EVALUACIÓN

SISTEMAS MULTIMEDIA

CURSO 2023-2024

Convocatoria ordinaria (Junio)

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.

A continuación, se detallarán los objetivos y contenidos particulares, así como las acciones a aplicar sobre cada medio. El diseño del interfaz queda abierto a criterio del estudiante, si bien la evaluación tendrá en cuenta la complejidad del diseño, por lo que se recomienda usar el mayor número posibles de elementos Swing.

Las especificaciones indicadas en este documento son los requisitos mínimos, si bien se podrán incorporar todas las alternativas que se consideren oportunas para mejorar la calidad de la aplicación.

Medios e interfaz de usuario

La aplicación permitirá trabajar, de forma integrada, con todos los medios estudiados a lo largo del curso: **gráficos**, **imágenes**, **sonido** y **vídeo**. Para ello, tendrá un escritorio central en el que podrán alojarse ventanas internas de diferentes tipos en función del medio[1]. A continuación, se describen las principales características del entorno, si bien se podrán introducir cuantas mejoras se deseen para optimizar la funcionalidad o el aspecto visual.

• Gráficos e imágenes

La aplicación permitirá la gestión conjunta de gráficos e imágenes en un mismo tipo de ventana[2]; así, dada una imagen (ya sea creada nueva, leída o capturada), podremos tanto (1) dibujar sobre ella como (2) procesarla.

La manipulación de estos medios requerirá la incorporación de un tipo específico de ventana interna capaz de mostrar imágenes, tanto las leídas de un fichero como las creadas nuevas por el usuario o generadas al capturar una imagen instantánea a partir de un vídeo o webcam. Cada ventana mostrará una única imagen (existiendo, por tanto, tantas ventanas como imágenes estemos tratando), incluyendo barras de desplazamiento en caso de que la imagen sea mayor que la zona visible. El título de la ventana corresponderá al (1) nombre del fichero, si es una imagen abierta o guardada, (2) "nueva", si ha sido creada por el usuario, o (3) "captura", si es una instantánea captada de un vídeo o de la webcam. Si se tratase de una imagen asociada a una banda, deberá indicarse a qué banda (número o letra) corresponde. *[Opcional: Al mover el ratón sobre la imagen, indicar en la barra de estado las coordenadas del píxel sobre el que se está situado, así como su valor (color o nivel de gris, según el tipo de imagen)]*

Sobre las imágenes que aparecen en estas ventanas se podrá:

---

[1] El hecho de que haya diferentes tipos de ventanas internas, cada una correspondiente a una clase distinta, hay que tenerlo en cuenta a la hora de usar el método `getSelectedFrame`; por ejemplo, si se va a aplicar una operación sobre una imagen hemos de comprobar que la ventana activa es del tipo adecuado (es decir, que contiene una imagen). Para abordar este punto, se aconseja crear una superclase `VentanaInternaSM` de la que hereden diferentes subclases correspondientes a diferentes tipos de ventanas.

[2] Véase práctica 8.

- **Dibujar** usando las formas y atributos incluidos en la correspondiente barra de dibujo (véase siguiente sección). Sólo se podrá dibujar sobre el área de la imagen, por lo que estas ventanas tendrán definidas como área de visualización (área "clip") el área rectangular correspondiente a la imagen.
- **Procesar** imágenes, usando para ello las operaciones incluidas en la correspondiente barra de herramientas (véase siguiente sección).

En posteriores secciones se detallan los requisitos exigidos tanto en la generación de gráficos como en el procesamiento de imágenes.

• Sonido

La aplicación permitirá tanto la **reproducción** como la **grabación** de audio[3]. Para ello contará con una lista de reproducción, que tendrá una lista desplegable asociada, de forma que al abrir un nuevo fichero de audio éste se incluiría en dicha lista de reproducción[4]. Además, la aplicación tendrá:

- Un botón para **reproducir** sonido. Al pulsar el botón, se reproduciría el audio que esté seleccionado en ese momento en la lista de reproducción.
- Un botón para **parar** la reproducción o la grabación.
- Un botón para **grabar** audio, de forma que al pulsarlo se iniciará el proceso de grabación, que terminará cuando se pulse el botón de parada. El sonido se grabará en el fichero indicado por el usuario[5] y se añadirá a la lista de reproducción.

Tanto la lista de reproducción, como los botones de reproducir, parar y grabar estarán dentro de una barra de herramientas.

• [Opcional: Vídeo]

La aplicación permitirá tanto la reproducción de video como la captura de la cámara; para ello, se definirán ventanas internas específicas para cada tarea[6]. Así, la **reproducción de vídeo** requerirá la incorporación de un tipo específico de ventana que dispondrá de una zona de visualización central donde se mostrará el vídeo. La ventana tendrá como título el nombre del fichero que se está reproduciendo, existiendo tantas ventanas como vídeos tengamos abiertos. Además, la aplicación tendrá:

- Un botón para **reproducir** video *[opcional: que sea el mismo que para la reproducción de sonido]*. Al pulsar el botón, se reproduciría el video de la ventana seleccionada en ese momento (si esta fuese de tipo video).
- Un botón para **parar** la reproducción *[opcional: que sea el mismo que para la reproducción de sonido].*

Adicionalmente, la **captura a través de webcam** también requerirá un tipo específico de ventana interna que vaya mostrando la secuencia que esté captando la cámara. Para abordar los requisitos anteriores, la aplicación deberá incluir:

- Un botón correspondiente a la opción "Cámara", que lanzará una ventana que muestre la secuencia que esté captando la webcam seleccionada (usando la resolución también seleccionada).
- Un botón asociado a la opción "Captura" que permitirá la captura de imágenes de la cámara o del vídeo

---

[3] Véanse prácticas 13 y 14. Nótese que la práctica 13 sólo trabajaba con la *Java Sound API* y, por tanto, con los formatos y códecs soportados por dicha API; para la evaluación, se puede ampliar la funcionalidad con la incorporación de *JMF* o VLCj (práctica 14) y los códec que éstos soportan, si bien no es necesario.

[4] Al añadir un nuevo sonido a la lista de reproducción, éste no debe de empezar a reproducirse. Para reproducir el sonido, deberá de pulsarse el botón *play* (*stop* para parar).

[5] A la hora de seleccionar el fichero donde almacenar el sonido grabado, existen dos opciones: que se elija el fichero antes de comenzar la grabación, o que dicho fichero se elija al finalizar el proceso de grabación.

[6] Véase práctica 14.

Requisitos de carácter general

La aplicación contará con una barra de herramientas de carácter general que incluirá, al menos, los siguientes botones asociados a las clásicas opciones de archivo (todos ellos deberán tener asociado un icono y un "*ToolTipText*"):

- **Nuevo**. Permite crear una nueva imagen (que aparecerá en una nueva ventana). *[Opcional: El usuario podrá indicar el tamaño de la imagen a través de un diálogo que se lance en el momento de la creación o asociado a un menú general de opciones]*.
- **Abrir**: Abre el dialogo "Abrir fichero" y permite seleccionar un fichero de imagen, sonido o vídeo[7]. Dependiendo del tipo de fichero, éste se mostrará en un tipo de ventana u otra (o en la lista de reproducción, en caso del audio). Los formatos reconocidos serán los estándares manejados por Java. [*Opcional: Asociado al diálogo abrir definiremos filtros para que sólo muestre extensiones correspondientes a ficheros de formatos admitidos*].
- **Guardar**: Lanza el dialogo "Guardar fichero" y permite guardar la imagen de la ventana que esté seleccionada, incluyendo las figuras dibujadas (esta opción estará desactivada para el caso del vídeo). [*Opcional: Se podrá almacenar en cualquiera de los formatos reconocidos por Java (JPG, PNG, etc.), obteniendo dicho formato a partir de la extensión indicada por el usuario.*] [*Opcional: Asociado al diálogo guardar definiremos filtros para que sólo muestre extensiones correspondientes a formatos admitidos*].

En la barra de menú, además del menú **Archivo** (que incluiría las opciones "Nuevo", "Abrir" y "Guardar" indicadas anteriormente), se debe de incluir la opción **Ayuda** (que tendrá la opción "Acerca de" que lanzará un diálogo con el nombre del programa, versión y autor).

Requisitos de dibujo

La aplicación permitirá dibujar diferentes formas geométricas (líneas, rectángulos, etc.) con diferentes atributos (color, grosor, etc.) sobre una imagen. Para ello, se incorporará una barra de herramientas que dé acceso a todos los elementos necesarios para poder dibujar, incluyendo formas y atributos (los elementos de esta barra deberán tener asociado un "*ToolTipText*").

A la hora de dibujar…

El usuario podrá dibujar sobre cualquier imagen utilizando la forma y atributos seleccionados (véanse siguientes subsecciones). Para ello, hay que tener en cuenta los siguientes requisitos de carácter general:

- El lienzo mantendrá todas las figuras que se vayan dibujando.
- Cada figura tendrá sus **propios atributos** independientes del resto de formas (es decir, no compartirán los mismos valores). Cuando se dibuje la forma por primera vez, ésta usará los atributos que estén activos en ese momento (que no tienen por qué coincidir con los de las figuras que ya estén en el lienzo). La consecución de este requisito implica obligatoriamente la definición de una jerarquía de clases asociadas a las formas y sus atributos[8].
- El usuario podrá activar el **modo edición**[9] para modificar las figuras ya dibujadas[10]. Concretamente:

---

[7] Se puede poner una opción/botón de abrir por cada tipo de medio (es decir, un "abrir imagen", "abrir sonido", "abrir video"). También se puede optar por un único botón "abrir" que agrupe las tres opciones (el correspondiente código deberá reconocer de que tipo de medio se trata y lanzar una u otra ventana).

[8] Véase práctica 7.

[9] Véase práctica 8.

[10] En caso de que se hayan volcado figuras sobre la imagen, éstas no tendrán que editarse (es decir, se editan las figuras que estén en el vector).

- Se incluirá un botón “edición” de dos posiciones dentro de la barra de tareas (agrupado con el resto de las herramientas de figuras, de forma que solo pueda estar seleccionado uno de ellos). Si está pulsado implicará que estamos en “modo edición”; si se pulsa el botón de una forma de dibujo, automáticamente se saldrá del modo edición[11].
- Una vez en modo edición, el usuario podrá seleccionar una figura pulsando con el ratón sobre ella. Al seleccionarla, la figura se marcará mediante un círculo rojo y discontinuo cuyo centro se sitúe en el punto de localización[12] de la figura seleccionada[13].
- Para la figura seleccionada, se podrán editar sus atributos, esto es, se podrá modificar cualquiera de sus propiedades (color, relleno, etc.) sin más que cambiarla en la barra de herramientas (y los cambios se tendrán que ver reflejados en la forma)[14].
- El usuario podrá mover la figura seleccionada mediante una operación de “arrastrar y soltar” con el ratón
- La figura seleccionada se podrá “volcar” en la imagen. Para ello, se incluirá un botón en la barra de herramientas que, al pulsarlo, “vuelque” la figura seleccionada en la imagen (esto es, pinte esa figura en la imagen). Al volcar la figura ésta ya no estará en la lista de figuras del lienzo y, consecuentemente, no aparecerá marcada ni podrá moverse ni editarse (formará parte de la imagen).
- [*Opcional: Al seleccionar una figura, deberán de activarse sus propiedades en la barra de dibujo.*]

Formas de dibujo

La aplicación deberá permitir dibujar, al menos, las siguientes formas geométricas:

- Línea recta
- Rectángulo
- Elipse
- Forma personalizada (área) que defina un fantasma.

y opcionalmente[15]:

- Trazo libre *[opcional]*
- Arco *[opcional]*
- Curva con un punto de control *[opcional]*
- Curva con dos puntos de control *[opcional]*
- Polígono *[opcional]*
- Texto formateado[16] *[opcional]*

---

[11] Cuando se seleccione una forma de dibujo tras un proceso de edición (en el que la barra de dibujo muestra los atributos de la figura seleccionada), se podrá (1) volver a activar los atributos que hubiese en el lienzo antes de la selección o (2) mantener los atributos de la última figura seleccionada y que sean esos los que se usen para la nueva figura; se deja a criterio del estudiante elegir una u otra opción, si bien esto afecta a la lógica de la implementación.

[12] Si se siguieron las recomendaciones de la práctica 7, el punto de localización de una figura se obtiene mediante la llamada al método *getLocation()*.

[13] Se aconseja incluir una propiedad en las figuras que indique si está o no seleccionada. Añadir además en el método *draw* de la figura el código para pintar la marca en caso de que esté seleccionada. Nótese que será en el lienzo donde se activará/desactivará el estado “seleccionada” de las figuras

[14] Esto implicará que en los métodos *set* del lienzo asociados a atributos habrá que tener en cuenta si estamos en modo edición y, si fuese así, modificar el atributo de la forma seleccionada (si la hubiese).

[15] Solo se considerarán para evaluación las formas opcionales si se han implementado todas las obligatorias.

[16] La escritura del texto se podrá hacer directamente sobre el área de dibujo o bien utilizando un campo de texto o un diálogo previo en el que introducir la cadena. Independientemente de la forma de introducir el texto, éste deberá de aparecer en el punto de la imagen donde se haga el clic y con el formato indicado.

En la barra de herramientas aparecerá un botón (con icono) por cada forma de dibujo disponible[17]. Se usarán botones de dos posiciones agrupados, estando siempre pulsada la forma de dibujo que se va a pintar (salvo que haya una figura del lienzo seleccionada).

Atributos de dibujo

El usuario podrá elegir los atributos con los que se pintarán las formas (el diseño y organización del interfaz queda abierto al criterio del estudiante, si bien la complejidad de este se tendrá en cuanta en la evaluación). Al menos, deberán incluirse los siguientes atributos:

- **Color**. El usuario podrá elegir el color con el que se pinta tanto el trazo como el de relleno (siendo ambos el mismo). Para ello, la barra de herramientas ofrecerá la opción de elegir el **color** con un único botón que lance el diálogo de selección de colores; el botón tendrá como color de fondo el color activo en ese momento) *[Opcional: que se distinga entre el color del trazo y el de relleno, en cuyo caso en la barra de atributos deberá de haber una referencia al color de trazo y de relleno actuales[18]]*.
- **Trazo**. Se podrán modificar el grosor del trazo. [*Opcional: discontinuidad del trazo, pudiéndose dibujar, al menos, líneas continuas o líneas punteadas*] *[Opcional: estilos final y de unión de línea, más tipos de discontinuidad]*
- **Relleno**. El usuario podrá elegir entre rellenar con color liso o no rellenar. *[Opcional: relleno con degradado, en cuyo caso deberán de poder elegirse los dos colores que definen dicho degradado, así como la dirección del degradado, para lo cual se ofrecerá al menos dos posibilidades: horizontal y vertical]*. *[Opcional: más direcciones de degradado, relleno mediante imágenes predeterminadas, relleno radial]*
- **Alisado de bordes**. El usuario podrá activar/desactivar la mejora en el proceso de renderizado correspondiente al alisado de bordes. *[Opcional: incluir otras mejoras del renderizado[19]]*
- **Transparencia**. Se podrá establecer un grado de transparencia asociado a la forma [*Opcional: que el grado de trasparencia pueda variarse mediante un deslizador*[20]]

y opcionalmente:

- **Fuente del texto** (en caso de que se incluya el texto como forma). Se podrá establecer la fuente, tamaño y estilo del texto.
- **Reglas en la composición**. Se podrán definir reglas para combinar las nuevas formas con las ya existentes.
- **Transformaciones sobre la forma**. Se podrán aplicar traslaciones, escalados, rotaciones y deformaciones sobre la forma a dibujar.

[*Opcional: Cuando se cambie de una ventana interna a otra, los botones de forma y atributos de la barra de herramientas deberán activarse conforme a la forma y atributos de la ventana activa*].

---

[17] Nótese que, de las formas anteriores, hay algunas que se pintan en "un solo paso" (p.e., la línea), otras en "dos pasos" (p.e., la curva con un punto de control), otras en "tres pasos" (p.e., la curva con dos puntos de control), etc., entendido un paso como una secuencia *pressed-dragged-released*. Esto implica que, según la forma, habrá que llevar un control del "paso" por el que se va (y la tarea a hacer en dicho paso).

[18] Nótese que en las prácticas realizadas durante el curso no se distinguía entre color de trazo y color de relleno (se usaba el mismo para ambos); en este caso, habría que incorporar esta distinción.

[19] Cada mejora podrá ser activada/desactivada por el usuario de forma individual.

[20] Nótese que en este caso no sería sólo la semitransparencia (como en la práctica 7), sino que el usuario podrá definir el grado de transparencia que vaya desde opaco hasta totalmente transparente.

Requisitos de procesamiento de imágenes

La aplicación permitirá aplicar un conjunto de operaciones que se podrán llevar a cabo sobre cualquier imagen[21]. Cada una de estas operaciones se incluirán en una barra de herramientas donde cada elemento deberá tener asociado un "*ToolTipText*". Al menos, se deberán de poder realizar las siguientes operaciones:

- Duplicar, que creará una nueva "ventana imagen" con una copia de la imagen[22].
- Modificar el brillo y el contraste mediante un deslizador.
- Filtros de emborronamiento, enfoque, relieve y detector de fronteras.
- Emborronamiento "cometa" (véase práctica 9 para más detalles) donde el usuario podrá elegir el tamaño de la máscara mediante un deslizador.
- Emborronamiento "iluminado" 3x3 y 5x5 (véase práctica 9 para más detalles).
- Contraste normal, iluminado y oscurecido.
- Negativo (invertir colores).
- Operador basado en la siguiente transformación con parámetro $a \in [0,255]$: $T(x;a) = \frac{ax}{128}\ si\ x < 128;\ \frac{(255-a)(x-128)}{127} + a\ \ si\ x \geq 128$ (véase práctica 10 para más detalles). Incluir un deslizador con el que modificar el valor del parámetro "a".
- Operador "oscurecer zonas claras" basado en una transformación que aplique el siguiente efecto: "las zonas oscuras de la imagen se mantendrán iguales, mientras que las zonas claras se oscurecerán".
- Rotación de 180°.
- Escalado (aumentar y disminuir).
- Extracción de bandas
- Conversión a los espacios RGB, YCC y GRAY
- Tintado (con el color de frente seleccionado en ese momento) [*Opcional: incluir deslizador para indicar el grado de mezcla*].
- Sepia
- Ecualización
- Posterización (reducción del número de niveles por banda) mediante deslizador.
- Resaltado del rojo (esto es, mantener el color original en los pixeles rojos y llevar el resto a niveles de gris). [*Opcional: incluir deslizador que permita variar el umbral de selección de tono rojo*]
- Implementar un operador que permita cambiar un tono de color por otro (p.e., "que cambie los rojos por verdes") manteniendo su saturación y brillo. Los parámetros del operador serán (1) el color "C1" que se quiere cambiar, (2) el nuevo color "C2" por el que sustituirlo y (3) un umbral $T \in [0,360]$ que indique el margen de aceptación para que el tono de un color dado se considere "C1" (véase práctica 12 para más detalles). Para incorporar este operador a la aplicación, añadir un deslizador que permita modificar de forma interactiva el valor del parámetro $T \in [0,360]$, y dos botones para elegir los colores "C1" y "C2".
- [*Opcional: Un operador "LookupOp" basado en una función definida por el estudiante; dicha función deberá de tener, al menos, un parámetro*[23,24]].
- [*Opcional: Una nueva operación de diseño propio aplicada pixel a pixel; dicha función deberá de tener, al menos, un parámetro*[25,26,27,28]].

---

[21] Véanse prácticas 9-12.

[22] Ha de ser una copia, no una referencia a la original (es decir, no corresponde a una asignación entre variables, sino a la creación de una imagen nueva).

[23] Ha de ser una función no vista en clase ni disponible en *sm.image* (por lo tanto, no serían válidas las vistas en clase de teoría, la función seno/coseno, etc.)

[24] Respecto a los operadores basados en una función (lookup) definida por el estudiante, deberá indicarse claramente en la documentación qué función se aplica, mostrar una representación gráfica de la misma y explicar qué comportamiento se espera al aplicar esa función. Además, deberá explicarse qué parámetros tiene la función y cómo influyen en el resultado del operador. La documentación deberá de incluir ejemplos comentados donde se apliquen los mismos a imágenes reales. Si estas operaciones no están documentadas, o la explicación y justificación de las mismas es insuficientemente, no se darán por válidas.

[25] Necesario crear clase propia.

Las operaciones se irán aplicando de forma concatenada, es decir, una operación se aplicará sobre el resultado de operaciones aplicadas anteriormente. En el caso del **brillo**, el deslizador permitirá ir variando el brillo sobre la imagen que haya en ese momento, no sobre el resultado del cambio de brillo (es decir, si deslizamos el brillo a su máximo valor –lo que implicaría ver la imagen en blanco-, si después reducimos dicho valor se tiene que volver a ver la imagen inicial). Una vez que se elija otra operación, el brillo se aplicará de forma definitiva (y se concatenará con el resto de las operaciones). Esto mismo se aplicará a otros operadores que se basen en el uso de deslizador.

Requisitos de sonido

La aplicación permitirá tanto reproducir como grabar sonidos, tal y como se indicó anteriormente (sección Medios→Sonido). La reproducción se aplicará sobre los formatos y códecs reconocidos por Java. Respecto a la grabación, se pueden fijar valores para los parámetros de digitalización (codificación, resolución, frecuencia de muestreo, etc.) y formato de fichero[29].

[Opcional: Requisitos de video]

*La aplicación permitirá tanto **reproducir vídeo** como mostrar la secuencia que esté captando la webcam seleccionada (para ello contará con los correspondientes elementos en la barra de herramientas)[30]. Como se indicó anteriormente (sección Medios→Video), estas tareas estarán asociadas a ventanas internas específicas que dispondrán de una zona de visualización central; además, el usuario podrá controlar la reproducción (comenzar, pausar, etc.) mediante los botones situados a tal efecto en la barra de herramientas.*

*Adicionalmente, la aplicación permitirá al usuario **capturar imágenes** de la cámara [Opcional: del vídeo que se esté reproduciendo]; concretamente, lo hará de la ventana que esté activa, siempre y cuando sea una ventana de tipo "webcam". La imagen capturada se mostrará en una nueva ventana interna.*

Documentación

En esta convocatoria solo será obligatorio documentar el código usando *javadoc*[31] y generar la correspondiente API (especialmente importante en el caso de bibliotecas propias[32]). Debe incluirse tanto la descripción de la clase, como la de sus variables y métodos miembro (en este último caso, incluyendo tanto parámetros como información devuelta).

---

[26] Ha de ser una operación nueva no vista en clase, ni desarrollada en ninguno de los guiones de prácticas de cursos anteriores. No serían válidos los operadores incluidos en `sm.image` (p.e., umbralización, operador sobel, operadores binarios como la suma o la resta, mezcla de imágenes, etc.).

[27] La operación se aplicará pixel a pixel, considerando para el cálculo del pixel resultado todos los componentes del pixel origen.

[28] En relación a las operaciones de diseño propio, el estudiante deberá indicar claramente en la documentación qué operaciones ha implementado, justificar qué hace cada una de ellas y por qué, así como la correspondiente formulación matemática. Deberá especificarse qué parámetros tiene la operación e incluir en el interfaz los elementos necesarios para su selección y aplicación. La documentación deberá de incluir ejemplos comentados donde se apliquen los mismos a imágenes reales. Si estas operaciones no están documentadas, o la explicación y justificación de las mismas es insuficientemente, no se darán por válidas.

[29] Estos requisitos coindicen con lo desarrollado en la práctica 13.

[30] Estos requisitos coindicen con lo desarrollado en la práctica 14.

[31] Véase práctica 7.

[32] El javadoc se pude centrar en las clases de la biblioteca propia, siendo opcional para el resto (clases VentanaPrincipal, VentanaInterna, etc.).

Entrega

La entrega se realizará el día **3 de junio de 2024** a través de PRADO. Se deberá entregar un fichero comprimido (.zip o .rar) que incluya:

- Fichero ejecutable .jar de la aplicación. Este fichero deberá estar en la raíz del fichero comprimido y deberá empaquetar todas las librerías[33].
- Código fuente (proyectos Neatbean completos) incluyendo bibliotecas propias.
- API generada usando Javadoc (localizable en la raíz del fichero comprimido)

Examen

El examen se realizará el día **10 de junio a las 17:00**, correspondiente a la fecha fijada en el calendario de exámenes. Consistirá en un conjunto de preguntas teóricas y prácticas (éstas últimas basadas en implementaciones con ordenador) y tendrá una duración de 3-4 horas.

Para poder optar al examen se ha de superar esta práctica de evaluación (antes del examen, se publicará la relación de estudiantes que han superado la práctica de evaluación y, por tanto, pueden optar al examen final).

[33] Véase anexo de la práctica 7.

Anexo: Aclaraciones

En este anexo se incluyen algunas de las aclaraciones/sugerencias explicadas en clase a raíz de las consultas realizadas por los estudiantes en relación a este ejercicio de evaluación:

- Es **obligatorio definir clases propias para las distintas formas de dibujo consideradas**. Considerando el punto anterior, y como se indicaba en la práctica 7, el método `paint` del lienzo debería tener la siguiente forma:

```
public void paint(Graphics g){
  super.paint(g);
  Graphics2D g2d = (Graphics2D)g;

  g2.drawImage(img,0,0,this);
  for( XX s: listaFiguras) { //Para cada figura del vector
    // Una única llamada a un método de la clase XX que
    // pinte la forma 's'
  }
}
```

  donde "`s`" será una figura de la jerarquía propia (y "`XX`" su superclase). Nótese que en el cuerpo del bucle sólo debe haber una línea de código correspondiente a la llamada al método (externo a la clase del lienzo) que pinta la forma. El único atributo que podría activarse en el código anterior sería "`g2d.clip(areaClip)`" para definir el área de dibujo del lienzo (y, si se desea, la llamada a un método que dibujase un marco alrededor de la imagen y/o la figura seleccionada), si bien el resto de acciones propias de dibujo deben quedar delegadas a la forma.

  Cualquier solución que no tenga el `paint` anterior **se considerará errónea** e implicará dar por incorrecto todo el módulo de gráficos. Es preciso insistir mucho en esta parte ya que no hay que pensar que el ejercicio está bien "por el hecho de que pinte": tiene que hacerlo según exige la evaluación (es decir, hay que "contestar a la pregunta del examen").

- Las propiedades que puedan estar asociadas al lienzo (color, grosor, etc.) se usarán **para crear las figuras, nunca para pintarlas**. Esto queda claro en el punto anterior, donde en el método `paint` no se activa ningún atributo relativo a las formas.

- La **calificación** a la hora de evaluar se reparte por bloques: gráficos, imagen, sonido, vídeo, etc.; de estos bloques, gráficos e imágenes son los que tienen la mayor puntuación (en torno al 90%, por ser los que incorporan nuevas funcionalidades y retos), mientras que sonido y vídeo tiene una puntuación menor (por coincidir con lo desarrollado en clase). Para cada bloque, la forma de puntuar es, aproximadamente[34], la siguiente: se parte asumiendo que se tiene la totalidad[35] de la puntuación de ese bloque (p.e. 4 puntos) y se va restando por cada funcionalidad que falte o falle; la cantidad a restar dependerá de la funcionalidad: 0.25 si es básica, 0.5 si es media, o 0.75-1.0 si es alta (este último caso, está asociado a las funcionalidades nuevas y a los retos); también se sumará si se abordan requisitos de los indicados como opcionales (hasta saturar la puntuación de ese bloque). Lo anterior implica que, por ejemplo, si alguien entregase la unión de las prácticas P7-P14 sin abordar las nuevas funcionalidades o sin haber resueltos los retos, muy probablemente no "le daría" para aprobar.

[34] Este es un ejemplo basado en años anteriores, pero sirve de orientación a cómo podría ser este año. Es, por tanto, algo orientativo (no tiene por qué coincidir con el criterio, distribución, etc. que se determine para este año).

[35] Sobre ese total, se reserva un pequeño porcentaje para cubrir por los "extras" (así se pretende compensar a aquellos que aborden las partes optativas).