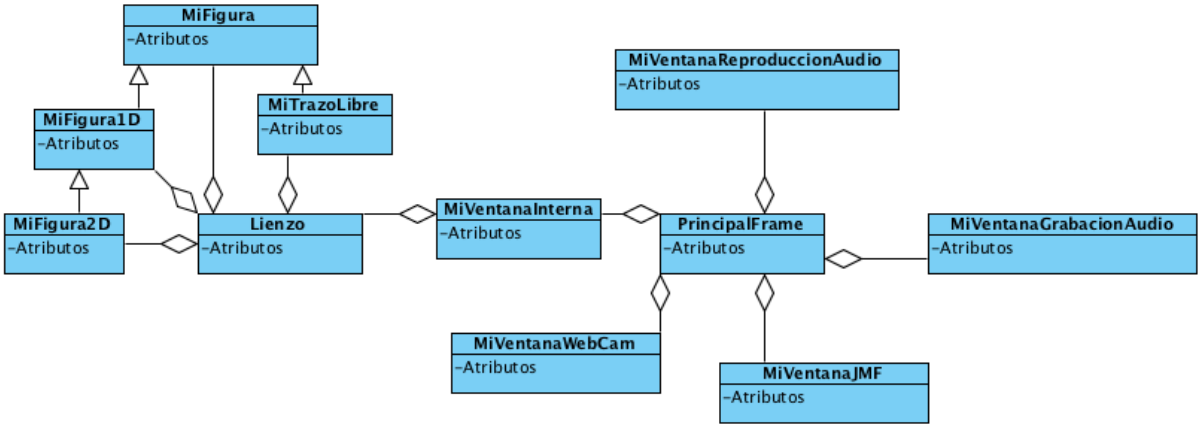

Figura 3: Diagrama de clases completo.

En este último diagrama observamos la generalización de las clases de Figuras, que explicaré con detenimiento a continuación, no sin antes mencionar que en los diagramas expuestos no se incluyen los atributos ni los métodos que cada clase tiene para que no se dificulten en demasía la comprensión de los mismos, para mayor conocimiento de éstos se puede mirar en el código de cada clase y leer los comentarios explicativos.

Desarrollo

Paso a comentar las clases, comenzaré por las clases de figuras seguiré por Lienzo y después comentaré las de reproducción grabación y proyección de webcam.

1. **MiFigura**: Es una clase abstracta, lo cuál significa que no se puede instanciar, que contiene los atributos y métodos comunes a todas las figuras que se pueden dibujar en el lienzo, los atributos que posee son : un entero que indica el tipo de figura que es, un Color que identifica el color del trazo o borde de las figuras y por último un “BasicStroke” que es un tipo de objeto que crea los trazos según los parámetros que se le indiquen. Los métodos que posee dicha clase son los relativos a la asignación y devolución de dichos atributos y un constructor que utilizaran las clases hijas. Las clases que heredan de MiFigura son:
    1. **MiFigura1D**: Con esta clase se pueden representar figuras del tipo punto, y ralla y añade un nuevo atributo Shape que es el tipo de figura que representa. Esta clase solo tiene un método constructor y los métodos set y get del nuevo atributo. De esta clase hereda:
        1. **MiFigura2D**: Con esta clase se pueden representar figuras como los rectángulos y las elipses, por lo que se tienen que agregar los atributos: Color1 y Color2 que indican los colores del relleno de la figura, dos booleanos relleno y gradual, que nos indican si la figura tiene relleno y si este es gradual(degradado) o no, y un entero que representa el tipo de degradado que puede o no tener la figura. Los métodos que tiene esta clase al igual que las anteriores son el constructor y los métodos set y get de cada uno de los nuevos atributos.
    2. **MiTrazoLibre**: Esta clase nos permite dibujar como si de un lápiz se tratara sobre el lienzo con el puntero del ratón, y hereda de MiFigura directamente porque al no se una figura de dos dimensiones no necesita los atributos de MiFigura2D, y como tampoco tiene un solo “Shape” si no que tiene varios tampoco puede heredar de MiFigura1D. En este caso el atributo que se añade es un “ArrayList” de “Shape”, puesto que lo que en realidad es un trazo libre no es mas que una sucesión de líneas unidas y eso se puede representar con este conjunto. Esta clase solo contiene un método

constructor y un método getFigura que devuelve el conjunto de “Shape”.

2. **Lienzo**: Esta es la clase mas compleja de toda la aplicación puesto que se encarga del manejo de todos los objetos del tipo de las clases explicadas con anterioridad, además de encargarse también de la edición de imágenes. No he separado imagen en una clase diferente puesto que por cada lienzo solo puede haber una imagen y todas las operaciones relativas se aplican en el mismo lienzo por ello todas están incluidas aquí. Para no extenderme demasiado explicaré los atributos mas importantes y los métodos que no correspondan a eventos del interfaz gráfico, ya que estos solo se encargan de enviar información a los otros. Comenzando por los atributos tenemos un “ArrayList” de tipo MiFigura que almacenará todas y cada una de las figuras que se van a dibujar en el Lienzo, una serie de enteros que representan los tipos de figuras, trazos, degradados, filtros, bordes y transparencias, una serie de colores que nos indican el color del trazo, y los colores primario y secundario de los rellenos, tres “BufferedImage” (Imágenes) que representan la imagen original, y las dos dos imágenes que se utilizan para las modificaciones. Un “BasicStroke” que nos indica el trazo actual, unos valores con decimales que representan el brillo, contraste, escala y ángulo de rotación de la imagen. También hay un “Shape” y un “Line2D” que se utiliza como auxiliar para las creaciones y actualizaciones de las figuras sobre el lienzo. Y dos dato tipo “Point2D” que se utilizan para la creación de figuras del tipo MiTrazoLibre. A parte de los métodos get y set de todos y cada uno de los atributos tenemos los métodos:
    1. getTipoTrazo: Que a parte de devolver un “BasicStroke” primero lo construye según el tipo identificado por un entero.
    2. IsNear: Método que devuelve true si la línea que se le pasa como argumento esta a 2 o menos píxeles del punto que se le pasa como argumento.
    3. GetSelectedShape: Devuelve un Shape con la figura seleccionada y asigna a la variable forma el tipo de forma de dicho Shape.
    4. MoveShape: Mueve la figura que haya seleccionada a partir del punto que se le pasa como argumento.
    5. UpdateShape: Actualiza la figura que se esta creando para ver el proceso de construcción en el lienzo.
    6. CreateShape: Crea las figuras con los parámetros seleccionados del interfaz gráfico y las añade al ArrayList de MiFigura del mismo lienzo.
    7. ConvolveOp: Realiza los cambios relativos al contraste y brillo de la imagen.
    8. RescaleOp: Realiza los cambios relativos a las transformaciones de la imagen.
    9. Escalar: Cambia la escala(tamaño) de la imagen.
    10. Rotar: Gira la imagen con respecto al centro de la misma.
    11. Contrastar: modifica el contraste de la imagen.
    12. ImagenGamaGrises: Convierte la imagen a gama de grises.
    13. MoverPosicionFigura: Mueve la figura por el lienzo.
    14. Paint: Se encarga de pintar sobre el lienzo primero la imagen que se decida cargar, si se decide alguna, y a continuación el ArrayList de MiFigura con los atributos que cada una de ellas posea.
3. **MiVentanaJMF:** Esta clase se encarga de representar el reproductor de video y también de audio de archivos codificados. Los atributos que maneja son un Player reproductor, dos Component de video y de control, el “path” del decodificador en el proyecto y un Codec que almacenará el codec necesario en cada ejecución. Para el funcionamiento de esta clase necesitamos la API JMF antes comentada y una biblioteca llamada “jffmpeg-1.1.0”(se puede descargar aquí: http://sourceforge.net/projects/jffmpeg/files/jffmpeg/jffmpeg-1.1.0/) que utilizo para reproducir ficheros codificados, tanto de audio como de video, entre otros MP3. Los métodos de esta clase son:
    1. El constructor de la clase que en este caso es privado ya que solo se quiere una instancia de la misma a la vez, que inicia todos los componentes y reproduce audio o vídeo según el formato del archivo que se le pase.
    2. GetPlayer: Devuelve el reproductor.

3. Close: Acaba con la reproducción.
4. Play: Arranca el reproductor.
5. GetInstance: llama al constructor y devuelve la instancia de la clase.
6. ControllerUpdate: Que se encarga de controlar la reproducción.

4. **MiVentanaReproduccionSonido:** Esta clase solo reproduce archivos de audio sin codificar, con formato wave. Utiliza la clase proporcionada en clase SMSoundPlayer y esta compuesta por un objeto de dicha clase y los métodos:
   1. Constructor: crea un nuevo objeto de la clase SMSoundPlayer.
   2. Los métodos asociados a los eventos de los botones del interfaz gráfica play y stop que arrancan y paran la reproducción.
5. **MiVentanaGrabaciónSonido:** Similar a la anterior pero en este caso utilizando un objeto SMSoundRecorder proporcionada también en clase. Los métodos son:
   1. Constructor: Crea el objeto grabador hacia el fichero que se le pasa como argumento.
   2. Los métodos asociados a los eventos de los botones del interfaz gráfica en este caso récord que comienza la grabación y stop que la para.
6. **MiVentanaWebCam:** También utiliza JMF y también se usa un getInstance ya que solo se quiere un objeto de este tipo a la vez. Tiene un objeto Player. Sus métodos son:
   1. Constructor: solo inicia los componentes gráficos y llama al método play:
   2. play: Método mas importante de la clase, se encarga de buscar el dispositivo de grabación y arrancarlo .
   3. Close: Finaliza la grabación.
   4. GetInstance: Llama al constructor creando el objeto de tipo MiVentanaWebCam y devolviéndolo.
7. **PrincipalFrame:** Esta es la clase principal del programa ya que se encarga de gestionar el interfaz gráfico y de crear los objetos de todas las demás clases según se vallan necesitando principalmente esta compuesta de los eventos relativos a los elementos del interfaz gráfico pero a parte de eso también tiene algunos métodos que son:
   1. Constructor: Inicializa los componentes del GUI.
   2. OranizaBotonesForma: se Encarga de poner todos los botones del tipo de forma a de-seleccionados.
   3. ConvertImageType: Convierte el tipo de imagen al tipo especificado por el argumento, en este caso lo convierte a RGB pero podría hacerlo a cualquier tipo.
   4. AbrirCaptura: Pinta la captura de pantalla realizada en un nuevo lienzo para su edición.
   5. GetFrame: realiza la captura de pantalla.
   6. AbrirVideo: Crea una nueva MiVentanaJMF y la añade al escritorio.
   7. AbrirSonido: Lo mismo que la anterior pero con sonidos de tipo wave.
   8. AbrirImagen: Igual que las dos anteriores pero con una imagen y un lienzo.

El resto de los elementos del interfaz gráfico vienen explicados en el “Manual de usuario” adjunto.

Bibliografía:

- Documentos de prácticas de la asignatura.
- Documentos de teoría de la asignatura.
- http://sourceforge.net/projects/jffmpeg/files/jffmpeg/jffmpeg-1.1.0/
- http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html
- http://www.youtube.com/playlist?list=PL0089BAD0145AE552
- http://www.slideshare.net/agustinmascheroni/jmf-14060707
- http://docs.oracle.com/javase/6/docs/api/java/awt/Shape.html
- http://docs.oracle.com/javase/6/docs/api/java/awt/geom/Line2D.html
- http://pic.dhe.ibm.com/infocenter/adiehelp/v5r1m1/index.jsp?topic=%2Fcom.sun.api.doc%2Fjava%2Fawt%2Fgeom%2FRectangle2D.html
- http://docs.oracle.com/javase/6/docs/api/java/awt/geom/Ellipse2D.html
- http://chuwiki.chuidiang.org/index.php?title=JFileChooser
- http://www.myjavazone.com/2010/06/jcolorchooser.html
- http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/image/BufferedImage.html
- http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/BasicStroke.html
- http://docs.oracle.com/javase/7/docs/api/java/awt/geom/Dimension2D.html
- http://docs.oracle.com/javase/7/docs/api/java/awt/RadialGradientPaint.html
- http://www.java2s.com/Code/Java/2D-Graphics-GUI/RadialGradient.htm
- http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/color/ColorSpace.html
- http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/archivos/file.htm