Universidad Complutense de Madrid

Proyecto de Sistemas Informáticos

Facultad de Informática
Curso 2011/2012

Muphic: Composición Musical Automática basada en Imágenes

Autores:
Johan Bertrand
Carlos Catalán
Vladimir Georgiev

Director:
Jaime Sánchez

25 de Junio de 2012

Prefacio

Synaesthesia is a neurological phenomenon in which stimulation of one sense induces the stimulus of another different sense. It is common for people who experience -or have experienced- this condition to identify certain images with an associated musical piece, and vice versa. This document presents a system which is able to convert a given image into a musical piece that identifies it. This process is performed in two stages: first, the image is analyzed in order to obtain an internal representation of it, then this representation is used as an input to a synaesthesia-based composition algorithm. The development of this project also includes a graphical interface, which helps the user operate the system, as well as allows him/her to handle the way the stages of analysis and composition work, to some extent. The aim of this project is to provide a multi-sensorial experience similar to synaesthesia, so that it can be used not only for purely artistic purposes, but for research and education objectives as well.

Keywords: synaesthesia, composition, music, image, analysis.

La sinestesia es un fenómeno neurológico en el que la estimulación de un sentido induce un estímulo en otro sentido distinto. Dentro de las personas que experimentan (o han experimentado) esta condición, es muy común la identificación de imágenes asociadas a una pieza musical y viceversa. Se presenta un sistema capaz de transformar una imagen dada en una pieza musical que le corresponda. Este proceso se realiza en dos fases: en primer lugar se analiza la imagen hasta tener una representación interna de la misma, y posteriormente se utiliza esa representación como entrada de un sistema de composición algorítmica basado en la sinestesia. El desarrollo del proyecto incluye además una interfaz gráfica que facilita al usuario el uso del sistema, así como un control en cierta medida de las fases de análisis y composición. El objetivo de este proyecto es proporcionar una experiencia multisensorial similar a la sinestesia, de forma que pueda ser usada no sólo con fines puramente artísticos sino también con objetivos de investigación o educativos.

Palabras clave: sinestesia, composición, música, imagen, análisis.

Autorización

Johan Bertrand, Carlos Catalán y Vladimir Georgiev autorizan a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes de los autores, la documentación y/o el prototipo desarrollado.

Johan Bertrand Carlos Catalán Vladimir Georgiev

a 25 de Junio de 2012

A Bruce, por sus consejos a lo largo del año,
y a todos aquellos que han estado ahí
para ayudarnos.

Índice general

Capítulo 1

Introducción

1.1. ¿Qué es Muphic?

Muphic es un software capaz de producir piezas musicales compuestas automáticamente a partir de una imagen. Es el resultado de un proyecto de sistemas informáticos comenzado en octubre del 2011 por 3 alumnos de Ingeniería Informática, cuyo propósito es generar música a partir de imágenes basándose en el fenómeno de la sinestesia.

Antes de proceder a describir el proyecto, es necesario explicar qué es la sinestesia, un término de gran importancia a lo largo del documento. Tal y como afirman Ramachandran y Hubbard [31]:

> *"La Sinestesia es una curiosa condición según la cual un individuo de cualquier otra manera normal experimenta sensaciones en una modalidad cuando una segunda modalidad es estimulada. Por ejemplo, un sinésteta puede experimentar un color determinado siempre que se encuentre con un tono particular(p. ej., C# puede ser azul) o puede ver un número dado tintado siempre de un cierto color (p. ej., '5' puede ser verde y '6' puede ser rojo)."*

Dentro de la sinestesia, se buscará la sinestesia auditivo-visual (aquella que relaciona los sentidos del oído y la vista), y más concretamente en la estimulación del sentido auditivo a partir de una percepción visual. Estudiaremos cómo reproducir esa asociación neurológica de las sensaciones visuales con los fragmentos musicales para poder generar composiciones musicales.

De forma resumida, dado que se entrará en detalle en secciones siguientes, la aplicación desarrollada analiza una imagen de entrada y utiliza el contenido de dicho análisis para generar una pieza musical mediante algo-

ritmos de composición automática.

El sistema trabaja en dos etapas secueciadas:

- Análisis de imágenes: la imagen de entrada se debe procesar y estudiar hasta obtener la información deseada (formas, colores, tamaños, ...).
- Composición algorítmica: con la información obtenida a partir de la imagen y el conocimiento de cómo se relaciona con la música, se pasa a componer, de forma automática, todas las partes que forman la pieza musical: ritmo, melodía y armonía.

Naturalmente, la segunda etapa va a depender del resultado del análisis de la primera: con distintos análisis de imagen se obtiene pequeñas diferencias en las descripciones de las imágenes de entrada. Y por tanto las composiciones musicales serán parcialmente diferentes, ya que la información procesada por el compositor será distinta en cada caso. Además, la elección de los algoritmos de composición cambiará de forma significativa la pieza musical final.

En el desarrollo de este proyecto hay ciertos aspectos importantes que se ha considerado y es necesario mencionar:

- *La base de la correlación audio-musical será la sinestesia*:

  La sinestesia, como ya se ha comentado, es la pieza clave en la creación de música basada en imágenes (Sección 1.3.1) frente a otras alternativas de planteamiento.
  Entre estas alternativas, cabe destacar la reacción psicológica sociocultural del cerebro humano ante la música y los colores. Es decir, estudios muestran que el verde, el azul y otros colores con longitudes de onda bajas se relacionan con la calma, mientras que colores con longitudes de onda altas aumentan el nerviosismo y la inestabilidad [2]. Hay, por otro lado, multitud de investigaciones sobre la psicología de la música y su relación con distintas emociones y sensaciones, como las asociaciones entre los modos griegos musicales y los estados de ánimo [11]. Una posible rama de desarrollo procedería juntando ambos ámbitos, la psicología del color y la de la música, para generar la música deseada en función de la imagen dada.

- *No se tienen en cuenta objetos físicos*:

  El objetivo del análisis no es tanto obtener información de qué es lo que la imagen representa (reconocimiento e interpretación), sino los colores y formas que contiene, y su distribución y características dentro de la

misma (segmentación y descripción). Es decir, si se tiene una imagen con un elefante no se quiere reconocer el "elefante" sino que hay una figura redondeada con una parte alargada (la trompa) y que tiene un colo azul grisáceo y se encuentra en el centro de la imagen.

- *La imagen es estática*:

  Se consideran para el análisis formatos de imagen estática (tales como bmp, jpg o png), y no animada (vídeo o archivos de animación). Este planteamiento condiciona el proceso de correlación imagen-música, ya que se pretende obtener una salida no estática, como es la música, a partir de una imagen inmóvil. Se verá más adelante cómo se obtiene ese efecto de *dinamismo* a partir de la información proporcionada por una imagen.

- *Generación de contenido frente a acoplamiento de creaciones preestablecidas*:

  Es importante recalcar que en el proceso de composición las piezas generadas se construyen desde cero y no parten de ninguna estructura predefinida. No se usarán por tanto piezas ya compuestas o estructuras conocidas como "ladrillos" para construir una pieza nueva, como una base de datos o adaptaciones de piezas musicales enteras. Es cierto que se pueden seleccionar ciertos parámetros de la composición pero sirven para matizar las composiciones (tempo, instrumentos, ...).

Además, resulta importante resaltar que, siendo el objetivo final del proyecto generar música, no se ha buscado que compita con obras de grandes compositores. El modelo a seguir es la creación de la llamada música de ambiente; es decir, música que, siendo voluntariamente no atrayente ni excesivamente interesante, tiene como requisito principal el no ser molesta. Se sigue por tanto la línea de la música mostrada por Brian Eno en *Música para Aeropuertos (1978)*:

> *"[La música ambiente es] Algo de lo que puedes entrar o salir discretamente. Puedes atender o puedes elegir no distraerte con ella si quisieras hacer algo mientras la música está reproduciéndose."*, ([4], entrevista con Brian Eno).

Por último, se ha de añadir que el ámbito de la composición basada en imágenes es uno muy estudiado pero a la vez poco desarrollado. Es decir, aunque existe una gran cantidad de estudios sobre la sinestesia y la composición musical automática (como se puede ver en la Sección 1.3), aún quedan muchas cuestiones que investigar y áreas que profundizar.

1.2. Motivación

La principal motivación de este proyecto nace, por supuesto, del interés de los participantes en la música y la aplicación de la computación a la misma. Aunque existe una lista interminable de aplicaciones orientadas a esa relación música-informática, el interés de este proyecto parte de un área en particular: la composición algorítmica.

El campo de la composición algorítmica consta de muchos estudios y trabajos realizados sobre la materia. Sin embargo, gran parte de ellos caen dentro de dos casos: o bien se basan en el acoplamiento y unión de diferentes piezas previamente compuestas aplicándoles ciertas modificaciones (consiguiendo resultados auditivamente agradables, pero en ningún momento "nuevos"), o bien busca una creación completa de la pieza musical. Como ya se comentó en la sección anterior, es este último campo el que motiva el desarrollo de este proyecto. Se busca por tanto la composición genuina de piezas musicales, útil como fuente de inspiración para usuarios compositores o la generación de música de ambiente.

Dentro de la composición algorítmica, el interés de los integrantes del proyecto se centra sobre todo en dos aspectos fundamentales:

- De todas las formas posibles de generación de música algorítmica existentes, se tiene especial interés en una generación determinista. Esto es, en vez de partir de algoritmos genéticos o cualquier otro tipo de diseño basado en un entrada aleatoria, se desea obtener una pieza musical que suponga la representación de un elemento de entrada perteneciente a contexto no auditivo. Es esta búsqueda la que lleva a plantearse el usar imágenes como entradas a estos algoritmos.
- Se busca además que la relación entre la imagen y la música generada no esté sujeta a concepciones culturales o personales, que pueden variar en cada usuario. Es por ello que se hará uso de la sinesteria, fenómeno que relaciona diferentes sentidos, y que además es foco de muchos estudios por la rama de la psicología.

Se relacionan así dos elementos que incitan gran interés en la comunidad científica y que, si bien han sido estudiados por separado (como se aprecia en la siguiente sección), juntos componen un objeto de estudio apenas observado. Es la motivación de este proyecto el estudiar y experimentar en este ámbito, con el objetivo de expandir su trasfondo académico y observar las posibilidades que ofrece.

Cabe destacar también la inclinación a crear una aplicación de esta índole para dispositivos móviles. Una versión simple y accesible de este sistema

puede ser de gran interés en este mercado, ya que las entradas gráficas se pueden obtener con facilidad gracias a las cámaras integradas en la mayoría de las plataformas portátiles. También se facilita enormemente el proceso de testeo de los diferentes resultados permitiendo su rápido progreso. Tras estudiar la viabilidad de enfocar la aplicación a sistemas móviles, se ha preferido orientar el proyecto a ordenadores personales, con el objetivo de simplicar la implementación y dar más importancia al diseño de algoritmos de composición. No obstante, se ha preparado la aplicación para facilitar una futura portabilidad a móviles.

1.3. Estado del Arte

1.3.1. Estudio sobre Sinestesia

Existen varios tipos de sinestesia. De entre todos ellos este proyecto se centra en la llamada sinestesia musical. Esta consiste en mezclar la experiencia auditiva con la visual. Se estudia este tipo de sinestesia por ser una fuente razonable de información a la hora de encontrar un algoritmo para pasar de imágenes a música. Según el neurocientífico David Eagleman, las personas de forma innata tienen la tendencia de conectar sentidos, entre otros visual y auditivo (sonidos con formas y sonidos con colores, [6]).

Varios artistas y científicos a lo largo de los años han expresado su capacidad sinestésica a través de sus obras o documentando y experimentando este fenómeno. Desde la antigua Grecia se intentaba encontrar una equivalencia entre los colores y los sonidos, Aristóteles en su ensayo (*De Sensu et Sensato* [3]) realiza una descripción de los colores comparándolos directamente con la armonía presente en la música. Otra teoría dentro de la vertiente científico-filosófica la desarrolló Newton que asignó un color a cada nota de la escala diatónica (siete notas) [13], siguiendo los colores obtenidos a partir de su prisma, que también eran siete, de tal manera que le asignó al C (Do) el color rojo (primer color del espectro visible) hasta llegar a B (Si) que se correspondería con violeta (último color).

Otra aproximación a la relación música-imagen que normalmente siguen los artistas es a través de la experiencia con el fenómeno de la sinestesia. Se encuentran entre ellos al compositor Scriabin [12], quien hizo una asociación entre los colores y los acordes que aparecen en música que queda reflejada en su círculo de quintas. Kandinsky [32], un pintor ruso, partiendo de la pintura ha investigado y defendido fervientemente la posibilidad de asociar la música a la pintura. Resalta aspectos como asociar la escala de intensidad de los sonidos con la intensidad de los trazos de la pintura, también distingue entre distintos timbres de instrumentos según los colores, asignando a

cada familia de instrumentos colores diferentes.

Las investigaciones llevadas a cabo por el psicólogo Köhler en 1929 [34] y las hechas recientemente demuestran que el ser humano tiene una conexión profunda entre los sonidos y las formas además de los colores. Marks, en *The Unity of the Senses* [20] encuentra una asociación entre los tiempos en música y las formas, tal que cuanto más angular e irregular es una figura más rápidas deben ser las notas o el tempo. También declara sobre los colores que:

> *"Por desgracia, al final uno descubre que no hay una asociación sinestésica entre notas musicales y colores que prevalezca ante las demás".*

Después de ver la aproximación de Kandinsky a través de la experiencia (como el trabajo de Scriabin ya mencionado) y de la lógica o las matemáticas-físicas (como los propuestos por Aristóteles, Newton o Köhler) queda claro que la capacidad sinestésica de cada persona es única y puede haber grandes diferencias entre las experiencias sinestésicas de diferentes individuos. Es por tanto importante poder encontrar asociaciones aceptables y ponerlas en práctica para comprobar los distintos resultados. Para poder probar las diferentes correspondencias entre música e imagen se ha habilitado en la herramienta el poder elegir diferentes asociaciones entre las citadas en esta sección y otros autores.

1.3.2. Composición basada en imagenes

Para poder componer música a partir de datos gráficos se necesita crear una correlación entre los elementos que se tienen en la parte gráfica y en la parte musical. Esta correspondencia no es fácil de conseguir, y a través de la historia se han desarrollado varias teorías que se han llegado a poner en práctica. Los primeros intentos de síntesis fueron los instrumentos llamados "órganos de color" que acompañaban al sonido con una muestra visual.

El primer instrumento lo construyó Louis Bertrand Castel (1730), se trataba de un clavecín al que se le había incorporado una pantalla y un sistema de iluminación. Cuando se pulsaban las teclas se iluminaban los colores correspondientes en la pantalla [33]. A este experimento le continuaron muchos otros que incorporaban mejoras, pero al ser una correspondencia muy directa entre nota y color, al final se obtiene poca diversidad. Además es una correspondencia de la música a la estimulación visual mientras que el interés de este proyecto se centra en la otra dirección, de la imagen a la música.

Desde hace unas décadas, gracias a los avances tecnológicos, se han investigado activamente los algoritmos automáticos de composición. Una posible clasificación de los diferentes algoritmos basada en su característica principal sería: modelos matemáticos, sistemas basados en conocimiento, gramáticas, evolutivos, sistemas con aprendizaje e híbridos [9].

- Modelos matemáticos: los más usados son los procesos basados en sistemas estocásticos y cadenas de Markov. Como ejemplo significativo de estos modelos destaca Cybernetic Composer [5]. Otra subcorriente son los basados en la teoría del caos [14].
- Sistemas basados en conocimiento: dependiendo de cómo se represente el conocimiento y cómo se manipula se pueden hacer diferentes clasificaciones. Como ejemplos relevantes se tienen CHORAL [15] o SICOM [8].
- Gramáticas: fueron las primeras técnicas usadas. Se suelen mezclar con técnicas probabilísticas obteniendo gramáticas indeterministas, ya que si no, puede producirse música poco variada. Destacan el proyecto EMI [15] o Steedman y su generador de música Jazz [9].
- Algoritmos evolutivos: se dividen en dos posibilidades según la función de evaluación. La primera es usando una función de evaluación automática. Como ejemplo se tiene McIntyre [9]. La otra posibilidad es usar una evaluación humana, que es bastante más lenta y además ambigua. La herramienta de improvisación de Jazz de Biles, GenJam [16], es un ejemplo importante de este tipo.
- Sistemas con aprendizaje: están abiertas varias líneas de investigación según las diferentes formas del proceso de aprendizaje (adquisición de conocimiento del sistema). Una posibilidad es través de redes neuronales artificiales, un ejemplo significativo es EBM [9]. Otra manera es con aprendizaje automático (aprendizaje máquina) donde destaca el ejemplo de MUSE [9].
- Híbridos: intentan combinar lo mejor que ofrecen los diferentes sistemas posibles, un ejemplo relevante es HARMONET [9] que combina sistemas con aprendizaje (redes neuronales) con sistemas basados en conocimiento.

Pero no solo la algoritmia es variada, también se puede clasificar de diferente forma según el tipo de música que se quiera componer: micro-composición (diseño de sonidos) y la macro-composición (combinación de sonidos ya diseñados para la creación de una obra musical) [18]. Para este proyecto, interesan los algoritmos de macro-composición basados en el

fenómeno de la sinestesia.

Hay varias aproximaciones de compositores basados en imagen, algunos de ellos como Phonogramme [35] [30] consisten en un editor gráfico que interpreta la imagen como una partitura. La imagen representa la relación bidimensional de tonos y duración de los sonidos, es decir, la imagen es una partitura que se lee de izquierda a derecha en el tiempo y de abajo a arriba en la altura de los tonos.
El problema de este algoritmo desde el punto de vista de la sinestesia es que las imágenes son partituras y deben estar diseñadas para ser usadas con este fin (usando el editor gráfico), no acepta cualquier entrada gráfica. Esta línea de investigación se hace valer de la consexión psicológica que se tiene entre formas y sonidos, pero realmente no busca la sinestesia como base.

Otra opción es la propuesta basada en el concepto de "croma" (o índice de cromatismo) [7]. Se subdivide la imagen en bloques (ladrillos cromáticos) los cuales tienen un índice de cromatismo, esto sirve para generar o asignar trozos de música que pueden ser cogidos de una base de datos o ser creados por un experto (compositor). Al no haber una correspondencia directa entre la imagen y la música, este proyecto no está fundamentado en la sinestesia. Si bien recoge algunas ideas más adelante pierde la esencia de la sinestesia.

También se destaca el trabajo realizado por Xiaoying Wu y Ze-Nian Li [35] en el que se analiza la imagen en tres pasos. Primero: hacer una *Partición* de la imagen en piezas (divisiones, trozos) más pequeñas. Segundo: realizar la *Secuenciación* de esas piezas para darles un orden en el tiempo. Tercero: aplicar un *Mapeado* de las piezas de imagen a notas musicales.
Al ser una correspondencia directa entre la imagen y la música, este trabajo sigue parcialmente la sinestesia y la psicología de las formas y colores aunque este no es su propósito final.

De forma adicional, cabe mencionar dos trabajos interesantes en un ámbito menos académico. El primero es un trabajo realizado por un equipo ruso [25], que permite hacer un dibujo simple (compuesto por un único trazo) y observar cómo se transforma en una melodía distinta dependiendo del trazo realizado y la herramienta con la que se ha realizado. Aunque parte de una base musical estática, a la que van "maquillando" de distintas formas dependiendo de la entrada gráfica, es digno de mención su facilidad de uso y la calidad de los resultados obtenidos.

El segundo de ellos es el realizado por el físico Lauri Gröhn [19], que aplica una serie de reglas basadas en la sinestesia para realizar composiciones algorítmicas postprocesadas. Basa el análisis en un estudio por secciones de la figura de entrada, mediante el cual va generando pequeñas porciones

musicales a medida que recorre distintas regiones de la imagen divididas previamente de forma uniforme.

Por último se destaca el trabajo de A. Pintado [1] en el que hace una investigación de la sinestesia y la percepción de las formas para poder generar ritmos. Analizando una imagen de entrada obtiene sus formas y líneas, los cuales traduce a ritmos que genera usando una relación directa entre el ritmo y la inestabilidad de las figuras. Este trabajo se centra en la línea de la sinestesia, especialmente en las formas de las figuras y los ritmos musicales, dejando de lado los colores y los tonos musicales. Por tanto ha sido una buena fuente de información aunque trate sólo parcialmente el objetivo de este proyecto.

Todos estos algoritmos tienen en común un problema implícito puesto que dada una imagen cada persona espera una correspondencia musical diferente. Por tanto es difícil determinar el fitness o validez de cada algoritmo sabiendo además que la entrada gráfica puede tener infinidad de interpretaciones de todos sus valores disponibles.

1.4. Visión general del documento

El presente documento está estructurado del siguiente modo:

En primer lugar, se ha realizado una primera toma de contacto del proyecto y su contexto en el Capítulo 1, *Introducción*. Este ha comenzado con una descripción del sistema a modo de introducción, seguida de la motivación que ha llevado al desarrollo de este proyecto. Finalmente se ha expuesto el estado del arte y los diferentes estudios sobre la materia realizados hasta el momento.

A continuación, se procede a explicar las diferentes características del sistema, comenzando por un manual de usuario en el Capítulo 2, *Guía de Uso*, donde se expone la forma de usar la herramienta mediante la interfaz gráfica proporcionada. Esta guía abarca también el proceso necesario para instalar la aplicación en las distintas plataformas.

El diseño realizado para el sistema, incluyendo la especificación de los algoritmos usados tanto en el análisis de imagen como en la composición musical, se presenta en el Capítulo 3, *Diseño y Especificación*. En él, se detallan los usuarios a los que está orientada la aplicación, así como la totalidad de los requisitos que detallan las funcionalidades de la aplicación final. En el resto del capítulo se expone la especificación de los algoritmos desarrollados, ya sean pertenecientes a la fase de análisis o a la fase de composición musical.

En el siguiente Capítulo, *Arquitectura*, se detalla de forma exhaustiva la implementación del sistema mediante el estándar UML. Cada sección de este capítulo hace referencia a cada uno de los módulos en los que se ha dividido la arquitectura del sistema, con la excepción de las dos primeras secciones. En ellas, se define la estructura diseñada e implementada de dos piezas claves en la realización de este proyecto: el formato de representación interna de una imagen y el de una pieza musical. Estos formatos y sus correspondientes implementaciones son usados por todos los módulos definidos a continuación. Estos módulos se ordenan en el documento por orden de importancia, siendo el primero de ellos el módulo de composición de música, seguido del módulo de análisis de imagen. Finalmente se tiene en cuenta tanto el módulo que se encarga de unificar los anteriores módulos mencionados como la interfaz gráfica desarrollada para facilitar el uso de la aplicación.

El contenido principal del documento finaliza en el Capítulo 5, *Conclusiones*. En él, se repasa el desarrollo del proyecto y los objetivos cumplidos a lo largo del mismo, dando una visión global del sistema implementado. Posteriormente se procede a explicar los diferentes usos y mercados a los que está destinada la aplicación, así como las posibles ampliaciones que se podrían realizar sobre el sistema si se continuara su desarrollo.

De forma adicional, el documento presenta un apéndice en el que puntualiza las tecnologías usadas en los diferentes módulos que componen la aplicación. Este es el Apéndice A, *Tecnologías utilizadas*.

Capítulo 2

Guía de Uso

2.1. Introducción

Esta sección pretende mostrar una vista general de la aplicación y sus opciones. A lo largo de ella se explicarán todos los módulos del programa con los que el usuario puede interactuar y se detallará la función de determinados mecanismos.

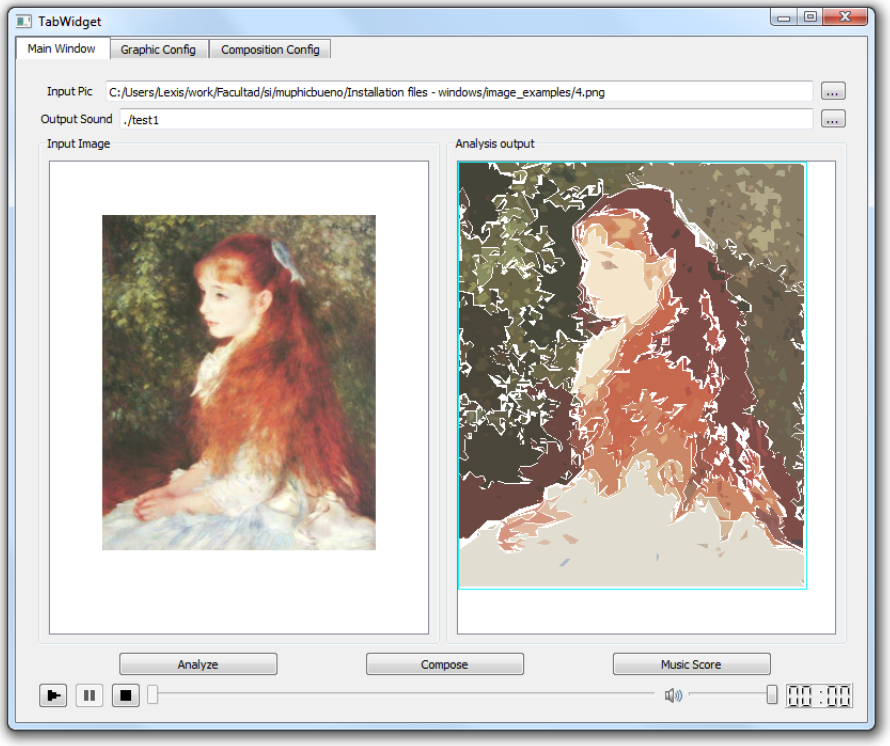


Figura 2.1: Vista general de la aplicación

Como se ve en la Figura 2.1, la aplicación dispone de tres pestañas: *Main Window*, *Graphic Config* y *Composition Config*. La primera dispone de toda la funcionalidad necesaria para lanzar la aplicación, mientras que las otras dos aportan opciones para configurar el comportamiento del análisis y la composición.

De forma general, todo usuario, experto o inexperto, se puede valer de la primera pestaña para usar la composición. Sin embargo, un usuario avanzado con conocimientos musicales y de qué tipos de análisis gráficos se realizan puede navegar por las pestañas restantes para configurar los procesos de composición y análisis de imagen respectivamente.

Se procede por tanto a explicar cada pestaña una por una.

2.2. Ventana principal

Se encarga de la interacción básica con el usuario: muestra los resultados obtenidos y permite lanzar los principales componentes de la aplicación.
Esta compuesta por los siguientes elementos tal y como se puede ver en la Figura 2.2:

Carga de imagen de entrada [G]: permite al usuario elegir la ruta desde donde se cargará la imagen de entrada.

Selección del archivo de audio de salida [H]: permite al usuario elegir la ruta donde se guardará el archivo de salida.

Input Image [I]: en este panel se muestra la imagen elegida para el análisis.

Analysis output [J]: en este panel se muestra el resultado del último análisis realizado pulsando el botón "Analyze".

Botón Analyze [B]: tal y como su nombre indica, Analyze realiza el analisis de la imagen pasada como parámetro de entrada. Tras analizarse, el resultado podrá observarse en el panel *[I]*. Una vez pulsado el botón, y mientras dure el análisis, el botón pasará a mostrar el mensaje "STOP", pudiendo pulsarlo de nuevo para detener el análisis manualmente. Esta opción es bastante útil para imagenes grandes o complejas (con muchas variaciones de color que originan gran cantidad de formas), ya que el análisis puede tardar más de lo esperado y el usuario puede de esta manera detenerlo para luego relanzarlo con otra configuración más rápida.

Botón Compose [A]: realiza la composición musical a partir de los datos

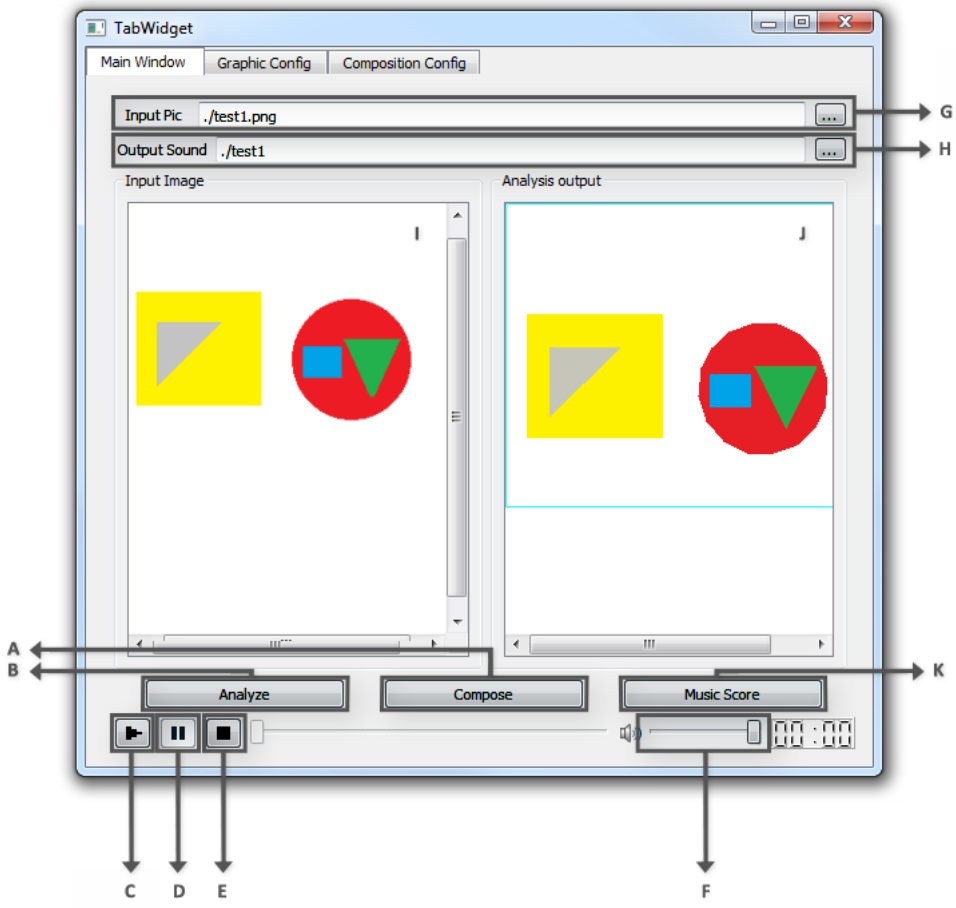


Figura 2.2: Vista general de la pestaña principal de la aplicación

analizados. Una vez compuesta la pieza musical, se podrá escuchar mediante los controles de control de sonido. Una vez pulsado el botón, y mientras dure el análisis, el botón pasará a mostrar el mensaje "STOP', pudiendo pulsarlo de nuevo para detener la composición manualmente.

Botón Music Score [K]: muestra la partitura de la última composición realizada en formato xhtml.

Botón de Inicio de Reproducción [C]: permite al usuario iniciar la reproducción de la pieza musical compuesta previamente.

Botón de Pausa de Reproducción [D]: pausa la pieza musical en reproducción.

Botón de Detención de Reproducción [E]: detiene la reproducción completamente.

Barra de sonido [F]: modifica el volumen de la reproducción en curso.

2.3. Configuración gráfica

En esta pestaña se encuentran todas las opciones relacionadas con el análisis de la imagen. En ella se podrán configurar opciones que hagan que el proceso de análisis varíe en velocidad, precisión o estilo.

De entre los parámetros de configuración, existen los llamados "generales", que siempre aparecen visibles al usuario, y los "específicos", que complementan a los generales y sólo aparecen cuando el contexto lo especifica. En la Figura 2.3 se puede ver detalladamente los parámetros generales.

Tipo de filtrado de imagen [A]: distintos filtros en la imagen producirán diferentes formas de entender las figuras que hay en ellas. De forma un poco más precisa, pero sin llegar a entrar en detalles técnicos, los filtros se encargan de transformar la imagen original a un mapa de bits en blanco y negro, que posteriormente se analizará para detectar polígonos en él. Un buen filtro diferenciará las deseadas superficies de color como manchas blancas independientes.

Selección de ruido [B]: mediante esta barra el usuario puede elegir el tamaño mínimo de las formas por debajo del cual no se considerarán relevantes en el análisis. El valor representa un porcentaje respecto al área total de la imagen, de forma que un valor n de Selección de Ruido determina que todas las formas con área menor a un $n\,\%$ del área total de la imagen se considerarán

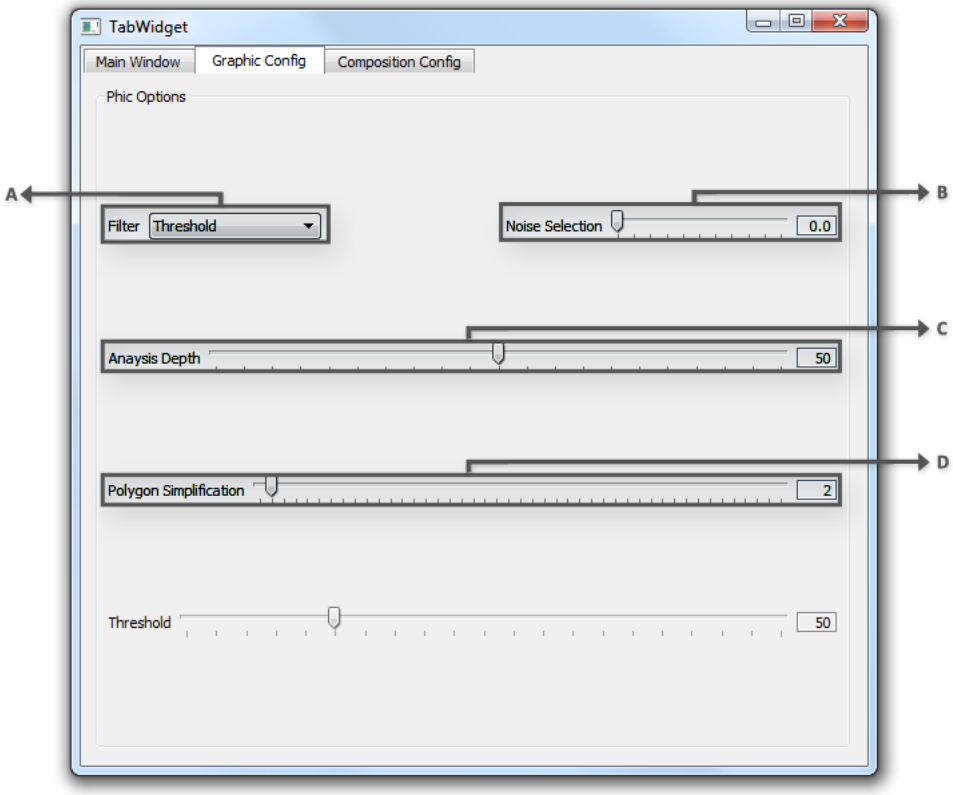


Figura 2.3: Vista general de la pestaña de configuración gráfica

ruido y no serán estudiadas.

Profundidad del análisis [C]: permite variar el nivel de detalle con el que se realizará el análisis, o para ser más precisos: determina el número de píxeles respecto del total que se tienen en cuenta para el anális. Un valor grande de profundidad hará que el análisis se realice examinando la mayor parte de los píxeles de la imagen (se establece como restricción que nunca se examinen más de 1000x1000 píxeles, por cuestiones de optimización), mientras que un valores pequeños aumentarán la velocidad del proceso a costa de tener en cuenta menos píxeles en la imagen.

Simplificación de polígonos [D]: una vez detectadas las formas de color, es necesario que se aproximen a una lista de vértices para reducir el peso de la información sin modificar la carga de la misma. El nivel de fidelidad en la transformación de formas a polígonos se establece con este parámetro: un valor alto determina una gran fidelidad a costa de mayor tiempo de análisis, y viceversa.

Los parámetros "específicos" dependen única y exclusivamente del tipo de filtro seleccionado [A], ya que cada tipo de filtro introduce nuevas opciones que configurar.

Todo filtro en esta aplicación, como ya se comentó anteriormente, realizan la misma función: detectar formas y expresarlas como superficies blancas. A continuación se explica de forma general el funcionamiento de cada filtro y sus parámetros "específicos".

Threshold

Este filtro, contenido en varias librerías de procesado de imágenes (ver [24]), transforma la imagen a escala de grises para luego marcar como blancos todos los píxeles con un valor de gris superiores a un umbral dado, y como negros el resto (una explicación más formal se puede encontrar en [10]).

Los parámetros específicos que usa son:

Valor del umbral: Establece el valor del umbral antes explicado.

Adaptive Threshold

Mientras que el filtro anterior utiliza un mismo umbral para toda la imagen, Adaptive Threshold (ver implementación en [24]) usa un umbral

distinto para diferentes regiones de la imagen, en función de los valores de los píxeles tratados.

Este filtro no usa parámetros específicos, ya que modifica el valor del umbral cada vez que le resulta necesario.

Canny

Utiliza el algoritmo de detección de regiones del mismo nombre, desarrollado en 1986 por John F. Canny (ver implementación en [24], y una descripción detallada en [10]). Una vez encontrados los bordes, establece las figuras que estos delimitan.

Este filtro no usa parámetros específicos.

Hue Division

Se trata de un filtro propio que busca asocia manchas de color con conjuntos de píxeles vecinos con valores RGB dentro de un mismo rango de rojo (R), azul (B) y verde (G). Su parámetro específico es:

Niveles de color: Este parámetro determina el número de intervalos posibles de color que puede haber en cada canal de color, y por tanto determinará la amplitud de cada rango. Un valor igual a 3 indica que un píxel cualquiera puede entrar dentro de 3 intervalos posibles en el canal rojo, otros 3 en el azul y otros 3 en el verde. De esta forma cada píxel entra dentro de una de las $3 * 3 * 3 = 27$ categorías de color, simplificando el análisis. Un valor muy grande de este parámetro hará que haya más categorías de color y que por tanto colores que antes se consideraban parte de una misma forma ahora se consideren como parte de formas independientes con colores más específicos. Un valor muy pequeño hará que se interpreten formas más amplias (incluyendo píxeles más diversos).

Color Threshold

Se trata de un filtro propio que adapta el filtro Threshold para una imagen de 3 canales. En vez de considerar como figuras aquellas agrupaciones de píxeles cuyos valores en escala de grises sean inferiores a un umbral, considera aquellas que en su canal de Matiz (Hue), Saturación (Saturation) y Valor (Value) sean menor que un umbral, originándose en cada canal formas independientes.

Los parámetros específicos de este filtro determinarán el valor de los umbrales de cada canal determinado en formato de imagen HSV.

Valor del umbral de Matiz
Valor del umbral de Saturación
Valor del umbral de Valor

Para dar una vista general de las características y formas de trabajar de cada filtro de análisis, se muestran en la Figura 2.4 los resultados que proporciona cada uno de ellos para una misma imagen de entrada. Todos ellos han usado como valores de entrada comunes los siguientes:

Noise Selection = 0
Analysis Depth = 100
Polygon Simplification = 1

Como se puede observar, el filtro Hue Division es el más fiel de los cinco, captando en la representación de la imagen la mayor parte de la información que había en la imagen de entrada.

2.4. Configuración de composición

Esta pestaña contiene todos los parámetros que se pueden configurar relativos a la composición algorítmica. Estos, como muestra la Figura 2.5 son:

Sistema de color[A]: permite alternar entre las diferentes relaciones tono-color explicadas en la Sección 3.5.

Algoritmo de composición para la primera voz[B]: con esta opción, el usuario puede elegir, de entre los distintos algoritmos de composición, el que se usará para que generar la melodía principal.

Algoritmo de composición para la segunda voz[G]: con esta opción, el usuario puede elegir, de entre los distintos algoritmos de composición, el que se usará para que generar la segunda melodía principal.

Algoritmo de composición para la tercera voz[D]: con esta opción, el usuario puede elegir, de entre los distintos algoritmos de composición del bajo, el que se usará para que generar el bajo armónico.

Algoritmo de composición para la cuarta voz[I]: con esta opción, el usuario puede elegir, de entre los distintos algoritmos de composición de ritmos, el

Imagen de Entrada

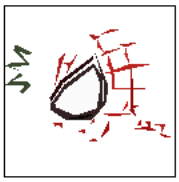
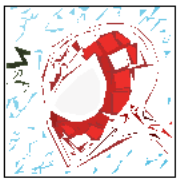
Threshold Valor del umbral: 83	
Adaptive	
Canny	
Hue Division Niveles de color: 3	
Color Threshold Valor del umbral de Matiz: 84 Valor del umbral de Saturación: 183 Valor del umbral de Valor: 71	

Figura 2.4: Tabla con las distintas salidas de cada filtro de análisis

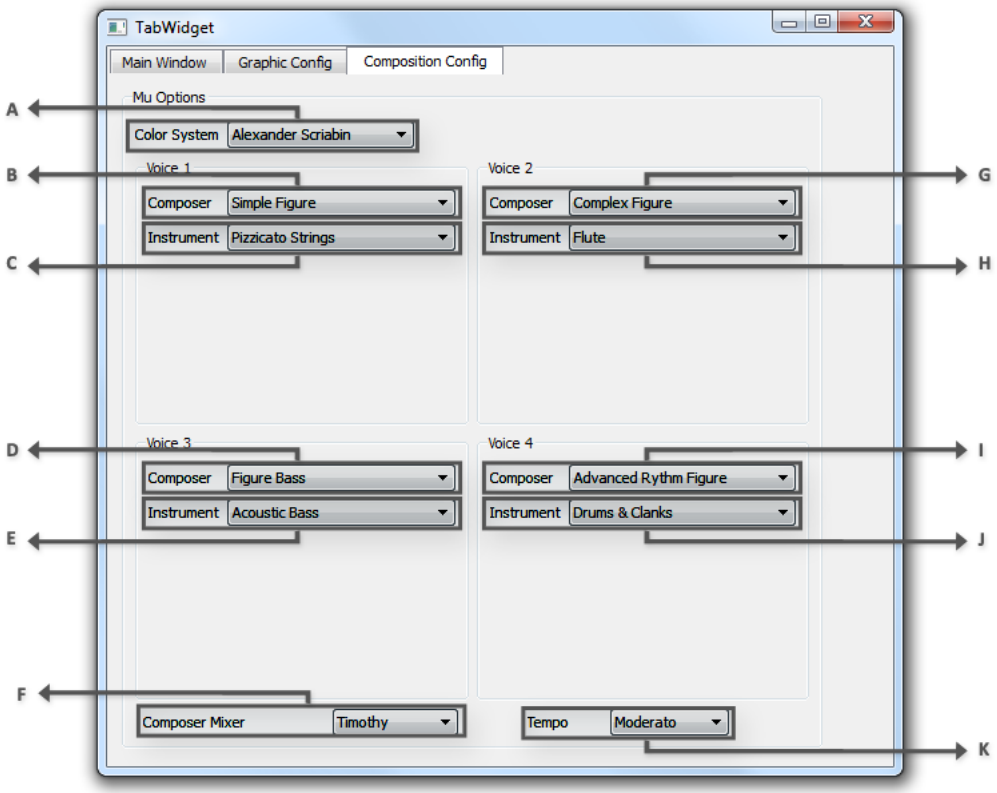


Figura 2.5: Vista general de la pestaña de configuración gráfica

que se usará para que generar el ritmo de la pieza musical final.

Selección de instrumentos [C], [H], [E] y [J]: permiten seleccionar los instrumentos con los que sonaran las distintas voces.

Algoritmo de composición general[F]: determina qué algoritmo se usará para recorrer la imagen y aplicar los algoritmos de las distintas voces. Se explica con detalle en la Sección 3.5.

Tempo[K]: como su nombre indica, su valor determina el tempo con el que sonará la pieza musical compuesta.

2.5. Requisitos de instalación

Muphic es una aplicación multiplaforma con soporte para Linux y Windows de 32 y 64 bits, siendo en ambos una aplicación portable. Para finalizar este capítulo, se detallará a continuación las peculiaridades de ejecución en cada plataforma.

2.5.1. Windows

Para ejecutar la aplicación basta con descomprimir el archivo descargado y ejecutar la aplicación.

2.5.2. Linux

Para utilizar la aplicación en cualquier distribución de Linux, habrá que descomprimir el archivo y generar los ejecutables con la herramienta make. Ciertos elementos del sistema (la interfaz gráfica y los controles de reproducción de sonido) requieren que el usuario tenga instaladas ciertas librerías que no se incluyen en la descarga de la aplicación. Estos paquetes son:

- Librerías de qt versión 4.7.4, disponibles en su página web (ver [28]).
- *libphonon-dev (multimedia framework from KDE-development files)*
- *libphonon4 (multimedia framework from KDE-core library)*
- *phonon (multimedia framework from KDE-metapackage)*
- *phonon-backend-gstreamer (Phonon GStreamer 0.10.x backend)*
- *libqtscript4-phonon (Qt Script bindings for the Qt 4 Phonon library)*

Si el paquete *phonon-backend-gstreamer*, por cualquier razón, fuera el origen de algún fallo; el usario puede elegir probar con otros tipos de backend, como por ejemplo *phonon-backend-vlc*.

Capítulo 3

Diseño y Especificación

3.1. Introducción

Este capítulo se centrará en el análisis y especificación realizados sobre el sistema, previos a la implementación. Para ello, se detallará la captura de requisitos que definen la funcionalidad del sistema, para luego analizar con detalle los algoritmos diseñados para llevar a cabo las dos funciones más importantes de la aplicación: el análisis de imágenes y la composición de piezas musicales.

3.2. Usuarios

Los usuarios que se tienen en cuenta para delimitar los requisitos son:

- Usuario: individuos con un conocimiento mínimo o nulo sobre música y ofimática que hayan leído o se les haya explicado el funcionamiento de la aplicación. Pueden requerir cierto período de entrenamiento para acostumbrarse a las distintas opciones y funciones que ofrece la aplicación.

- Desarrollador: individuos con un conocimiento amplio de programación y con conocimiento básico o avanzado de música. Podrán expandir la aplicación de forma sencilla añadiendo nuevos algoritmos de composición.

3.3. Requisitos

Esta sección describirá las funcionalidades del sistema: su comportamiento esperado así como sus restricciones de rendimiento y capacidad. Aunque el proyecto ha sido desarrollado sin ningún cliente específico, se han elaborado requisitos basados en la visión de la aplicación como conjunto, de forma que

tanto la planificación como el desarrollo se pudieran abordar más fácilmente.

Una vez determinandos qué usuarios harán uso de la aplicación, se procederá a determinar cuáles son los requisitos funcionales y los no funcionales.

3.3.1. Requisitos funcionales

En el caso de la interfaz gráfica y la estructura general de la aplicación, la interfaz deberá ser capaz de:

- Lanzar tanto el módulo de análisis como el módulo de composición de forma independiente.
- Interpretar archivos de imagen del formato .png y mostrarlos por pantalla.
- Interpretar archivos de audio del formato .wav y permitir funciones básicas de reproducción sobre los mismos.
- Generar un archivo de configuración interpretable tanto por la aplicación de análisis y composición a partir de los parámetros de entrada establecidos por el usuario.
- Mostrar la partitura de la música compuesta tras haber sido generada mediante un programa externo o por sí misma.
- Mostrar la salida del proceso de análisis de imagen (mediante un programa externo o por sí misma), en forma de polígonos con color plano distribuidos unos dentro de otros.
- Controlar posibles errores cometidos por el usuario, surgidos al tocar parámetros no disponibles para la configuración seleccionada, a su vez deberá de limitar las acciones del usuario a la hora de modificar dichos parámetros según las configuraciones elegidas.
- Prestar la opción de detener módulos lanzados para que el usuario pueda pararlos, si estos no han acabado aún.

En el caso del módulo de análisis de imágenes:

- El analizador deberá, dado un archivo de configuración que determine como realizar el análisis y un archivo de imagen que analizar, producir un XML con los siguientes datos:
  - Los vértices de las figuras que componen la imagen de entrada y el número total de vértices por figura.
  - Los colores que contienen dichas figuras en formato RGB

- Una estructura jerarquizada de figuras al igual que se presentan en la imagen, es decir, si una figura está dentro de otra en el XML se verá reflejado siendo la segunda figura un elemento incluido en la primera.

La estructura detallada de este documento se mostrará en la Sección 4.2.

En el caso del módulo de análisis de composición, el compositor deberá ser capaz de:

- Generar un archivo abc interpretable por cualquier programa que pueda recibir como entrada dicho tipo de archivos, dado un archivo de configuración y un archivo con los resultados del análisis correctamente estructurados.
- Comunicarse con programas externos para que transformen el archivo abc generado a los formatos de audio midi y wav.

3.3.2. Requisitos no funcionales

En el caso de la interfaz gráfica y la estructura general de la aplicación:

- La aplicación debe ser multiplataforma, pudiendo funcionar tanto en Windows como en sistemas Unix.
- La aplicación debe dar resultados en un intervalo de tiempo pequeño para una configuración de parámetros determinada.
- La aplicación no debe instalar ni alterar los registros Sistema Operativo del usuario, ya que será portable.
- La implementación del sistema debe ser suficientemente general como para poder reimplementarse en otras plataformas.

En el caso de la interfaz gráfica, se da que:

- La aplicación debe ser accesible al usuario, facilitando al usuario la entrada de datos de configuración o imágenes que desee analizar.

Para el módulo analizador de imágenes:

- El módulo debe devolver, para al menos una configuración dada de parámetros, un análisis que se asemeje a "primera vista" a la imagen original.

Para el módulo de composición;

- El módulo, mediante su estructura interna, será fácilmente ampliable por un usuario desarrollador, de forma que pueda añadir cómodamente algoritmos de composición nuevos.

- El compositor principal de la aplicación deberá de ser capaz de generar una música que, aunque no sea "interesante", nunca "no suene mal". En ambos casos los resultados en ultima instancia dependerán de la opinión subjetiva del usuario

Para finalizar, resta decir que, por comodidad para el lector, se resumirán los casos de uso en dos principales y se procederá a detallar su especificación: realizar un análisis de imágenes y componer una pieza musical.

3.4. Algoritmos de Análisis

Esta sección explicará con detalle los algoritmos de análisis de imágenes que se han desarrollado a lo largo del proceso de desarrollo. El objetivo de estos análisis es obtener, a partir de una imagen dada, información de los polígonos presentes tal y como se explica en la Sección 4.2. Esta información abarca tanto el color y vértices de cada polígono como qué polígonos se encuentran dentro de cuáles.

Puesto que interesa la eficiencia temporal del proceso de análisis para aumentar la comodidad de la experiencia de usuario, será necesario que los procesos de análisis no consuman mucho tiempo o, al menos, que permitan aumentar su velocidad mediante configuración externa. Es por ello que cada proceso de análisis contendrá parámetros mediante los cuales se podrá modificar la velocidad del análisis a costa de menor precisión en el mismo.

Para el diseño e implementación de los mismos se ha hecho uso de la librería multiplataforma de procesado de imágenes OpenCV [24]). Esta librería, desarrollada por Intel y orientada a la visión por computador, ofrece multitud de funciones centradas principalmente en el procesado de imágenes en tiempo real.

Aunque en los análisis no se haga uso de sus funciones en tiempo real, sí que se usarán sus subrutinas de procesado de imágenes para conseguir algoritmos que funcionen a velocidades deseadas. Estas subrutinas, que se detallarán más adelante, permiten detectar contornos y aproximarlos a polígonos, así como usar y operar imágenes de forma cómoda.

Para realizar el análisis ayudándondos de OpenCV, el proceso que se lleva a cabo consiste en:

- **Detección de formas:** la imagen dada se transforma en una sucesión de imágenes binarias (únicamente en blanco y negro), donde las manchas blancas representan figuras independientes. Dado que varias figuras pueden superponerse (una figura contiene a otra), si se devolviera una única imagen binaria, sólo podría representarse una de las figuras (la que envuelve a las demás, ya que todas se pintaría como superficies blancas por ser figuras independientes), perdiendo información en el análisis. Es por ello que, siempre que resulte necesario, se devolverán varias imagenes de salida en esta fase del proceso, para asegurar que todas las figuras presentes se tienen en cuenta.

  En numerosas ocasiones se referirá a esta fase como "filtrado de la imagen" debido a que, para llevarse a cabo, se aplicarán filtros de procesado de imágenes sobre la imagen de entrada.

- **Detección de contornos y aproximación a polígonos:** por cada mancha blancha, se recorre el perímetro en el mapa de bits y se almacena en una lista de puntos. Para finalizar, cada contorno se aproxima a un polígono y se analiza por separado, con el fin de desechar información no deseada y calcular datos relevantes.

Mientras que para la última fase existen rutinas de openCV que realizan casi toda la funcionalidad deseada, la primera fase requiere un estudio más exhaustivo.

3.4.1. Detección de formas

Para llevar a cabo esta primera fase se han probado y evaluado cinco métodos distintos. Se detallarán a continuación todos ellos.

3.4.1.1. Threshold

Consiste en aplicar un filtro de paso alta con respecto a un umbral de gris dado como entrada. Para ello, la imagen se convierte de 3 canales (rojo, verde y azul) a una de 1 único canal (escala de grises). Posteriormente se le aplica la siguiente transformación a cada píxel de la imagen:

$$pixel(x,y) = \begin{cases} MAXVAL & \text{cuando } pixel(x,y) > umbral \\ 0 & \text{e.o.c.} \end{cases}$$

donde $MAXVAL$ representa el máximo valor que puede tener un pixel en un canal determinado (será 255 o 1, dependiendo de la representación de la imagen).

Este filtro produce una única imagen de salida donde las zonas blancas conexas se interpretan como figuras independientes. Es por ello que, si dos manchas de color se comportan de igual manera ante el filtro (porque su color predominante es inferior o superior al umbral en ambas manchas), y se encuentran superpuestas una respecto de la otra, en la salida aparecerán como una única mancha blanca conexa, por lo que se considerará como una única figura (Figura 3.1.)

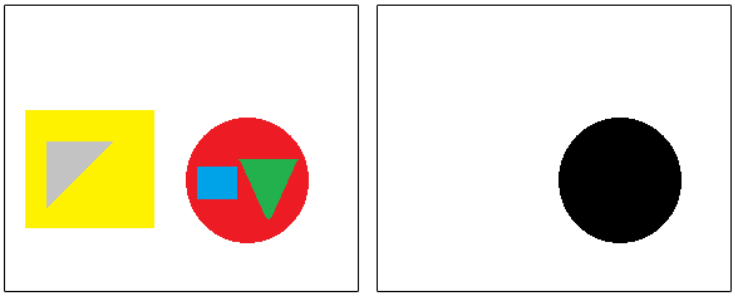


Figura 3.1: Ejemplo de filtrado threshold con umbral a 100

Si, dado un caso similar en el que dos figuras estén superpuestas, pero que se comporten de forma distinta ante la aplicación del filtro (una se encuentra por debajo del umbral y otra por encima), entonces sí que se interpretará como dos figuras distintas (Figura 3.2). Esto se produce debido a que, como se explica en la Sección 3.4.2, los contornos se buscan tanto en los perímetros externos de las figuras como en los internos.

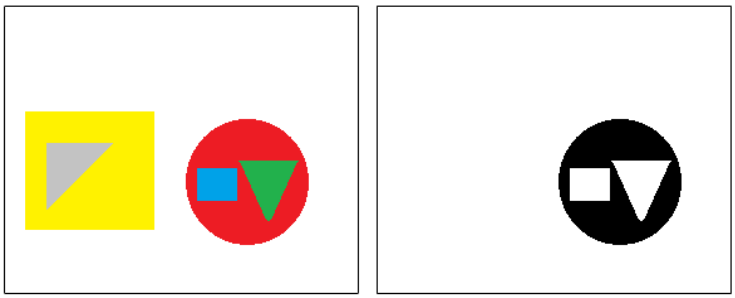


Figura 3.2: Ejemplo de filtrado threshold con umbral a 150

Se trata de un algoritmo muy sencillo y rápido, pero en el que él éxito de los resultados depende tanto del umbral de entrada como de la imagen a analizar.

3.4.1.2. Adaptive Threshold

Se trata de una modificación del algoritmo anterior, donde el valor del umbral varía en cada región de la imagen. Este algoritmo también devuelve sólo una imagen binaria, pero aplica sobre ella un filtro más sofisticado que repara la mayor parte de los fallos del filtro anterior:

$$pixel(x,y) = \begin{cases} MAXVAL & \text{cuando } pixel(x,y) > T(x,y) \\ 0 & \text{e.o.c.} \end{cases}$$

El umbral ya no es constante, sino que se convierte en el término $T(x,y)$, que depende tanto de x como de y. Existen muchas formas de calcular este término, pero en este caso se calcula mediante la suma ponderada de los píxeles vecinos al (x,y) dentro de un bloque de 3×3 píxeles donde el centro es el pixel a transformar. Tras haber realizado la suma, se le resta una constante teórica c, que especifica cuánto debe alejarse el valor de un píxel dado respecto al de la media calculada para que se considere que ha sobrepasado el umbral. En la implementación de este filtro se usará 3 como valor de esta constante, ya que proporciona resultados acordes a los intereses de este sistema.

Con esta especificación, los resultados obtenidos son los mostrados en la Figura 3.3. Como se puede observar, el resultado del filtro no depende de ningún valor y es más fiel que la mayoría de los resultados del threshold para figuras con este grado de simplicidad.

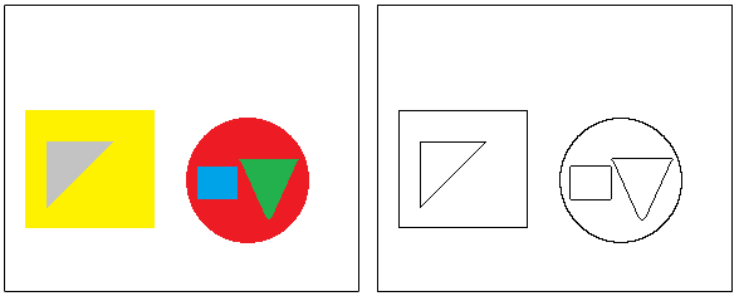


Figura 3.3: Ejemplo de filtrado adaptive threshold

3.4.1.3. Canny

Usando como base el algoritmo de detección de bordes creado por John F. Canny en 1986 (cuya especificación se puede ver en [10]), este filtro aborda el problema de la detección de figuras de una forma diferente: en vez de detectar el contenido de la figura, detecta el perímetro de las mismas.

Usando el algoritmo de canny, que también devuelve una única imagen, se obtiene el resultado mostrado en la Figura 3.4.

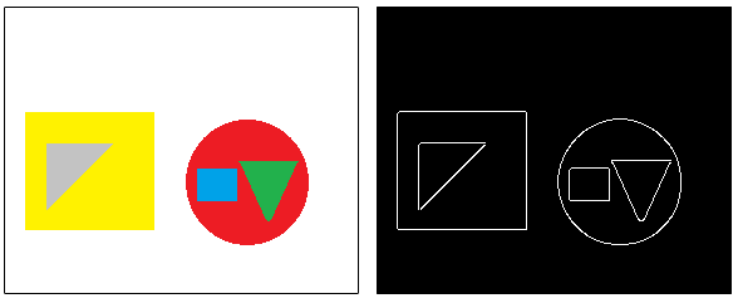


Figura 3.4: Ejemplo de filtrado canny

Esta nueva aproximación produce resultados satisfactorios, pero origina nuevos problemas: el perímetro detectado en el algoritmo puede no estar cerrado, provocando que la figura resultante no se represente de forma correcta en la estructura de salida.

3.4.1.4. Hue Division

Este algoritmo se basa en un concepto más simple: buscar manchas de color como agrupaciones de píxeles vecinos con colores "parecidos". Es decir, se dividen los tres canales de color en diferentes rangos, y se agrupan los píxeles contiguos cuyos valores de color caigan dentro de un mismo rango en todos los canales.

Para establecer los rangos, se divide cada canal de color (rojo, verde y azul) en un número n de intervalos. El parámetro n es de entrada y común para los tres canales. De esta forma, se recorre la imagen por cada intervalo existente de colores, devolviendo las figuras encontradas en cada intervalo en imágenes de salida distintas.

Por ejemplo, si n es 3, quiere decir que por cada canal se establecerán 3 intervalos, en total cada color podrá caer dentro de uno de los $3 * 3 * 3 = 27$ intervalos. Por cada intervalo se devuelve una imagen con las figuras presentes en él, comprobando antes que no esté vacía (no existen figuras con colores dentro de ese intervalo). Se puede ver un ejemplo en la Figura 3.5.

3.4.1.5. Color Threshold

Se trata de una expansión del filtro Threshold para intentar resolver uno de sus principales problemas: dado que el filtro Threshold convierte la

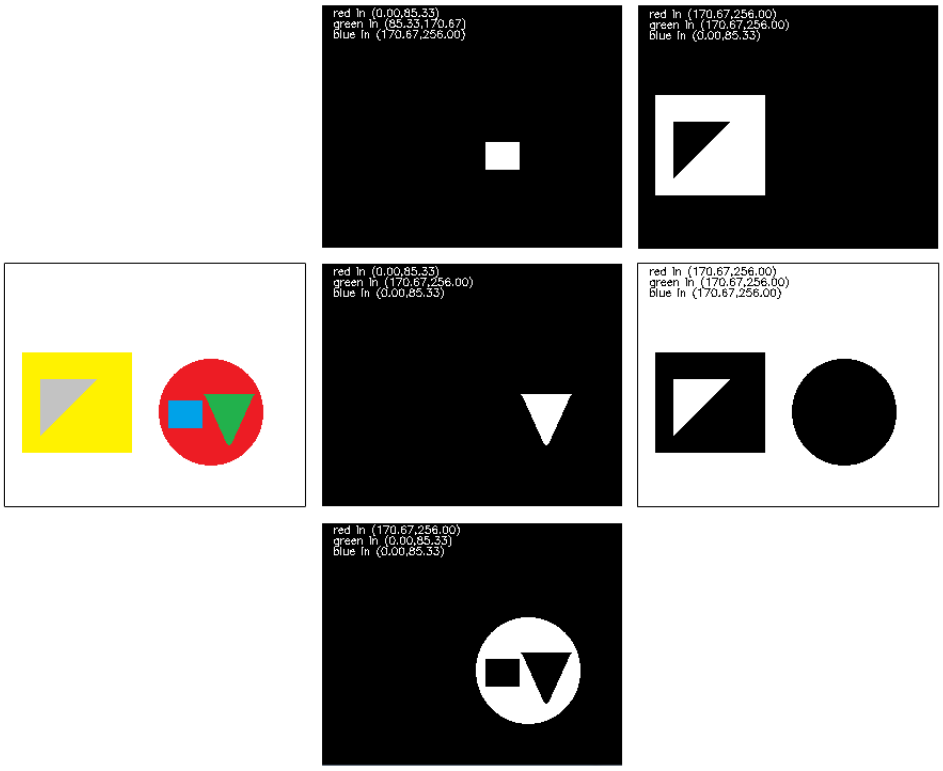


Figura 3.5: Ejemplo de filtrado hue division

imagen en escala de grises para buscar píxeles que superen el umbral, puede darse el caso de que dos figuras con colores distintos se conviertan en el mismo tono de gris al convertirse a escala de grises. Este problema es independiente del método de transformación de RGB a escala de grises, e inherente al proceso en sí: al pasar de tres canales de color a un único canal, necesariamente varios valores de entrada van a coincidir en el mismo valor de salida (la función es sobreyectiva y el dominio de entrada es más grande que el de salida).

Para resolver ese fallo, se realiza un filtro Threshold con tres umbrales distintos, uno para cada canal de color. Dado que estos umbrales van a poder ser manipulados por el usuario, se transformará previamente la imagen de RGB (rojo, verde y azul) a HSV (matiz, saturación y valor), que es un modelo más intuitivo para el ojo humano.

Se pueden ver diferentes resultados de este filtro en las Figuras 3.6 y 3.7.

3.4.2. Detección de contornos y aproximación a polígonos

Una vez se tienen la lista de imágenes binarias con las figuras localizadas, se debe proceder a representarlas en forma de polígonos. Para ello, primero se detectarán los contornos y posteriormente se aproximarán los mismos a polígonos.

Para la primera parte (detección de contornos en base a una imagen binaria) se hará uso de la función *cvFindContours* de OpenCV, cuyo resultado se puede ver en la Figura 3.8.

El ejemplo mostrado se ha basado en una imagen filtrada mediante el algoritmo Adaptive Threshold. Como se puede observar, una misma figura (el círculo) ha dado lugar a varios contornos. Este es un problema inherente al enfoque usado de detección de formas que se tratará más adelante.

Posteriormente se usa otra función de OpenCV, *cvApproxPoly*, para aproximar los contornos detectados a secuencias cerradas de vértices que se podrán tratar con facilidad. El usuario puede decidir el grado de aproximación (de más simple a más fiel). Un ejemplo del resultado de esta función se ve en la Figura 3.9

Teniendo los polígonos listados, sólo resta organizar la información obtenida y guardarla en un archivo XML.

Para ello se procede a calcular el área de cada polígono. Si es menor que la especificada como ruido por el usuario (ver Guía de Usuario, Capítu-

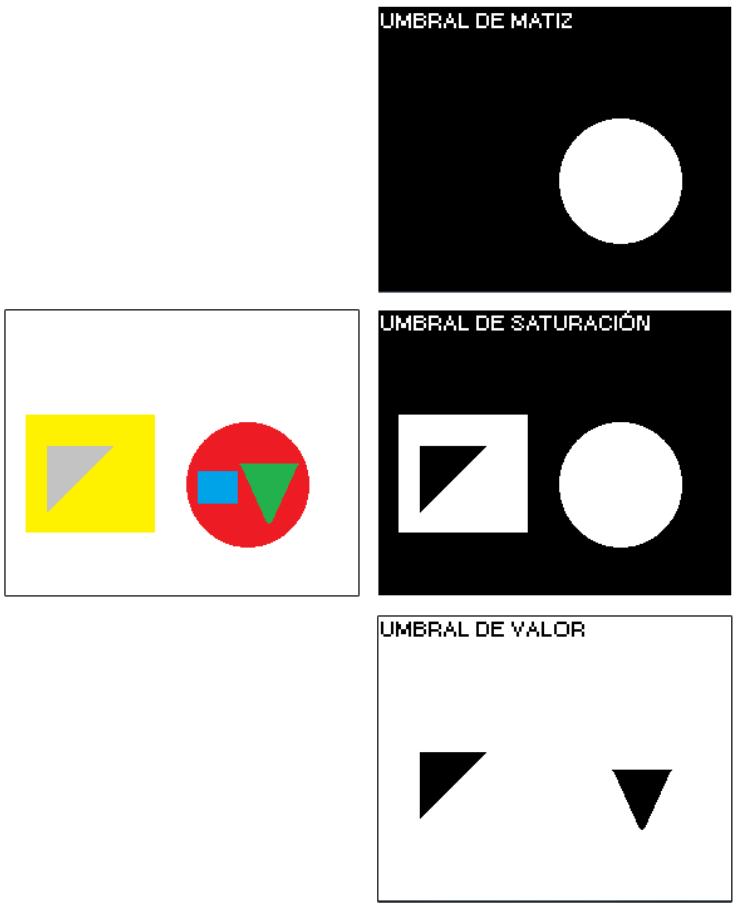


Figura 3.6: Ejemplo de filtrado color threshold: $Umbral_H = 55, Umbral_S = 150, Umbral_V = 205$

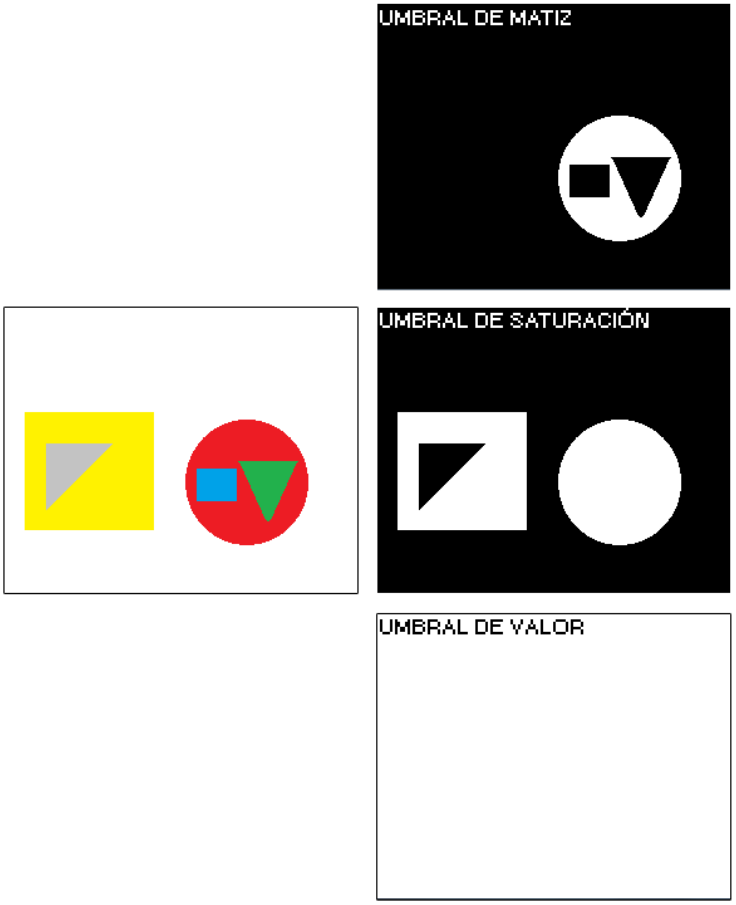


Figura 3.7: Ejemplo de filtrado color threshold: $Umbral_H = 105, Umbral_S = 100, Umbral_V = 125$

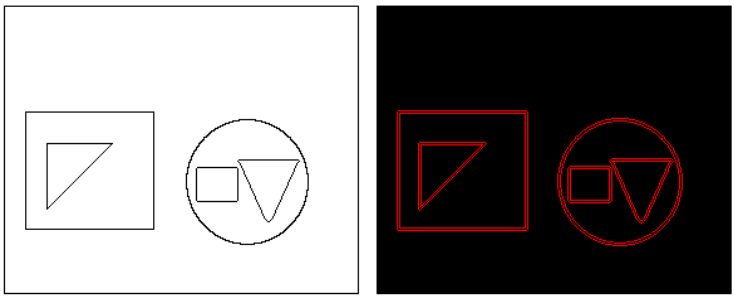


Figura 3.8: Detección de contornos mediante OpenCV

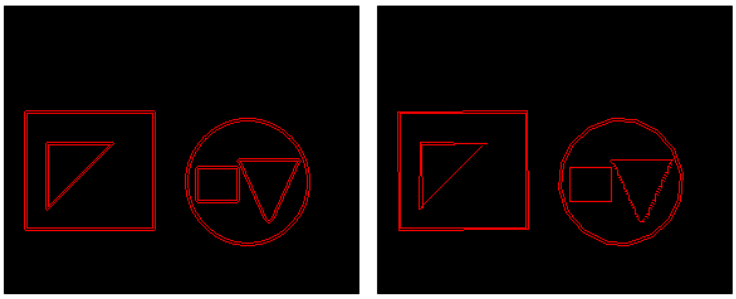


Figura 3.9: Aproximación de polígonos mediante OpenCV

lo 2), entonces se desechará. Si no, se almacenará dentro de la estructura de figuras a archivar en XML. El color de cada polígono, por otro lado, se calculará haciendo la media del color de todos los píxeles que contiene.

Para acabar, es necesario organizar los polígonos resultantes en estructura de árbol, de forma que los polígonos que se encuentran dentro de otro polígono sean sus hijos en el árbol resultante. Es aquí donde surge el principal problema del análisis de imágenes: **los polígonos repetidos**.

Como ya se ha comentado anteriormente, una misma mancha de color puede originar varios contornos, debido a que la figura se ha detectado dos veces en el proceso de filtrado (por ejemplo, si se encuentra dentro de dos umbrales en el filtro Color threshold) o a que el proceso de detección de contornos ha devuelto dos contornos similares a la figura dada, por particularidades internas del algoritmo de OpenCV.

Esto va a originar que, irremediablemente, haya ocasiones en las que aparecerán varios contornos (y posteriormente polígonos) similares. Se procede pues, antes de continuar el análisis, a eliminar los polígonos repetidos. Para ello, se comprueba primero que tengan el mismo número de vértices y que los rectángulos mínimos que los contienen tengan área y posición similares. Podría darse el caso de que un polígono A, teniendo más vértices que otro polígono B, se considerase idéntico a él, si se da uno de los dos siguientes casos: o bien un mismo vértice de B aproxima varios en A o bien en uno de los dos polígonos existe un "vértice falso" (es decir, el vértice intermedio de tres vértices alineados, que no aporta ninguna información nueva a la figura) en alguna arista. Sin embargo, la librería OpenCV nunca devuelve en un mismo polígono dos vértices lo suficientemente cercanos como para que sean considerados "idénticos" según el criterio establecido, por lo que no se estudiará ese caso. De la misma manera, la librería no genera nunca polígonos con "falsos vértices", por lo que se puede asegurar que dos polígonos

no serán considerados idénticos (también serán denominados "similares") si tienen distinto número de vértices.

Posteriormente, se procede a calcular si son efectivamente similares o no, es decir, si todos los vértices de un polígono tienen un vértice idéntico en el otro polígono y éstos pares siguen el "orden" de los polígonos (por ejemplo, si un vértice v del polígono A es idéntico al vértice v' del polígono B, entonces uno de los vértices contiguos a v debe ser idéntico a uno de los vértices a v' para que sean considerados idénticos). De una manera más formal:

Sean dos polígonos A y B. Los polígonos serán similares, o considerados idénticos, si existe una disposición de vértices en A, $V_A = [v_1, v_2, ..., v_n]$, y otra en B, $V_B = [v'_1, v'_2, ..., v'_n]$ que asumen vértices contiguos tomados en sentido horario (o antihorario), tales que:

$$\forall k, 0 < k < N, dist(v_k, v'_k) < \epsilon$$

Donde $dist(p, q)$ es la función que calcula la distancia entre dos puntos del espacio bidimensional. La constante ϵ determina lo cerca que deben estar los vértices entre sí par considerarse idénticos.

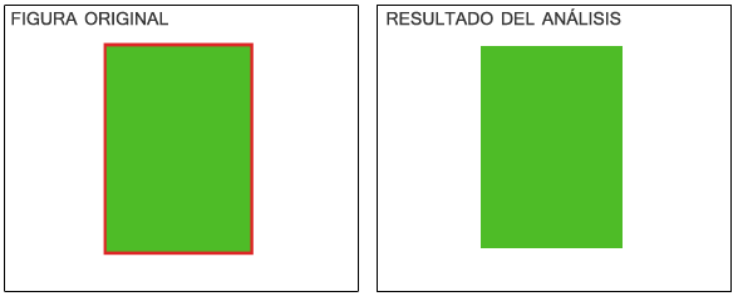


Figura 3.10: Caso de polígonos similares que no deben ser eliminados

Sin embargo, puede darse el caso en que dos polígonos similares sean distintos y diferenciarse principalmente por el color (uno es la silueta del otro, ver Figura 3.10. Dado que nunca se trabaja con el color, y éste se calcula *a posteriori* (haciendo una media del color de los píxeles que contiene el polígono), no se podrá usar el color de cada figura para distinguirlas, ya que sería bastante parecido (ambas figuras encierran aproximadamente los mismos píxeles), por lo que se carece de la información necesaria debido a la naturaleza del proceso realizado. A pesar de todo, este caso es muy particular y no afecta al correcto desarrollo de la aplicación, por lo que no supone un problema en el análisis.

Una vez eliminadas las figuras repetidas, se puede estructurar los polígonos en árboles, teniendo en cuenta que un polígono está dentro de otro cuando todos sus vértices están dentro de él. El rectángulo principal (que delimita el lienzo de la imagen) se elimina del análisis si ha sido detectado como polígono, ya que se encuentra en todas las imágenes y no aporta ninguna información relevante.

El color calculado para cada polígono tenía en cuenta todos los píxeles que este contenía. Sin embargo, los polígonos que contienen otros polígonos han introducido información de más a la hora de calcular este color, ya que dentro de los polígonos que contienen se han tenido en cuenta los de sus hijos también. Es ahora que se conoce la estructura arbórea de la imagen cuando se pueden resolver estos detalles, recalculando el color de los polígonos que contienen a otros de forma que no se tenga en cuenta el color de sus polígonos internos.

Una vez realizadas estas operaciones, los polígonos detectados en el análisis quedan tal y como se muestra en la Figura 3.11.

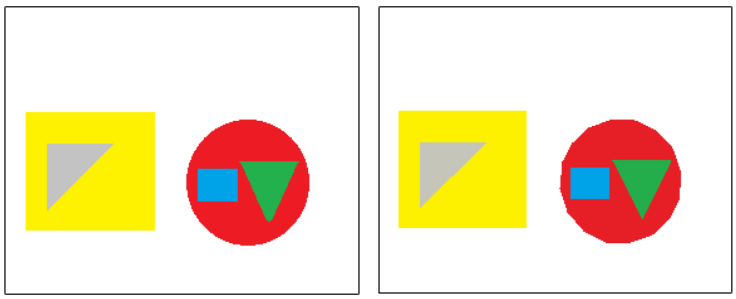


Figura 3.11: Resultado final del análisis

Finalmente, se guarda el contenido del análisis en un fichero XML, dando por terminado este proceso.

3.5. Algoritmos de Composición

En esta sección se explicarán de forma exhaustiva los procesos que se siguen para componer música a partir de los datos que proporciona el análisis de imagen.

Para generar la música se separa el proceso de composición en diferentes voces, de tal modo que la unión de todas las voces da como resultado una pieza musical completa. En el estado actual del proyecto se han identifi-

cado cuatro voces diferentes, cada una con un papel especial dentro de la composición.

Cada voz sigue un proceso de composición independiente, aunque concuerdan en algunos aspectos para poder realizar una unión correcta de todas ellas, como por ejemplo, la duración total de las mismas. Todos los compositores necesitan recibir una figura, en la que se van a basar para componer, y una duración que será el tiempo que se prolonga el trozo de música a componer.

A continuación se explicará el funcionamiento de los algoritmos de cada voz y al final se presentará el algoritmo que se encarga de ir usando cada una de las voces para luego juntarlas y producir una pieza musical.

3.5.1. 1ª Voz: Melodía Principal

Normalmente en una composición suele haber una voz que destaca sobre las demás, que es la que el oído reconoce primero. Esta voz reproduce la melodía. Para poder crear la melodía, se va componer a nivel de notas y la relación entre ellas. Una nota se separa en dos dimensiones: duración y tono. Siguiendo esta separación se ha dividido el proceso de composición en cuatro fases: procesar la figura, calcular duraciones, obtener tonos y sintetizar o unir las duraciones con los tonos.
Para la voz principal se ha dejado la posibilidad de elegir entre dos versiones diferentes del compositor. La primera que se desarrolló tiene algunos pequeños defectos que luego en la segunda versión del compositor se corrigieron.

La primera versión (ComposerFigMelody):

- Procesar la figura: el algoritmo de entrada dispone de una figura, en la que se va a basar más adelante para componer. Antes hace falta un proceso de normalización para que todas las figuras puedan ser interpretadas correctamente. Para ello se extraen los vértices ordenados de la figura de tal forma que el primer vértice sea el más cercano a la parte superior (en caso de tener varios candidatos se escoge el que esté más a la izquierda) de la imagen y los siguientes se recorren en sentido horario. Además, durante el proceso, se irá calculando la longitud de los lados de la figura.

- Calcular duraciones: para poder obtener las duraciones de las futuras notas se hace una correpondencia directa con la longitud de los lados. De este modo la duración total del trozo de música a componer es la suma de todas la longitudes y para cada lado se calcula cuánta duración le corresponde. Obviamente se hace una aproximación, ya

que los valores de duración de las notas son limitados respecto al rango de valores de los números reales.

- Obtener tonos: se hallan los tonos a partir de los ángulos de la figura. En otros trabajos se han explorado diferentes maneras de conseguirlo ("Croma"[7] o [35]). El proceso de cálculo de tono usado se basa en la idea de estabilidad e inestabilidad que se desarrolla en el trabajo de A. Pintado [1]. Una figura es estable cuanto más suave sea su contorno, es decir, cuantos menos picos, ángulos diferentes e irregularidades tenga. A. Pintado usa esta cualidad para generar ritmos a partir de figuras y líneas, aquí se va a usar para calcular tonos, de tal modo que un ángulo más abierto implica mayor cambio en el tono y viceversa. La primera nota desde la cual se parte para hacer los cambios de tono es la nota correspondiente al color de la figura, la cual se saca a través del sistema de colores-notas utilizado.

- Sintetizar las notas: se combinan las duraciones con los tonos obtenidos en la anterior fase teniendo en cuenta que cada vértice tiene asociado un tono y una duración. Al final se obtiene un trozo de música.

La segunda versión (ComposerFigMelody2):

- Procesar la figura: esta fase será idéntica a la realizada por la primera versión.

- Calcular duraciones: respecto a la anterior versión se simplifica la correspondencia entre longitud y duración usando sobre todo duraciones simples y evitando las duraciones compuestas por su irregularidad musico-temporal. Puede parecer que se pierde la equivalencia fiel entre la longitud de los lados y la duración de las notas, ya que para ciertos rangos de longitudes diferentes se calculan duraciones iguales. Tras varias pruebas con el primer algoritmo (ComposerFigMelody), usando la correspondencia directa entre duraciones y longitudes, en varias ocasiones se pierde el ritmo musical dentro de la pieza generada. Con esta modificación se intenta subsanar este problema.

- Obtener tonos: respecto al anterior algoritmo, en este apartado se sigue sacando una correlación entre los ángulos y los tonos, pero se va a usar un proceso diferente. Para calcular el tono se tendrá en cuenta la diferencia de ángulos entre vértices adyacentes. De esta manera a mayor diferencia de ángulos, mayor será el cambio en el tono.
  De forma experimental y tras varias pruebas, se ha obtenido una correspondencia entre la variación de ángulos y la cantidad que deberá variar el nuevo tono respecto al anterior (Figura 3.12).

  De este modo si la diferencia de ángulos es menor a 10^o se sigue usando el mismo tono, si la diferencia está entre 10^o y 30^o se usa el tono

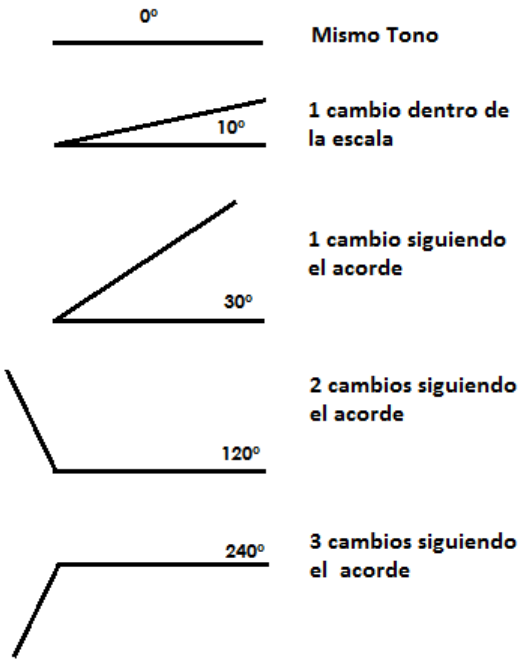


Figura 3.12: Tabla correspondencias entre ángulos y cambios de tono

adyacente dentro de la escala. Si está entre 30º y 120º el primer tono más cercano del acorde, entre 120º y 240º el segundo tono más cercano del acorde y hasta 360º el tercer tono más cercano del acorde. El acorde que se usa es el acorde fundamental del tono que da el color de la figura. Esta relación se basa en la asociación que tiene Scriabin con los colores [12]: él hace corresponder un color a un tono y a su acorde fundamental sin distinguir entre acorde mayor o menor para cada una de las notas de la escala cromática.

- Sintetizar las notas: mismo procedimiento que en la anterior versión.

Un ejemplo completo de composición sobre una figura de entrada (Figura 3.13) usando la segunda versión del algoritmo (ComposerFigMelody2):

- Procesar la figura (Figura 3.14): se ordenan los vértices (identificados con números) y se halla la longitud de los lados.

- Calcular duraciones (Figura 3.15): se puede apreciar cómo el lado más largo obtiene una duración mayor (blanca) que los otros dos lados (negra). A pesar de que los dos lados más pequeños tienen diferentes

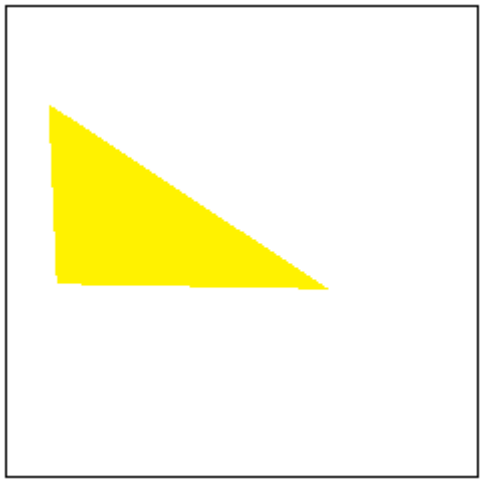


Figura 3.13: Figura de entrada

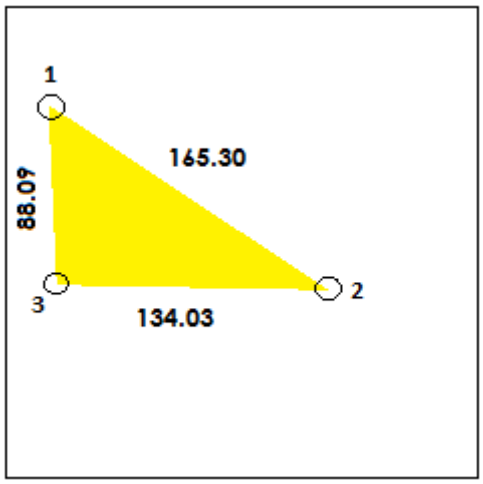


Figura 3.14: Figura de entrada con los vértices ordenados y con las longitudes de los lados

longitudes, se ha aproximado su valor a una misma duración.

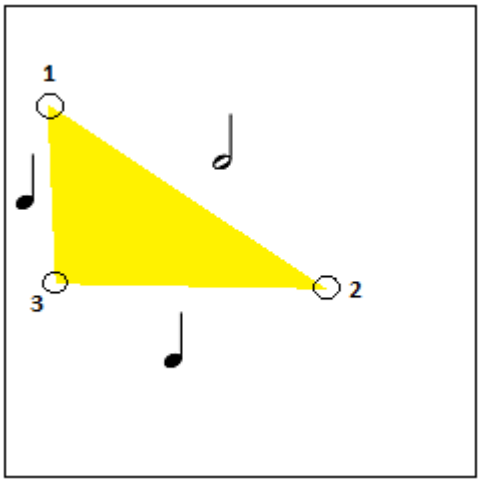


Figura 3.15: Figura de entrada con el ritmo producido

- Obtener tonos (Figura 3.16): para conseguirlo primero se tiene que ver qué nota corresponde al color de la figura (en este caso se usa la correspondencia propuesta por Scriabin en la que el amarillo es D "re"). Una vez se tiene la nota y su ángulo, se calcula la diferencia entre el nuevo ángulo y el anterior, el resultado es negativo por lo que se tiene que bajar de tono y como la diferencia en valor absoluto está entre los 10º y 30º, sólo se baja un escalón en la escala actual (en este ejemplo escala de Re Mayor), obteniendo la nueva nota: C "do". Para conseguir el tercero y último tono, se calcula la diferencia del último vértice restando el ángulo del tercer vértice con el del segundo. Se obtiene un valor positivo luego se tiene que subir el tono. Al estar entre 30º y 120º se sube de un salto en el acorde de nota fundamental el color de la figura (en este caso es el acorde de D "re": D - F "fa"- A "la"). Como no se está en el acorde con el último tono obtenido (C "do") se viaja al tono más cercano que pertenezca al acorde, es decir, D "re". Y de ahí como la diferencia de ángulos es positiva se va a hacer un salto ascendente (a tono superior) de una unidad dentro del acorde, se pasa al tono F "fa".
- Sintetizar las notas (Partitura de Figura 3.16): se tienen las duraciones y se tienen los tonos. Sólo falta juntarlos para crear las notas teniendo en cuenta el orden de los vértices.

Y con la misma figura (Figura 3.13), se ve cual sería la salida de la

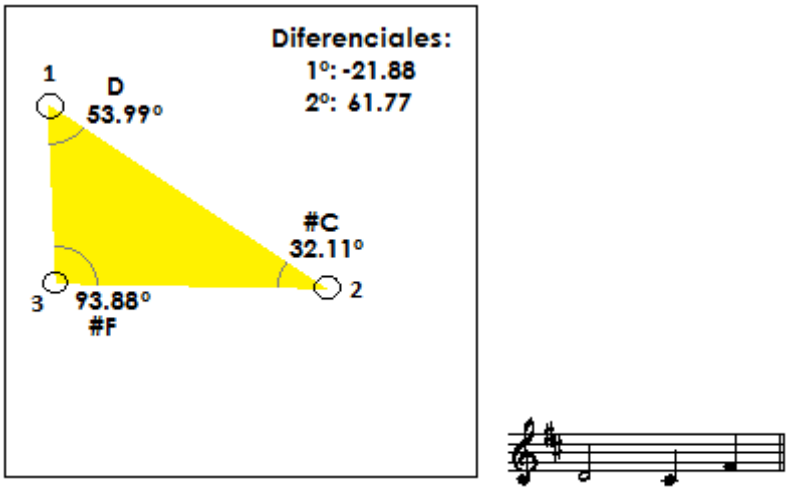


Figura 3.16: Figura de entrada con los ángulos y los tonos que produce

primera versión del algoritmo (ComposerFigMelody1):

- Procesar la figura (Figura 3.14): este proceso es el mismo en ambas versiones.

- Calcular duraciones (Figura 3.17): se puede ver cómo el algoritmo es más fiel a la correspondencia de las longitudes de los lados con las duraciones. Ahora los lados que antes tenían la misma duración, son distintos con una pequeña diferencia.

- Obtener tonos (Figura 3.18): en esta versión el algoritmo es de correspondencia directa entre el ángulo del vértice y el tono. Al igual que en el anterior algoritmo, el primer tono lo determina el color de la figura. Después se tiene el ángulo del segundo vértice que es pequeño y por tanto genera un pequeño salto en el tono. El tercer vértice tiene un ángulo mucho mayor y como se puede observar, se da un salto mayor respecto al anterior tono.

- Sintetizar las notas (Partitura de Figura 3.18): el mismo proceso que en el algoritmo de la segunda versión.

3.5.2. 2ª Voz: Melodía Secundaria

La segunda voz sirve de acompañamiento a la primera. Se usa la misma estructura que a la hora de componer la melodía principal aunque con

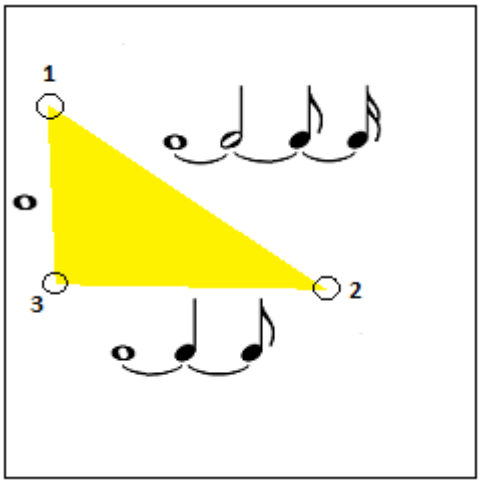


Figura 3.17: Figura de entrada con los lados y el ritmo conseguido

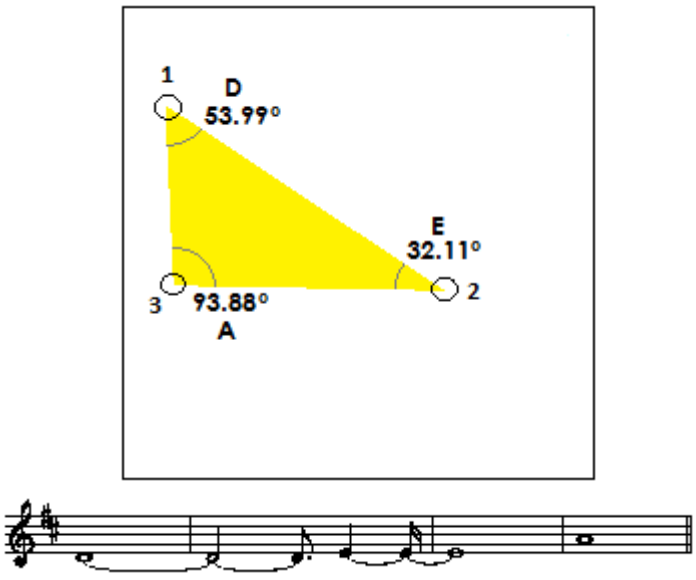


Figura 3.18: Figura de entrada con los ángulos y los tonos producidos

algunos cambios. Además se necesita el segmento de melodía principal como entrada para poder basarse en ella a la hora de componer, pudiendo conseguir resultados que siguen las bases armónicas de la música.

- Procesar la figura: esta fase será la misma que la usada en la primera voz.
- Calcular duraciones: se usa el mismo proceso que la segunda versión de la primera voz pero en este caso se añade un paso más. Las duraciones obtenidas se adaptan para que el total sea la longitud pedida y nunca mayor que el segmento de la melodía. Además, si la duración total del acompañamiento es menor que la de la voz principal, hay que decidir en qué momento de la melodía va a empezar el acompañamiento.
- Obtener tonos: respecto al anterior algoritmo, en este apartado se sigue sacando una correlación entre los ángulos y los tonos, pero se añade un proceso intermedio. Mientras se generan los tonos se analiza el comportamiento de la segunda voz respecto a la melodía principal y si es necesario, se aplican pequeñas variaciones para disminuir las disonancias que puedan aparecer al juntar ambas voces. Estos cambios, que siguen los principios de las técnicas más básicas del contrapunto, se activan cuando el intervalo entre la primera y la segunda voz es disonante y además se mantiene un periodo alargado en el tiempo.
  Las variaciones aplicadas intentarán que los cambios sean mínimos, por ejemplo, si un tono ha bajado pero lo ha hecho a un tono disonante de larga duración respecto a la primera voz entonces se cambia el tono por otro que esté lo más cerca del original.
- Sintetizar las notas: se sigue el mismo algoritmo pero con un añadido. Hay que tener en cuenta que la segunda voz puede tener una duración menor que la primera voz. Para que sea consistente la composición, todas las voces tienen que tener la misma duración y por tanto hay que rellenar con silencios la parte que falta a la segunda voz.

Se muestra un ejemplo con la siguiente figura de entrada (Figura 3.19):

- Procesar la figura: Igual que en los anteriores ejemplos se calcula la longitud de los lados de la figura y el orden de los vértices.
- Calcular duraciones (Figuras 3.20 y 3.21): En la primera figura se ven las duraciones obtenidas sin adaptar y en la segunda figura se puede observar cómo la duración de algunos lados ha disminuido para conseguir la duración deseada.
- Obtener tonos (Figuras 3.22 y 3.23): En la primera figura se puede observar los tonos generados sin usar técnicas de contrapunto. La tercera y sexta nota generada produce una disonancia de larga duración

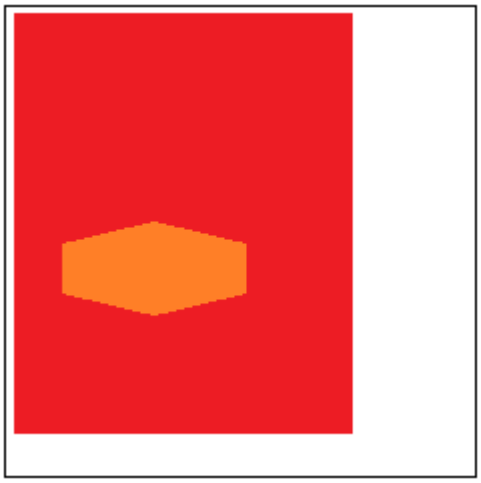


Figura 3.19: Figuras de entrada para componer con la figura interior

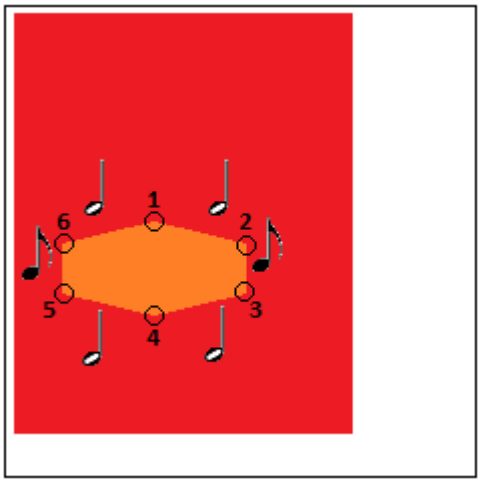


Figura 3.20: Figuras de entrada con los lados de la figura interior y las duraciones obtenidas sin adaptación

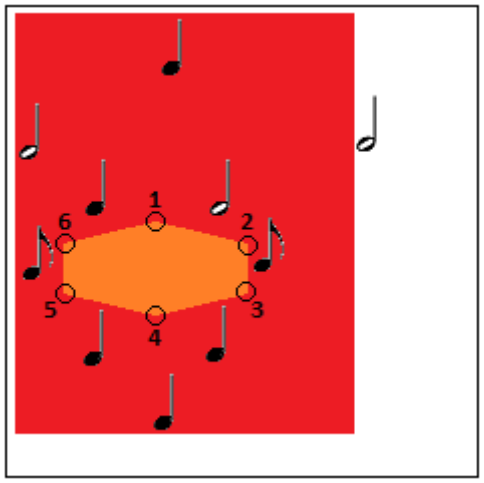


Figura 3.21: Figuras de entrada con la duración de la figura exterior y con las duraciones adaptadas de la interior

(negra) con la voz principal. Si se aplican las técnicas contrapuntísticas (reglas melódicas de intervalos horizontales e intervalos verticales, [23]) se puede comprobar cómo se han cambiado por un tono cercano sin perder la idea del salto de tono que indicaba el ángulo de los vértices.

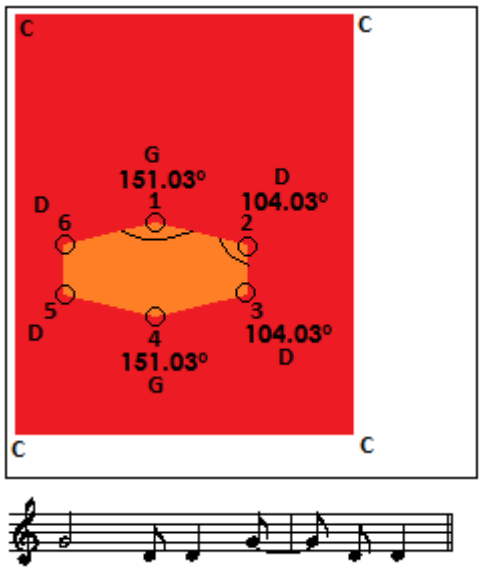


Figura 3.22: Figuras de entrada con los ángulos y los tonos producidos sin técnicas contrapuntísticas

- Sintetizar las notas (Partitura de Figura 3.23): Igual que la primera

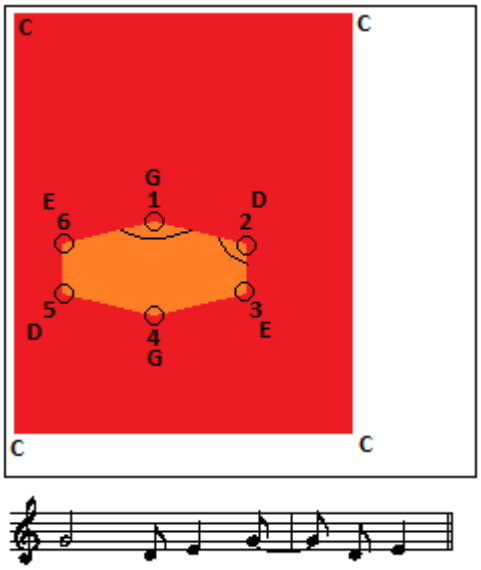


Figura 3.23: Figuras de entrada con los ángulos y los tonos producidos con técnicas contrapuntísticas

voz pero se añaden silencios si son necesarios. En este ejemplo no lo son.

3.5.3. 3ª Voz: Bajo

El bajo es la voz que se encarga de mantener la armonía de la pieza musical compuesta. Por tanto se ha optado por una estructura sencilla.

El algoritmo desarrollado consiste en usar siempre notas de una misma duración (por defecto redondas) cuyo tono sea el color de la figura de entrada y cuya duración sea la de la figura dada (Figura 3.24). Si las duraciones no son suficientes para rellenar la duración total, se termina usando una duración menor para la última nota.

Otra posibilidad habilitada para el bajo es usar el algoritmo de la primera voz de forma experimental, pero transportando los tonos una o varias octavas más grave (Figura 3.25). El número de octavas depende del tono más agudo y del tono más grave de la melodía compuesta evitando que al transportar la melodía se salga del límite de representación de las notas.

3.5.4. 4ª Voz: Ritmo

La cuarta voz será la encargada de la parte de percusión dentro de la pieza musical.

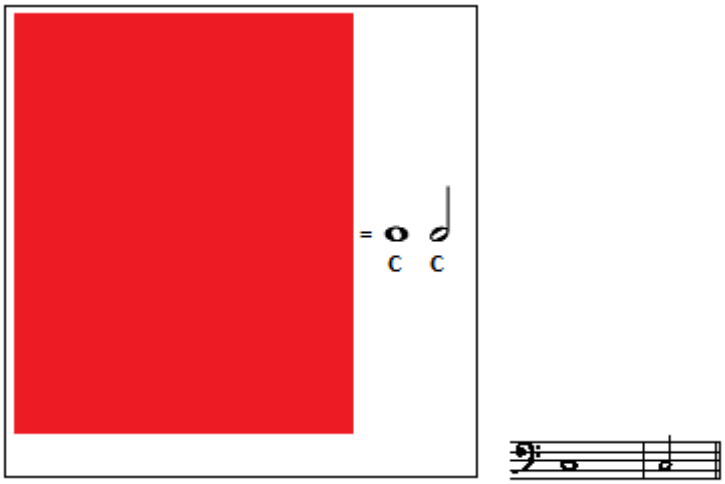


Figura 3.24: Figura de entrada con los tonos producidos

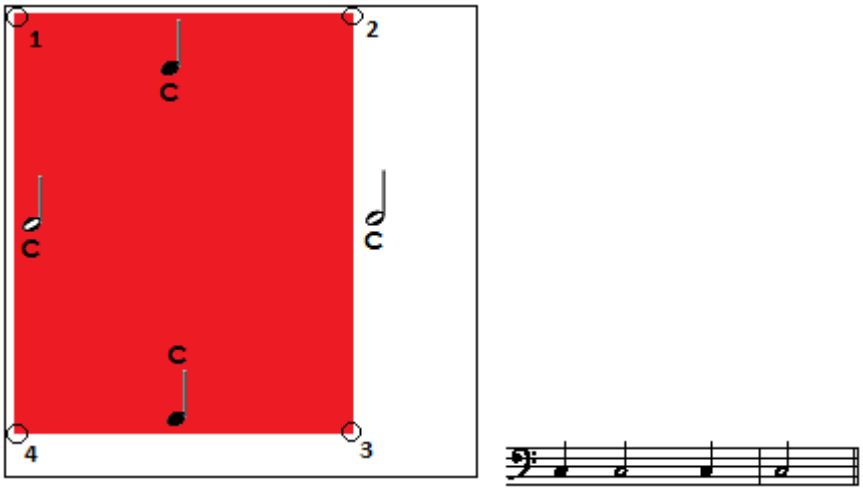


Figura 3.25: Figura de entrada con los tonos producidos

El algoritmo desarrollado se inspira en el concepto de inestabilidad de una figura de A. Pintado [1]. Se separa el proceso en 3 pasos:

- Asignación de vértices: Primero se crea un ritmo que consiste en pulsos largos de duración configurable (por defecto redondas). A cada pulso le es asignado una serie de vértices dependiendo de la densidad de vértices de la figura (a mayor número de vértices, se le asignarán más a cada pulso). Hay que tener en cuenta que los vértices están ordenados como en los anteriores algoritmos.

- Calcular círculo equivalente: Se calcula el círculo de área equivalente a la figura dada. Este círculo tendrá el mismo área y de centro el punto de equilibrio de la figura de entrada (Figura 3.26).

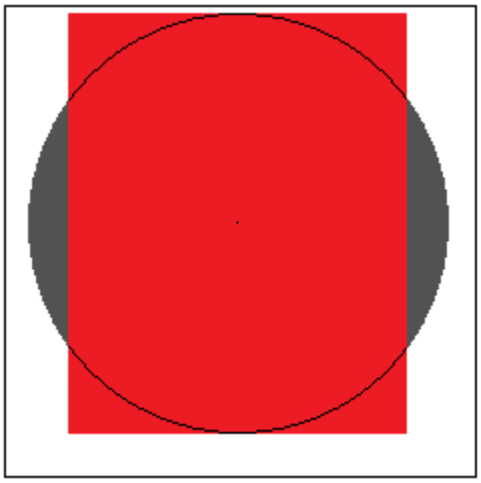


Figura 3.26: Figura de entrada y el círculo equivalente

- Dividir el ritmo: Después se va calculando cuanta desviación tienen los vértices respecto al perímetro del círculo. A mayor distancia, mayor desviación y por tanto mayor inestabilidad (Figura 3.27). La inestabilidad produce que el ritmo se subdivida en más notas de menor duración (Figura 3.28).

3.5.5. Mezclador de voces

Los anteriores algoritmos sirven para crear un trozo de música de una voz a partir de una figura, el siguiente paso es juntar las distintas voces para formar una pieza musical completa, para ello se utiliza el "Mezclador". Este algoritmo se puede dividir en tres pasos diferentes. Actualmente se tienen dos diferentes versiones:

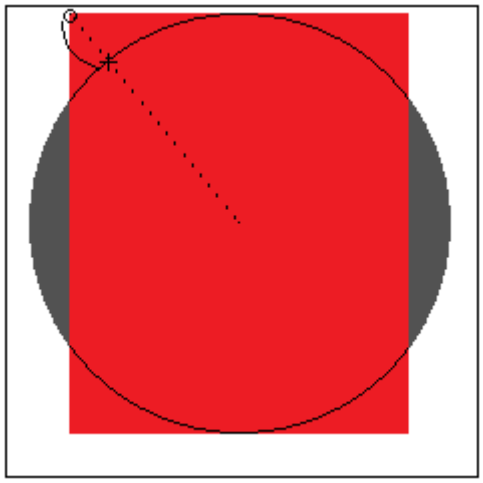


Figura 3.27: Figura de entrada con el círculo equivalente y la desviación del primer vértice

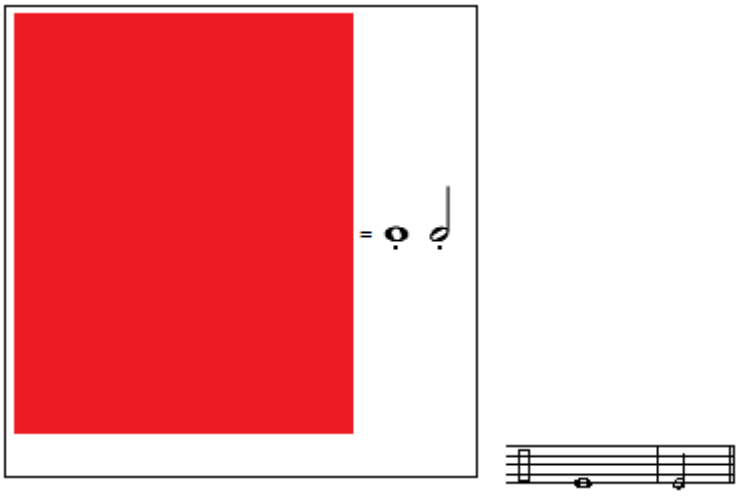


Figura 3.28: Figura de entrada y los ritmos producidos

La primera versión del Mezclador sólo tiene en cuenta la primera voz (melodía) y la cuarta (ritmo).

- Ordenar las figuras: Lo primero que se hace es ordenar las figuras según su relevancia. Para conseguirlo se utiliza un parámetro de vistosidad que se calcula a partir de tres características importantes de cada figura:

  - La primera es lo llamativo que es el color de la figura. Se descompone el color en sus tres componentes principales RGB y se da peso a cada una de ellas. Luego este valor se multiplica por la saturación del color.
  - La segunda componente es el área que ocupa la figura. Cuanto más grande sea, más vistosa es.
  - La tercera componente es la distancia que tiene del centro de la imagen. El ojo humano enfoca de centro a laterales cuando ve una imagen por primera vez, por ello tiene más vistosidad una figura que se encuentra en el centro de la imagen que en un lateral o esquina.

  La fórmula con los pesos de cada componente es la siguiente:

  $$vistosidad(Figura) = A * rgb.saturation * (rgb.red * pR + rgb.green * pG + rgb.blue * pB) + B * area + C * distanceCenter$$

  donde pR = 0.45, pG = 0.35 y pB = 0.20 son las constantes de influencia de cada color básico y las constantes A = 0.5, B = 0.3 y C = 0.2 son los valores de importancia de cada atributo de la figura. Estos valores se han obtenido de forma experimental.

- Componer voces: Tras haber ordenado las figuras por vistosidad se va componiendo con cada una de forma iterativa las distintas voces. En este caso solo se compone la melodía y el ritmo con cada una de las figuras.

- Mezclar voces: Una vez se tienen todas las voces compuestas se juntan y se crea la pieza musical añadiendo todos los parámetros necesarios (tempo, armadura de clave, título de la pieza musical, ...).

En la segunda versión del Mezclador, se tienen en cuenta las cuatro voces disponibles. Por ello tendrá más complejidad que la primera versión.

- Ordenar las figuras: Este primer paso se hace de la misma forma que la primera versión.

- Componer voces: Para la segunda versión del Mezclador se van a diferenciar dos tipos de figuras: figuras padre y figuras hija. Las llamadas figuras hija son aquellas que se localizan dentro de otra figura. Las figuras hija pueden tener a su vez otras figuras hija. Las figuras padre son aquellas que no forman parte de otra figura (es decir, no son hijas de ninguna figura) y pueden contener figuras hija. Es fácil de clasificarlas por la estructura arbórea en la que están organizadas.

  Una vez están clasificadas, se va iterando por cada figura padre componiendo su melodía, bajo y ritmo y si tiene alguna figura hija, entonces también se añade la segunda melodía dando como entrada al algoritmo de composición de acompañamiento (segunda voz) la figura hija y la melodía ya compuesta. Por cada figura interior (hija) se compone un nuevo bloque de la figura padre con diferentes acompañamientos.

- Mezclar voces: Se realiza de la misma forma que la primera versión del Mezclador.

Se muestra un ejemplo de la segunda versión del algoritmo, para simplificar se usará las figuras vistas anteriormente (Figura 3.29).

- Ordenar las figuras: Se ordenan las figuras de tal forma que en primer lugar está el rectángulo rojo, en segundo lugar el triángulo amarillo y al final el hexágono irregular.

- Componer voces (Figura 3.30): Primero se compone la voz principal, bajo y ritmo de la figura padre, que en este caso es el rectángulo. Después para cada figura hija (triángulo y hexágono) se compone su acompañamiento. Se puede observar en la partitura cómo primero se introduce el trozo correspondiente al rectángulo con el triángulo y que seguidamente se compone el trozo del rectángulo con el hexágono.

- Mezclar voces (Partitura de Figura 3.30): Por último se añaden las voces correctamente a la pieza musical y se rellenan los valores de los atributos necesarios para crear la partitura y el archivo de audio midi.

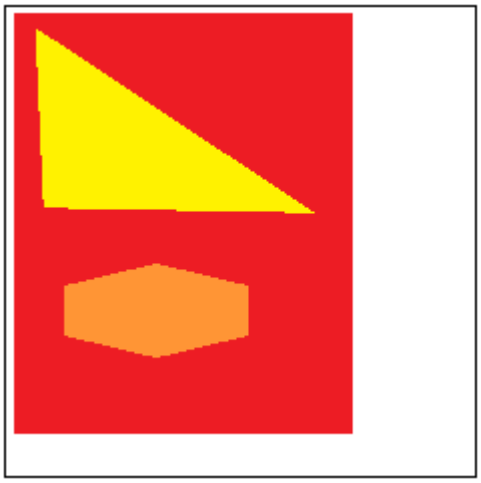


Figura 3.29: Figuras de entrada

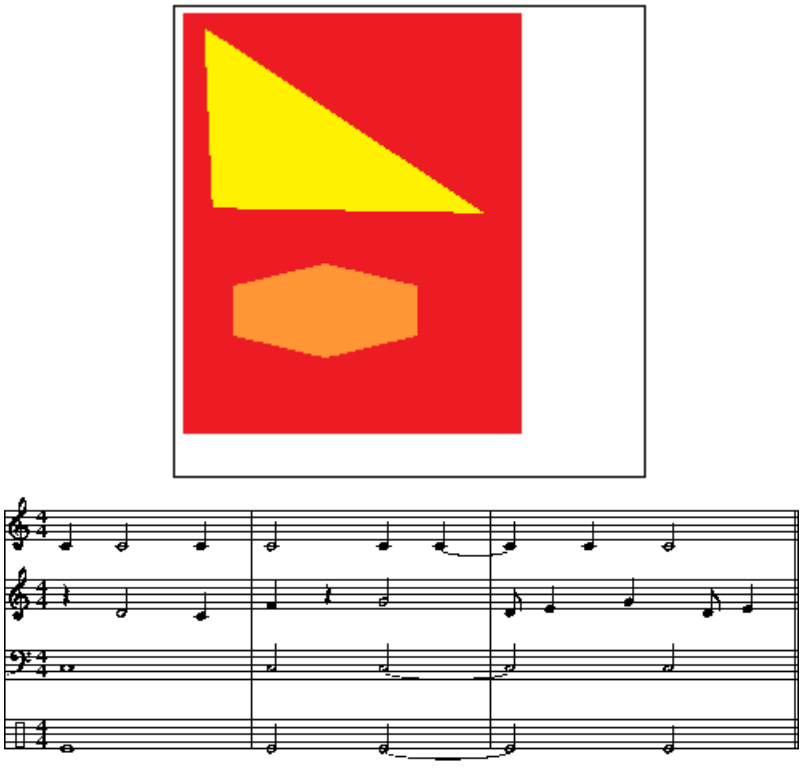


Figura 3.30: Figuras de entrada con las voces producidas

Capítulo 4

Arquitectura

4.1. Introducción

Este capítulo explicará de forma detallada la estructura del sistema desarrollado usando el estandar UML. Este sistema ha sido implementado usando el lenguaje C++ para todos los módulos, apoyándose de librerías y tecnología detalladas en el Apéndice A.

Se comenzará explicando dos pilares básicos del proyecto: la representación de una imagen analizada y la de una pieza musical, para después describir cada uno de los módulos del proyecto.

4.2. Representación de la imagen analizada

Tanto en el análisis de imágenes como en el proceso de composición hay un aspecto fundamental a estudiar: la representación de datos de las figuras.

De acuerdo con lo explicado en previas secciones, el producto de salida (y el de entrada de la composición musical) es un archivo XML que almacena el análisis de la imagen dada como entrada. Este archivo contiene toda la información que se considera para representar los elementos de una imagen, y servirá como base para explicar el contenido y estructura lógica de los datos analizados.

Por otro lado, en esta sección se mostrará también la estructura usada en la implementación de los Objetos relacionados con estos datos y sus usos. Dado que tanto el módulo de análisis como el de composición hacen uso de esta información , se introducirán también en esta sección las extensiones y capacidades de la estructura de datos creada.

4.2.1. Almacenamiento y estructura

La información correspondiente a la salida del análisis de imágenes es tal y como muestra la Figura 4.1. En ella, los nodos están representados por cajas rectangulares, sus atributos con cuadrados y, finalmente, los elementos con círculos.

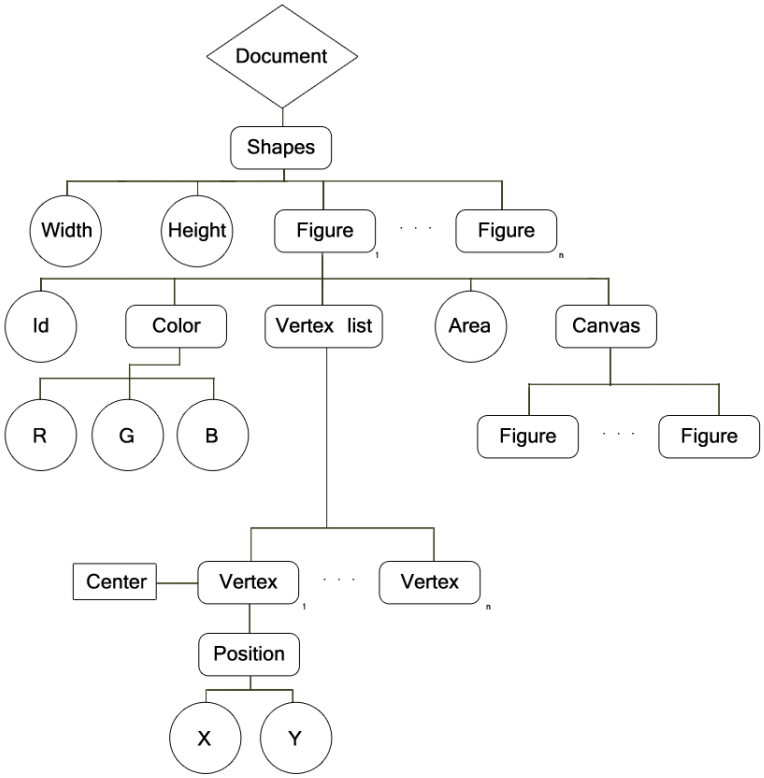


Figura 4.1: Estructura del archivo xml de análisis

La estructura (representada en formato XML) es:

- **Shapes**: representa la imagen completa y contiene un listado de las figuras resultantes del análisis.
- **Width**: almacena el ancho total de la imagen.
- **Height**: almacena el alto total de la imagen.
- **Figure**: contiene la información relativa a un polígono en la imagen.
  - **Id**: identificador único de la figura.

- **Color**: identificador para almacenar la información sobre el color de la figura.
  - **RGB**: delimitar la información sobre los valores RGB del color.
    - **R**: valor referente a la cantidad de rojo en la escala RGB.
    - **G**: valor referente a la cantidad de verde en la escala RGB.
    - **B**: valor referente a la cantidad de azul presente en escala RGB.
- **VertexList**: lista de vértices que componen el polígono.
  - **Vertex [type="normal"]**: identificador para delimitar un vértice que pertenece a uno de los extremos de una recta o curva del polígono.
    - **Position**: identificador para delimitar la posición de un vértice en la imagen.
      - **X**: componente x de la posición del vértice.
      - **Y**: componente y de la posición del vértice.
    - **Vertex [type="center"]**: identificador para delimitar un vértice que señala el centro de circunferencia que forma la curva que conecta el vértice anterior a este con el siguiente.

- **Area**: valor numérico del área de una figura.

- **Canvas**: identificador que delimita las figuras situadas dentro de la figura que contiene este elemento.

Como se puede ver, la imagen analizada está compuesta por una lista de figuras (o polígonos) jerárquicas según su posición geométrica. Cada figura representa una mancha de color detectada por el analizador de imágenes para ello guarda la información de sus vértices, color y área.

4.2.2. Vista de Implementación

La vista principal de la estructura en un sistema basado en objetos como el tratado se muestra en la Figura 4.2.

Como se puede ver, se ha implementado mediante tres clases principales más una librería externa:

- **Figures:** almacena la lista de figuras presentes en la imagen, permite manipular y acceder a ellas tanto en forma de lista (colocándose todas las figuras existentes de forma seguida y sin jerarquías) como en forma

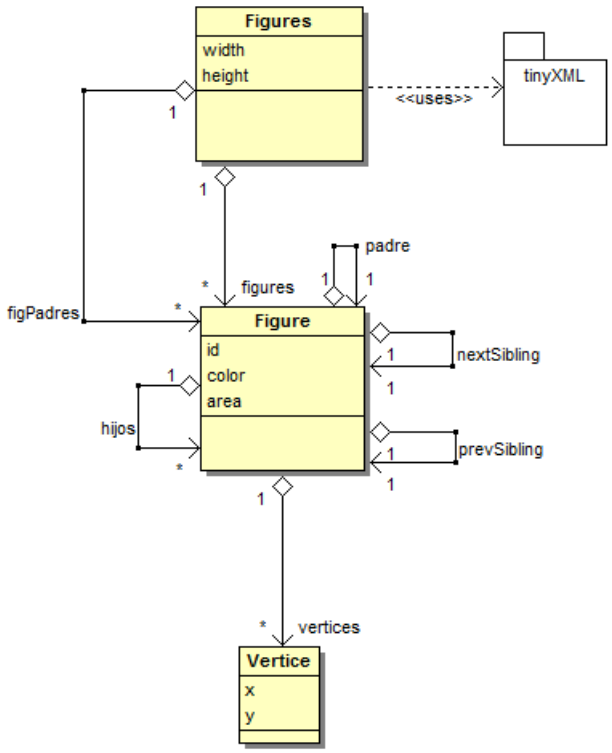


Figura 4.2: Diagrama de clases de la estructura de datos *Figuras*

de árbol (jerarquizándolas según quién está dentro de quién: una figura que contiene en su interior todos los vértices de otra será su "padre"), ocupándose de forma autosuficiente de mantener ésta última estructura al añadir nuevas figuras.

Además posee también funcionalidad para guardar y cargar su información en formato XML, así como para eliminar figuras repetidas o volver a calcular el color de cada figura teniendo en cuenta las figuras que contiene (ver Sección 3.4 para más detalles sobre estas funcionalidades).

- **Figure:** representa cada elemento almacenado en la clase Figures. Contiene, como nodo de la estructura arbórea creada en Figures, los atributos necesarios para referenciar a sus padres, sus hijos y sus hermanos, y como elemento representativo de una figura en la imagen (Sección 4.2.1), el color, area y lista de vértices correspondiente a la imagen.

  Presenta métodos para poder acarrear funciones geométricas básicas (como transformar su representación de vértices de coordenadas cartesianas a euclídeas) que simplifican el desarrollo de los algoritmos explicados en la Sección 3.5.

- **Vertice:** se trata de la unidad más pequeña de esta estructura, y representa las coordenadas en 2 dimensiones de un vértice en el plano cartesiano. Como se detalla en la Sección 4.2.1, da la opción de ser considerado un "centro", de forma que, en la lista de vértices de la Figura que lo almacene, puede actuar como el centro entre el vértice anterior y el siguiente, formando por tanto un arco entre los 3 vértices y permitiendo realizar curvas en las figuras.

- **TinyXML:** se trata de un paquete externo de código libre que permite generar y manipular archivos XML.

4.2.3. Usos de la estructura

Como ya se ha comentado, tanto el proceso de análisis como el de composición hacen uso de esta estructura y, dado que para la implementación de ambos módulos se ha seleccionado el mismo lenguaje orientado a objetos (C++), estos compartirán la implementación de esta estructura.

Sin embargo, existen diferencias entre los usos y ampliaciones que cada módulo aplica, y es por eso que no hacen uso de la estructura de forma directa, sino que cada módulo heredará la clase *Figure* para satisfacer sus

propias necesidades. El resultado es el diagrama mostrado en la Figura 4.3, con diferenciación entre *FigureImg* y *FigureMusic*.

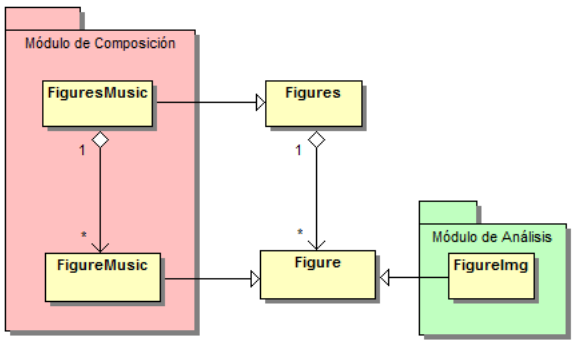


Figura 4.3: Diagrama de clases de los diferentes tipos de Figura

Los detalles de cada clase hija serán explicados en posteriores secciones.

4.3. Formato de representación de música

La estructura para representar la música propuesta en el proyecto está determinada por un árbol que trabaja desde el elemento más general, la canción que se va a componer, al más específico, cada una de las notas que componen dicha canción. Se puede ver la estructura en el siguiente esquema (Figura 4.4).

Music: al nivel de Music se trabaja con la información relativa a toda la canción que se va a componer. Por un lado está la información relativa al nombre de la pieza y del compositor que la ha hecho así como la ruta del archivo de sonido cuando se guarde. Pero la parte más importante es el contenido de las voces a partir de las cuales está formada la canción. También es importante el tempo global de la pieza y la herramienta que se usará para crear el archivo de audio de salida.

Voices: esta clase es un envoltorio de un array al que se le ha añadido pequeña funcionalidad, como la posibilidad de editar algún parámetro a todas las voces que contiene.

Voz: por debajo de Music se trabaja con las voces, Voice. Dentro de cada Voice se determina el instrumento, la tonalidad y el tempo con el que será transformada esta voz en sonido. La parte sustancial de cada voz son

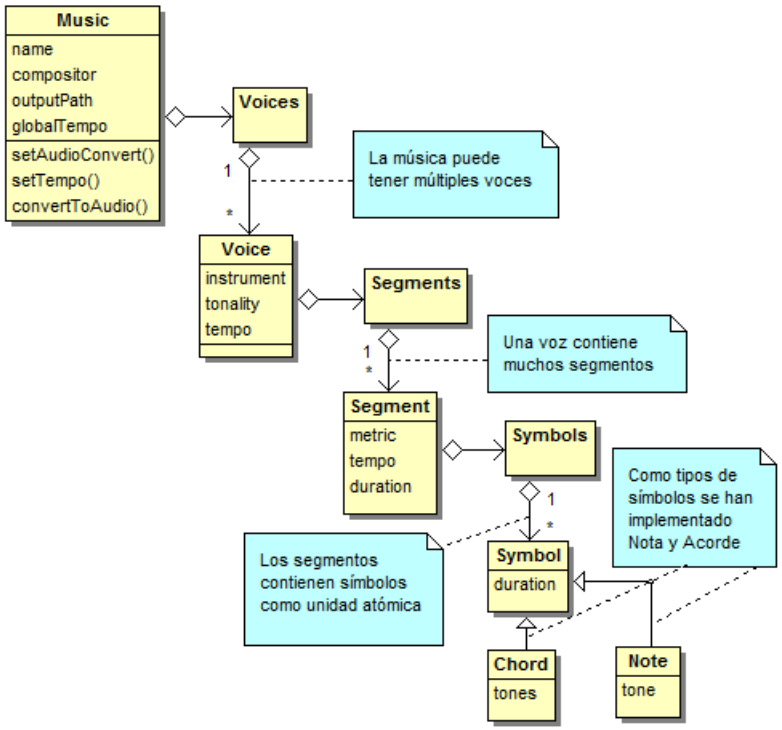


Figura 4.4: Estructura de datos de una pieza musica

los Segments, donde se almacena los datos musicales.

Segments: al igual que Voices, es un envoltorio de un array al que se le ha añadido alguna pequeña funcionalidad adicional.

Segment: su principal cometido es almacenar de forma serializable los diferentes símbolos musicales de los que está compuesta la pieza musical. A su vez, estos elementos tienen la posibilidad de establecer su propia métrica y tempo. Además es importante la duración del segmento, que se irá incrementando cada vez que se añada un nuevo símbolo.

Symbol: son la unidad a partir de la cual se crean todos los elementos básicos de la música, Symbol está formado únicamente por una duración. Actualmente se ha desarrollado dos tipos de elementos a partir de Symbol:

- **Note**: estos símbolos tienen asociado además de duración, determinada por el componente anterior, un tono que viene representado por un número que posteriormente será interpretado con la herramienta que generará el archivo de audio.

- **Chord**: los acordes están formados de la misma manera que las notas, pero en lugar de tener asociado un único tono pueden tener de dos a cuatro tonos diferentes. En la práctica, esto se traduce a varios tonos sonando a la vez durante el mismo tiempo dentro de la pieza musical sin la necesidad de múltiples voces.

4.4. Módulo de composición

El módulo de composición es el encargado de crear un archivo de audio a partir de un archivo XML con información de figuras. Previo a crear el archivo de audio, será necesario componer la música en base a la configuración del usuario y a los datos de las figuras. La composición se verá plasmada en un fichero ABC siguiendo la notación musical ABC (Apéndice A.1). De esta manera se facilita la conversión en archivos de audio y además la posibilidad de mostrar la partitura de la pieza musical creada.

Para la conversión y manipulación de los archivos de audio se usará los sistemas externos abcMIDI (Apéndice A.2) y Timidity (Apéndice A.4). abcMIDI se usarán para convertir el archivo generado ABC en formato MIDI y a partir del MIDI se generará el formato WAV de sonido con Timidity. El proyecto de abcMIDI está formado por varios programas de los cuales sólo se utilizará abc2midi.

4.4.1. Vista de Implementación

La implementación realizada sigue el esquema de la Figura 4.5.

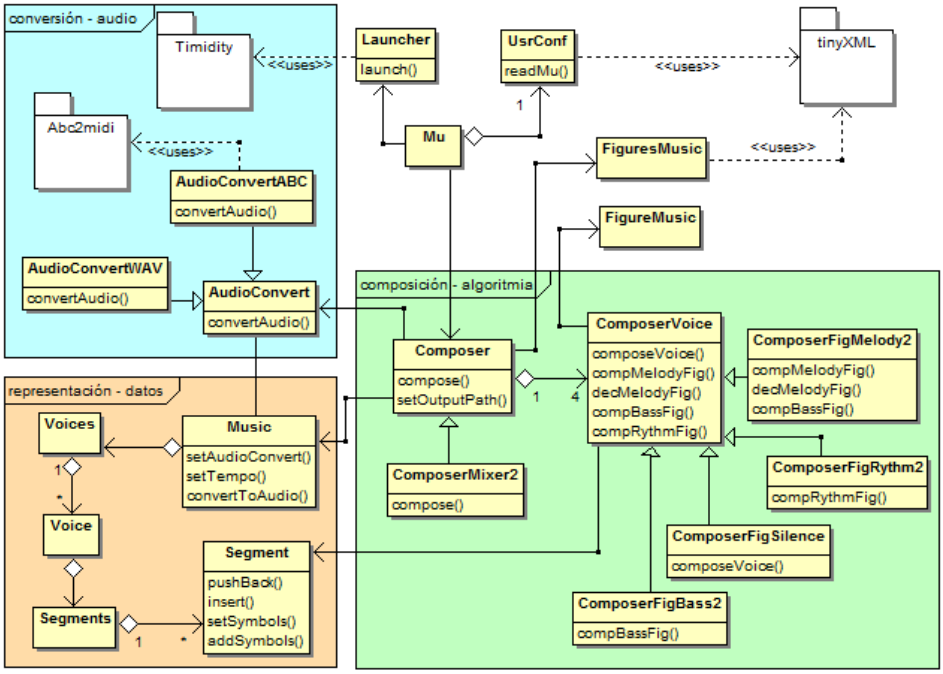


Figura 4.5: Diagrama de clases del módulo de Composición

Las clases más importante que se pueden observar son:

- **Mu:** se trata de la clase que controla la ejecución de este módulo. Se ocupa de crear y coordinar las clases principales de la composición. Además, al final se encarga de llamar al programa Timidity para que realice la conversión de Midi a Wav.

- **UsrConf:** como se explica en posteriores secciones, es una clase común que se ocupa de la lectura de los archivos de configuración que el usuario a especificado mediante la interfaz gráfica.

- **FiguresMusic:** es necesario crear una clase a partir de Figures que brinde una funcionalidad extra necesaria para el módulo de composición. Se explica más adelante los motivos de esta necesidad.

- **FigureMusic:** se trata de una extensión de la clase Figure, a la que toma como padre. Se explica más adelante los motivos de su creación.

- **Launcher:** ofrece la funcionalidad para ejecutar una aplicación externa. Se detalla más adelante, en la Sección 4.6.
- **TinyXML:** se trata de un paquete externo de código libre que permite generar y manipular archivos XML.

El resto de clases se pueden separar en tres apartados diferentes según su objetivo principal:

Compositores - Algoritmia (color verde Figura 4.5):

- **Composer:** encargado de procesar las figuras de entrada y crear la estructura de los datos de música. Su principal cometido es componer la música sirviendose de los métodos de composición que albergan los compositores de voces. Otra responsabilidad importante es crear el tipo de conversor adecuado para transformar la notación musical usada en archivos de audio.
- **ComposerVoice:** la tarea principal de un compositor de voz es crear un segmento de música a partir de una figura de entrada. Existe la posibilidad de componer música para las cuatro diferentes voces identificadas (ver Sección 3.5). De esta clase extienden los diferentes algoritmos de composición implementados.
- **ComposerFigMelody2:** un ejemplo de clase que extiende de ComposerVoice que implementa la funcionalidad de crear música para las voces primera, segunda y tercera con los métodos compMelodyFig( ), decMelodyFig( ) y compBassFig( ).
- **ComposerFigBass2, ComposerFigRythm2:** se explican con mayor minuciosidad los algoritmos implementados en la Sección 3.5).
- **ComposerFigSilence:** clase encargada de crear una voz que esté en silencio. Esto ocurre cuando el usuario desactiva alguna de las voces de composición.
- **ComposerMixer2:** clase que hereda de Composer. Es una de las implementaciones realizadas que cumple con los cometidos de los que está encargado la clase padre.

Representación - Datos (color salmón Figura 4.5):

- **Music:** encargado de almacenar la pieza musical que se está componiendo. Además debe comunicarse con AudioConvert para generar el archivo de audio. Esta clase se detalla con mayor detalle en la Sección 4.3.

- **Voices, Voice, Segments:** todas estas clases se detallan en la Sección 4.3.
- **Segment:** es la unidad sobre la que trabajan los compositores de voces. Se usa como contenedor de las notas creadas por los diferentes algoritmos. Más información en la Sección 4.3.

Conversión - Audio (color azul Figura 4.5):

- **AudioConvert:** la principal función es convertir los datos que se manejan en los compositores dentro del sistema en un archivo de audio que permita reproducirse.
- **AudioConvertABC:** esta clase implementa una posible solución a la conversión entre la notación musical usada y una salida estandar de audio. En este caso primero se transforma la notación en un archivo con notación ABC y seguidamente se usa la aplicación externa Abc2midi para convertir el archivo ABC en un archivo de sonido Midi.
- **AudioConvertWAV:** de forma experimental se desarrolló un sintetizador de sonido que, a partir de la notación, generaba directamente un archivo wav sin compresión. Actualmente no se usa, ya que se obtienen con Abc2midi y Timidity archivos de sonido con mejor calidad y riqueza.
- **Abc2midi:** aplicación externa encargada de convertir un archivo de entrada con formato ABC en un archivo de sonido midi.
- **Timidity:** aplicación externa capaz de repoducir y convertir diferentes formatos de audio. En el sistema se utiliza para convertir el archivo de sonido midi en wav.

El fujo de ejecución de una composición completa viene representado la Figura 4.6 y la Figura 4.7 con diagramas de secuencia o flujo. El primer diagrama reproduce la ejecución a alto nivel mientras que el segundo diagrama detalla la parte de ejecución encargada de la composición y conversión a archivo de sonido.

4.4.2. Figuras Musicales

El módulo de composición trata las figuras según sus necesidades. Por ello, lo que necesita saber de cada figura además de todos los datos que se encuentran dentro de Figures es la relevancia o vistosidad de cada una, para ello hereda de las clases Figures y Figure mencionadas anteriormente en la Sección 4.2.3, con las clases nuevas llamadas FiguresMusic y FigureMusic (Figura 4.8).

A la clase FigureMusic se le añade la siguiente funcionalidad:

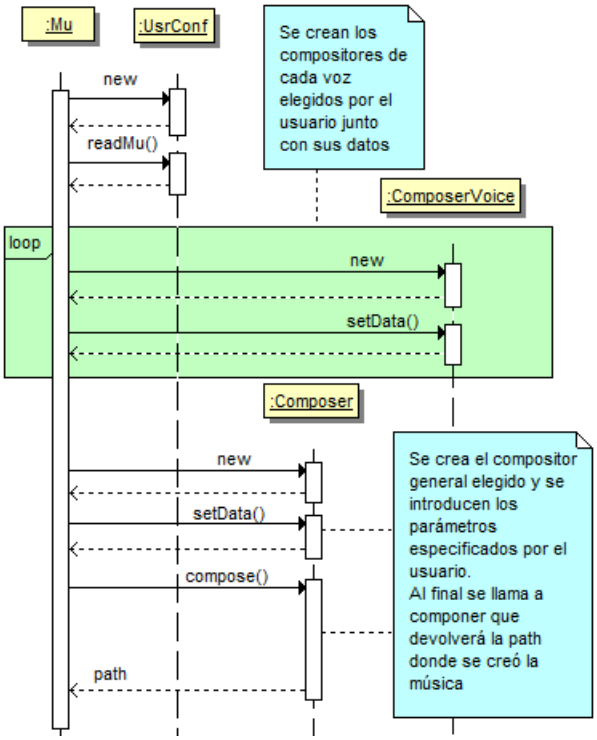


Figura 4.6: Flujo de ejecución alto nivel de abstracción del módulo de composición

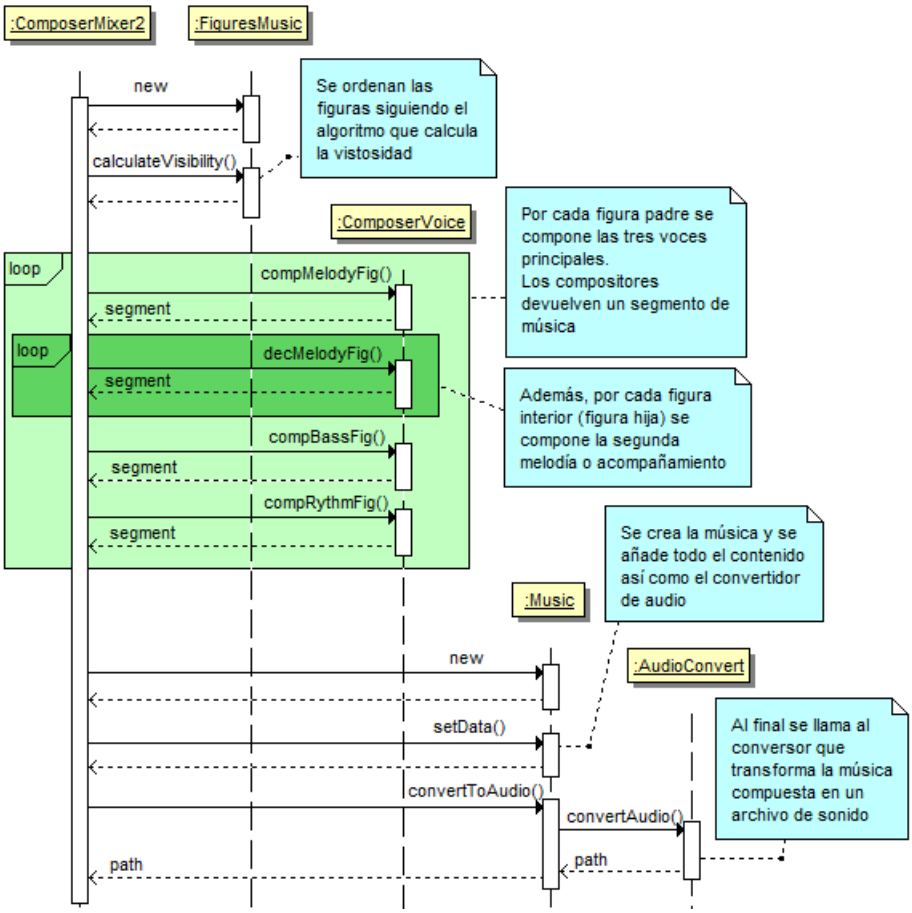


Figura 4.7: Dentro del módulo de composición, flujo de ejecución de los compositores

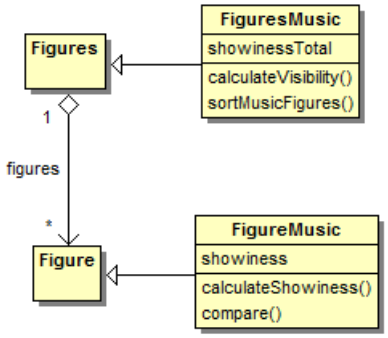


Figura 4.8: Diagrama de las clases Figures y FiguresMusic

- calcularVistosidad: se calcula la relevancia de una figura dentro de una imagen con los siguientes valores: cantidad de rojo de la figura, cantidad de verde de la figura, cantidad de azul de la figura, área de la figura y distancia al centro de la imagen. Cada característica tiene su propio peso. La fórmula que relaciona todas estas características se ve con detalle en la Sección 3.5. Es necesario este valor para poder clasificar las figuras y así los compositores puedan usar las figuras de forma organizada.
- compare: compara dos figuras según el valor de la vistosidad. Si dos figuras tienen misma vistosidad entonces tienen misma relevancia dentro de la imagen. Se emplea para poder ordenar las figuras.

A FiguresMusic se le añaden los elementos necesarios para poder considerar y utilizar la nueva funcionalidad añadida a FigureMusic:

- calcuteVisibility: primero calcula la vistosidad de cada figura si es que no se ha calculado ya, tomando los valores. Después se normalizan esos valores teniendo en cuenta el total de todas las figuras y la media. Función que usa sortMusicFigures para poder después ordenar las figuras.
- sortMusicFigures: dada una lista de figuras, devuelve la lista ordenada según la vistosidad de las figuras. Esta funcionalidad se utiliza por parte de los compositores, para ordenar las figuras por su criterio de relevancia dentro de la imagen.

4.5. Módulo de análisis

El módulo de análisis, como ya se ha comentado, es el encargado de, dada una imagen y una configuración de entrada, producir un archivo XML con la lista de polígonos que componen la imagen. Esta lista debe formar una nueva imagen lo más fiel posible a la imagen de entrada, teniendo en cuenta los ajustes especificados en el archivo de configuración.

Este módulo hace un gran uso de la librería externa OpenCV (Apéndice A.3) , de la cual se ayuda tanto para tratar imágenes como para componer formas y polígonos.

4.5.1. Vista de implementación

La estructura general del módulo se ve en la Figura 4.9.

En ella, se pueden observar las siguientes clases:

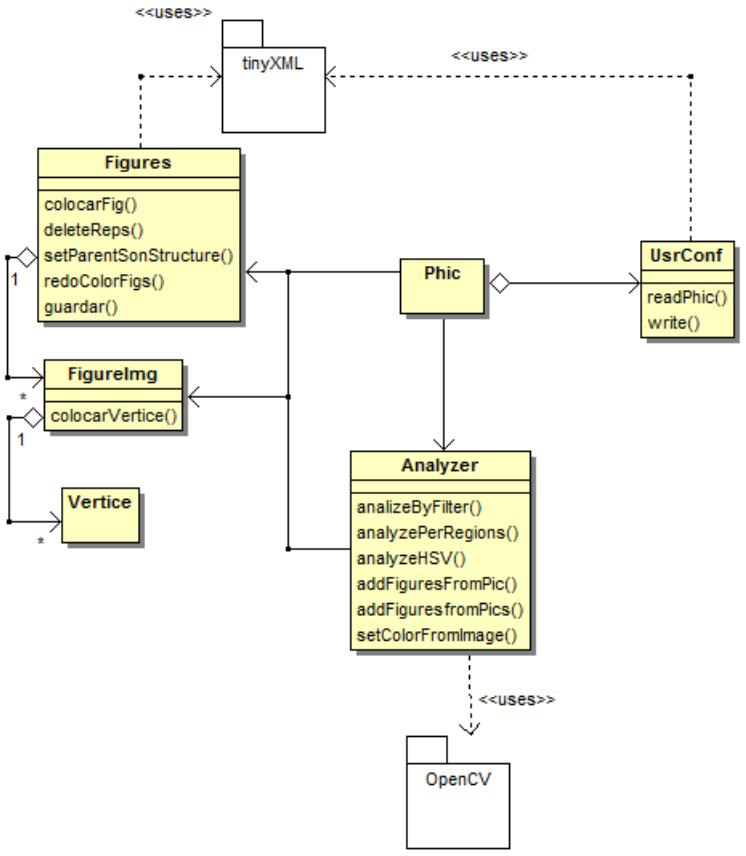


Figura 4.9: Vista de las clases del módulo de anális

- **Phic:** se trata de la clase que controla la completa ejecución de este módulo. Se ocupa de coordinar el resto de clases para que realicen las operaciones requeridas para completar el análisis.

- **Analizer:** realiza todas las operaciones de análisis y se comunica con la librería OpenCV.

- **UsrConf:** como se explica en posteriores secciones, es una clase común que se ocupa de la lectura de los archivos de configuración.

- **Figures:** el módulo de análisis utiliza de forma directa la clase Figures anteriormente mencionada. Se muestran en el diagrama los métodos que llevan a cabo las funcionalidades necesarias para la ejecución de este módulo.

- **FigureImg:** se trata de una especificación de la clase Figure, que toma como padre.

- **Vertice:** como se comenta en previas secciones, representa cada uno de los vértices que componen un polígono.

- **TinyXML:** Se trata de un paquete externo de código libre que permite generar y manipular archivos XML.

El flujo de ejecución de un análisis cualquiera realizado con este módulo se ve en las Figura 4.10 y 4.11. En la primera imagen se observa el proceso de inicialización del módulo en su totalidad, mientras que en la segunda se observa cómo es la clase Analyzer la que se comunica con OpenCV para realizar las opciones necesarias, ordenadas por Phic.

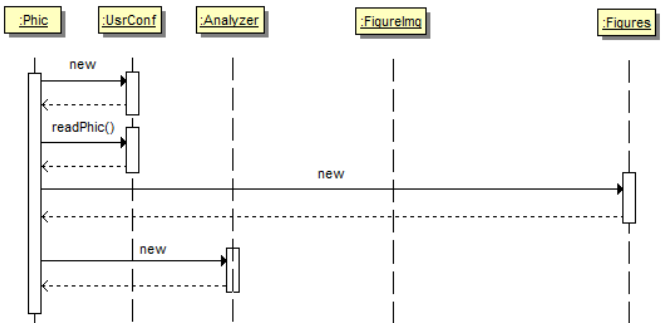


Figura 4.10: Flujo de inicialización del módulo de anális

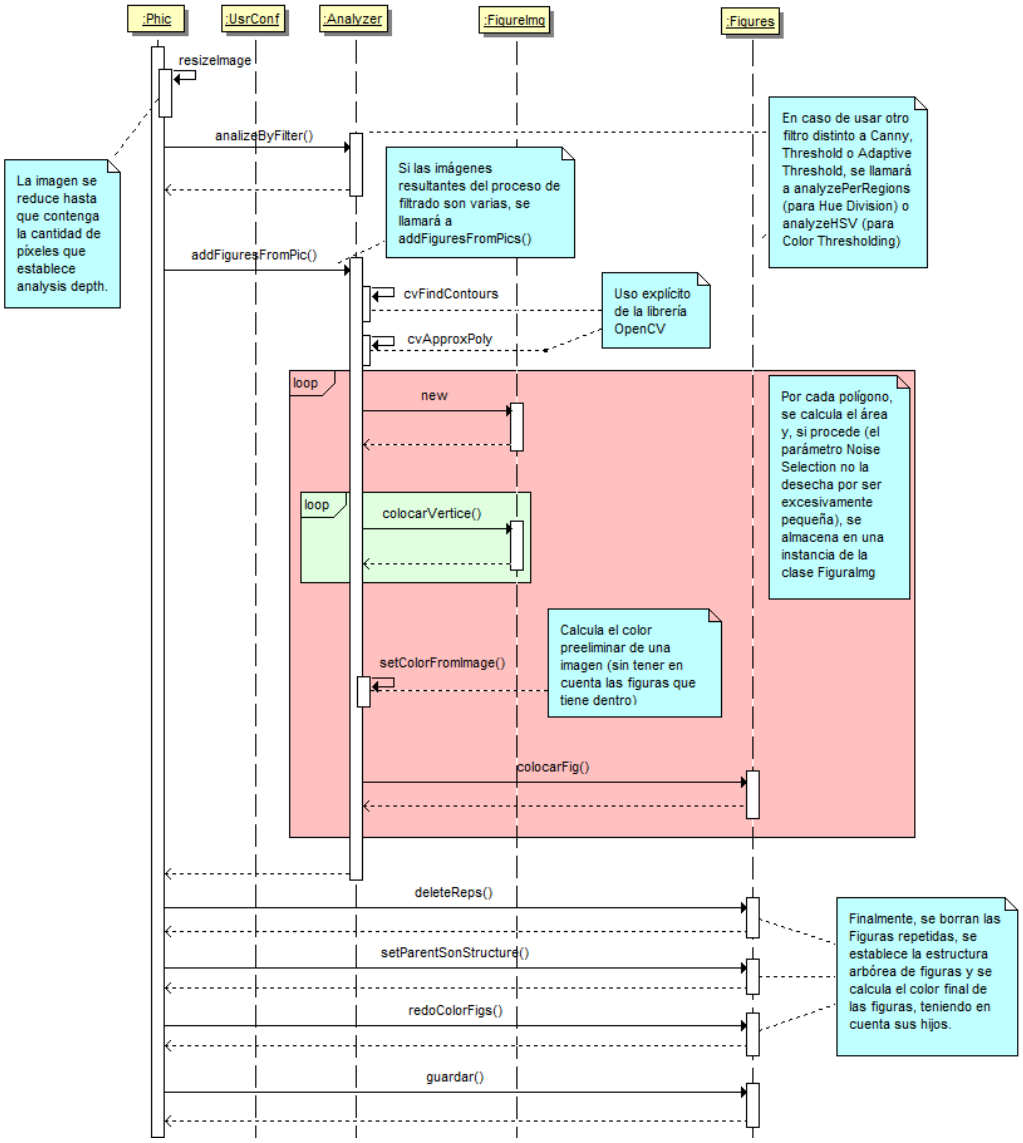


Figura 4.11: Flujo de ejecución del módulo de anális

4.6. Módulo de conexión

El módulo de conexión se encarga de lanzar los módulos de análisis y/o el de composición, según la demanda del usuario. Para ello, debe ejecutar los correspondientes programas asociados a cada módulo, de tal forma que la vista de despliegue es tal y como muestra la Figura 4.12.

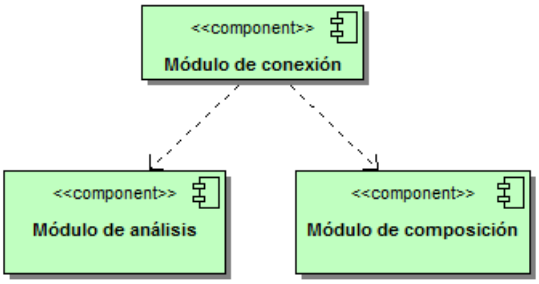


Figura 4.12: Diagrama de despliegue del módulo de interconexión

Es aquí donde surge el principal problema de que la aplicación sea multiplataforma, ya que distintos sistemas operativos, como Windows o los basados en UNIX, proponen distintas maneras de lanzar archivos ejecutables. Para solventar estos problemas, se han concentrado las consiguientes diferencias del código para cada sistema operativo en la clase *Launcher*, como se ve en la Figura 4.13.

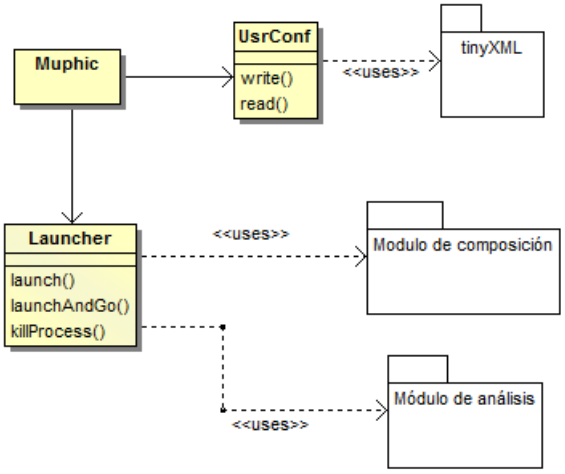


Figura 4.13: Diagrama de clases del módulo de interconexión

La clase Launcher ofrece funcionalidad para ejecutar una aplicación interrumpiendo la ejecución en curso (*Launcher::launch()*), o bien en paralelo (*Launcher::launchAndGo()*), así como para detener la ejecución de un programa lanzado (*Launcher::killProcess()*). De esta forma, la totalidad del resto del código del proyecto se ha realizado sin preocuparse de la plataforma de ejecución, delegando de Launcher siempre que era necesario realizar operaciones propias de cada sistema operativo.

En la Figura 4.14 se muestra un ejemplo de ejecución del módulo, en el que se ejecutan tanto el módulo de análisis como el de composición.

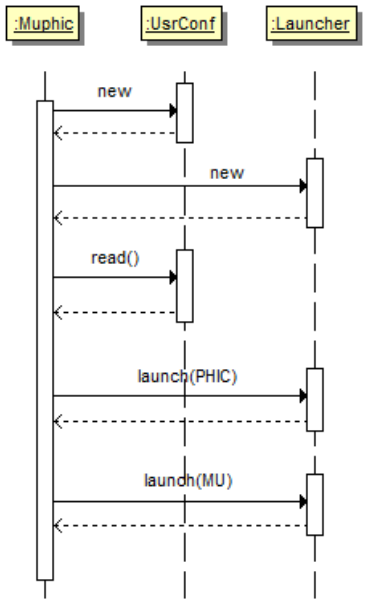


Figura 4.14: Diagrama de flujo del módulo de interconexión

Cabe destacar en esta sección el uso de la clase *UsrConf*, encargada de administrar el archivo de configuración del completo proceso de análisis y composición. Este archivo de configuración es el que permite la "comunicación" entre los distintos módulos del sistema. Está estructurado en forma de XML y contiene tanto los parámetros necesarios para la ejecución del módulo de análisis como los necesarios para la ejecución del módulo de composición (vistos en la Sección 2). Contiene también la información de la ruta destino de la composición. De forma adicional, permite indicar qué módulo se debe ejecutar, de forma que se puede ejecutar únicamente uno de los dos módulos o los dos a la vez. La clase *UsrConf* ofrece funcionalidad para escribir y leer (ya sea en su totalidad o la parte relativa a un único módulo) el archivo XML.

4.7. Interfaz Gráfica

La interfaz gráfica de la aplicación es la encargada de aunar todas las funcionalidades del sistema en un entorno integrado. Se ocupa de la creación del archivo de configuración necesario para la ejecución de los distintos módulos, así como de facilitar el acceso a los resultados del análisis y la composición.

Para su diseño se ha usado el quit de desarrollo Qt (Apéndice A.6), por ser multiplataforma y estar en sí misma orientada a móviles, facilitando así una futura movilización del sistema a máquinas portátiles. La versión de la librería usada está aplicada al lenguaje de programación C++, gracias a lo cual se ha podido reusar gran parte del código implementado en el resto del sistema, como son las clases UsrConf (encargada de almacenar la configuración de los módulos, como se vio en la Sección 4.6), y el paquete de representación de figuras (las clases *Figures* y derivadas, vistas en la Sección 4.2. La interfaz, como se ve en en los requisitos (Sección 3.3), debe ser capaz tanto de lanzar el módulo de conexión como de mostrar el resultado del análisis y la partitura de la pieza musical compuesta, así como reproducir dicha pieza generada.

Mientras que para mostrar el análisis y reproducir música se podrán utilizar los mismos recursos ofrecidos por la librería Qt, para mostrar las partituras será necesario hacer uso del visor de archivos gráficos predeterminado del sistema (en concreto se usa el formato de archivos xhtml, por facilidad de uso en las diferentes plataformas a las que está orientada la aplicación). Es por ello que se utiliza también la clase *Launcher* anteriormente explicada, que en esta ocasión lanzará cada aplicación en paralelo a la interfaz gráfica, permitiendo así detener la ejecución de módulos externos si el usuario considera que su período de actividad ha sido excesivo.

La siguiente sección analizará con detalle la implementación de las partes más importantes de esta interfaz.

4.7.1. Vista de implementación

La aplicación sigue la estructura básica de una interfaz diseñada sobre Qt, con un controlador encargado de manejar los eventos (*QMainWindow*) y una estructura que se encarga de actualizar la parte gráfica (*MuphicUI*), como se ve en la Figura 4.15.

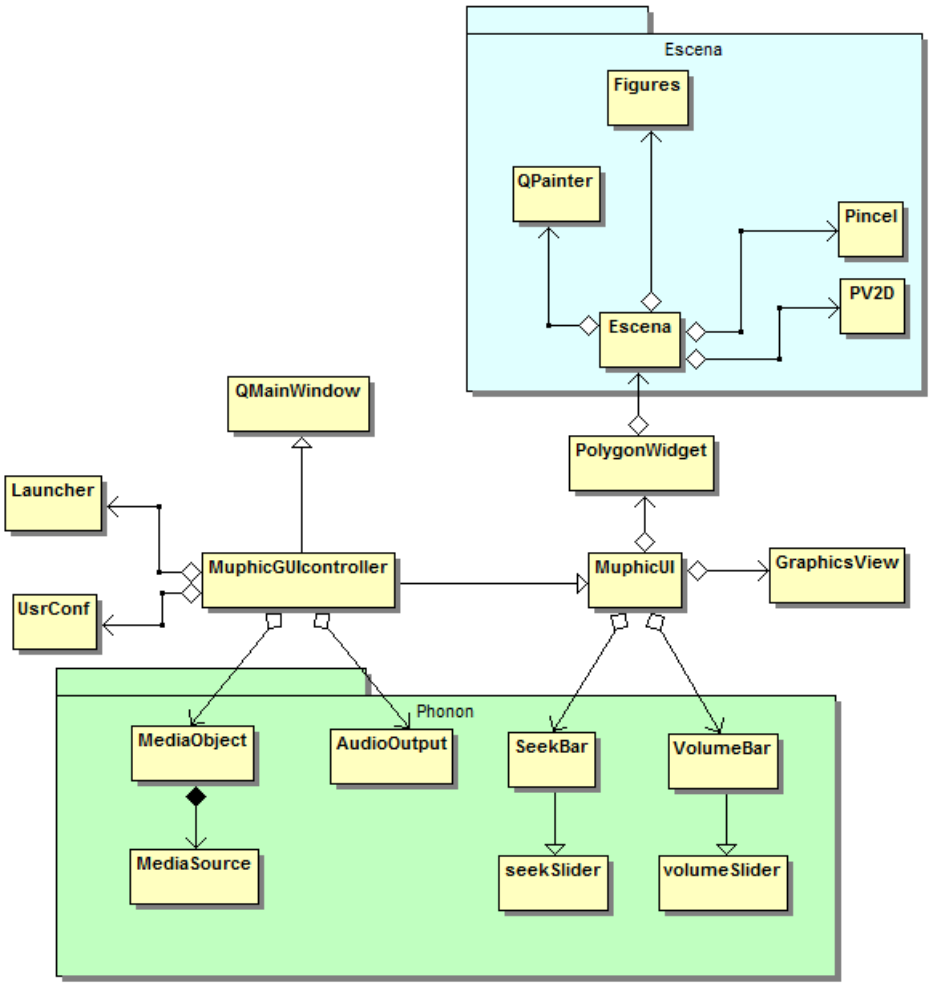


Figura 4.15: Diagrama de flujo del módulo de interconexión

Para mostrar la imagen seleccionada como entrada se usa un elemento propio de la librería Qt (*GraphicsView*), mientras que para la visualización del resultado final de un análisis recién efectuado, se usa una clase propia: *PolygonWidget*. Su estructura, como muestra la Figura 4.15, consta de un elemento principal, la *Escena*, encargada de administrar el marco de dibujo y sus contenidos. Ella hará uso de la estructura de representación de Figuras para cargar la imagen recién analizada y pintarla en el marco designado, gracias a la clase pintora *QPainter* proporcionada por la librería. Con ella, y utilizando la clase auxiliar *Pincel* (que facilita el pintado de líneas y arcos aproximados por rectas), mostrará por pantalla la escena cargada, proporcionando al usuario una visión fiel del análisis realizado.

Por otro lado, para la reproducción de audio se hace uso de una librería compaginable con Qt: Phonon (Apéndice A.7). Esta librería se encarga de cargar así como de iniciar, parar y modificar el transcurso de la reproducción de una pieza musical. El uso de esta librería, que soporta archivos de audio tales como wav o mp3, pero no midi, obliga a que el módulo de reproducción produzca una salida acorde a los formatos que esta librería es capaz de reproducir.

4.7.2. Vista de despliegue

A diferencia de otros apartados, donde la totalidad de los elementos externos se incluyen en el despliegue del módulo, en este caso se exigirá como requisito de instalación a los usuarios de sistemas UNIX el tener instalados en sus máquinas anfitrionas tanto la librería Qt como la librería adicional Phonon, para ahorrar espacio en el producto final. Se encuentran detalladas estas librerías en la Sección 2.5.

En la versión para sistemas Windows, por proporcionarse un instalable y no el código fuente, no será necesario que exista ninguna librería instalada en el sistema.

Capítulo 5

Conclusiones

El objetivo inicial del proyecto era, como se ha detallado a lo largo del documento, construir una herramienta capaz de transmitir musicalmente lo que se percibe de una imagen. Tras 9 meses de proceso de desarrollo, se puede afirmar que se ha desarrollado de forma satisfactoria una aplicación capaz de llevar a cabo dicho objetivo, así como el resto de requisitos (tanto funcionales como no funcionales) mostrados en la Sección 3.3.

Para llevar a cabo la aplicación, ésta se ha dividido en dos módulos independientes que trabajan de forma secuencial: el módulo de análisis de imagen en primer lugar, y el módulo encargado de la composición musical en segundo; los cuales se pueden asemejar a la etapa de estimulación visual y a la etapa de estimulación auditiva, respectivamente. Esta distinción ha dado la posibilidad de estudiar y probar las etapas de manera independiente y así poder experimentar las diferentes formas de abordar la interpretación música-imagen. Cabe destacar que, dado que esta relación debe ser objetiva y no dependiente de consideraciones culturales o personales (por especificación del sistema), se ha elegido la sinestesia como método de asociación entre la percepción visual y la auditiva.

Por un lado, el módulo de análisis ha resultado ser capaz de transformar cualquier tipo de imagen de entrada en una representación propia del sistema de forma suficientemente fiel en un espacio de tiempo corto (segundos): dada una correcta configuración de parámetros por parte del usuario, las imágenes de grandes dimensiones pueden ser tratadas con menos nivel de detalle para aumentar su velocidad. Además, se puede cambiar la manera de reconocer formas alternando entre los distintos algoritmos de análisis, permitiendo al usuario experimentar con la forma en la que quiere que una imagen sea captada. La correcta elaboración de este módulo ha sido esencial para el desarrollo del proyecto ya que, aunque no sea el objetivo principal del mismo, proporciona la entrada al módulo de composición y por tanto es

una pieza clave para el perfecto funcionamiento de la generación de música.

Por otro lado, el módulo de composición ha cumplido con las expectativas mencionadas a lo largo del documento, siendo capaz de componer piezas musicales completamente originales que, si bien no buscan propiedades como ser "pegadizas", en ningún momento dejan de ser agradables al oído. Además, el módulo permite al usuario cambiar la forma de componer cada voz, así como los instrumentos con los que se interpretará cada una de ellas.

Con todo esto, la aplicación satisface la posibilidad de ser testeada por personas sinestésicas, de forma que éstas sean capaces de reconocer parcial o totalmente la similitud entre la imagen de entrada y la música generada. Con las diversas pruebas realizadas durante el desarrollo del proyecto, se ha determinado que la posibilidad de identificar la música creada a partir de las imágenes es factible, pero razonablemente limitada. Esto es debido a la cantidad de información que proporciona cada análisis de imagen, ya que sólo en las imágenes simples (con poca información) se puede anticipar la música generada. Por poner un ejemplo, una figura de cuatro vértices se asociará con cuatro notas. A cada figura interna dentro de ella se le asignará una melodía secundaría que sonará a la vez que la melodía de la figura padre. No hay que olvidar que, para añadir riqueza a la pieza musical, se añade una tercera voz (el bajo) y otra voz rítmica también basadas en las figuras presentes. Con todo esto, una figura de pocos vértices con unas pocas figuras dentro de ella tendrá asociada una pieza musical con cuatro voces tocando melodías distintas de una duración moderadamente corta a la vez. Por otro lado, una imagen analizada tiene del orden de un centenar de polígonos dentro de ella, cada uno de ellos con decenas de vértices. Puede imaginar el lector la complejidad que presentará la pieza musical generada, razón por la cual aumenta considerablemente la dificultad de comprender el origen de las notas que ha compuesto el algoritmo en cada momento.

5.1. Usos de la aplicación

Esta aplicación ha sido desarrollada teniendo en cuenta varios tipos de mercados (escenarios) y no sólo los pertenecientes al sector musical, ya que además de ellos se proponen otras alternativas. Es por ello que los usos para los cuales este sistema ha sido diseñado son:

- **Ayuda a la composición:** uno de los mayores problemas a la hora de componer es la búsqueda de inspiración para crear nuevas piezas musicales. Es por ello que un generador de música de esta índole, capaz de generar piezas originales y de un estilo razonablemente predecible (ya que depende de forma determinista de una imagen de entrada) es

de gran utilidad en este ámbito, siendo capaz de proporcionar ideas a los usuarios compositores. Con el objetivo de enfocar la aplicación a este aspecto, se ha establecido contacto con varios músicos quienes han mostrado su interés y admitido la posibilidad de usar la aplicación para tal fin. Durante sus experiencias, han preferido usar la aplicación con una gran variedad de imágenes, con la finalidad de abastecerse de piezas musicales variadas en lugar de intentar ver los diferentes efectos que puede tener el cambio de parámetros en una misma imagen, dado que el resultado sería una pieza musical similar a la pieza generada por primera vez. Sus opiniones han sido tremendamente positivas, apreciando la utilidad de dos aspectos de la aplicación: la creación de partituras, ya que sirve como base para componer una canción más complicada a partir de ella, y la reproducción de la música generada dentro del mismo entorno, gracias a la cual se puede comprobar rápidamente el resultado de una composición y se agiliza la realización de pruebas.

- **Composición visual:** en una rama puramente artística, la asociación de imagen-música proporcionada por esta aplicación puede servir como punto de partida de una corriente artística que consista en elaborar piezas gráficas con la intención de ser interpretadas como música por ésta o una aplicación similar.

- **Generación de música ambiente temática:** como se ha establecido en la introducción de este documento, la generación musical desarrollada no pretende componer piezas musicales que sean capaces de ser el foco de atención del usuario durante su interpretación, sino que es su objetivo generar una música de ambiente capaz de sonar de fondo mientras el usuario realiza otra actividad. Dado que se une la percepción visual con la musical, es una forma más de realidad aumentada que puede servir como hilo musical de fondo en museos (con melodías asociadas a cuadros), anuncios (melodías asociadas al logo de la marca promocionada) o distintas situaciones cotidianas.

- **Educación:** tanto los niños pequeños como los discapacitados (con enfermedades como el autismo o similares) sienten una gran conexión con la música y están bastante atraídos por ella. Mediante el uso de esta aplicación pueden aprender a crear música de forma sencilla y además divertida, al mismo tiempo que entrena su percepción visual, desarrollando por tanto los dos sentidos simultáneamente. Se ha comentado esta idea a personas dentro del sector educativo que han mostrado interés en probar esta aplicación en cursos con alumnos más pequeños.

5.2. Futuras ampliaciones

Una vez desarrollada la aplicación, y siendo ésta completamente funcional, cabe plantearse las posibles ampliaciones que se podrían realizar sobre el sistema para incrementar su funcionalidad y usos. Muchas de ellas parten de la idea inicial del proyecto, con la intención de proporcionar robustez al sistema o mejorar su funcionalidad; otras buscan dar al sistema un nuevo enfoque de uso. Estas ampliaciones son:

- **Realización de nuevos y distintos algoritmos de composición.** Aunque los algoritmos desarrollados se basan todos en una idea común (la sinestesia), existen muchas maneras de interpretar una imagen estática como una pieza musical que evoluciona en el tiempo. El proyecto ha sido diseñado para poder añadir nuevos algoritmos al código fuente de forma razonablemente sencilla. El proceso de desarrollo de estos algoritmos se basa intrínsecamente en el desarrollo y realización constante de pruebas, por lo que no existe una solución definitiva.

  Para ello, se puede partir de ayuda externa para producir ideas (o bien búsqueda y lectura de investigaciones llevadas a cabo sobre estos aspectos de la composición, o bien contacto con compositores profesionales). Además, es necesaria en la elaboración de nuevos algoritmos un post-proceso de testeo de la cualidad de los mismos, ya que la calidad de las piezas musicales generadas por un algoritmo no se puede demostrar con completa seguridad hasta la finalización de la implementación de los algoritmos.

- **Permitir la inclusión de nuevos algoritmos externos al código fuente.** Actualmente, la única manera de incluir nuevos algoritmos de composición o análisis es incluyéndolos en el código fuente de la aplicación. Una posible vía de desarrollo consistiría en dar la oportunidad al usuario de incluir nuevas formas de componer y analizar de forma externa, mediante nuevos módulos ejecutables o scripts que la aplicación sea capaz de lanzar.

- **Expandir el formato de representación de imágenes.** De forma que sea capaz de almacenar más información sobre la imagen. La aplicación desarrollada sólo es capaz de representar internamente polígonos y sus posiciones y colores; pero hay mucha más información dentro de una imagen que nos puede ser útil:

  - *Reconocimiento de simetrías:* una imagen puede tener diferentes tipos de simetrías entre los objetos/colores que la forman. Información de este tipo puede servir de base a nuevos algoritmos de

composición para que realicen melodías que repitan patrones o segmentos en función de las simetrías encontradas.

- *Reconocimiento de composiciones:* no es poco común que las figuras de una imagen estén dispuestas siguiendo formas geométricas básicas (círculos o triángulos, por poner dos ejemplos). Un ejemplo bastante famoso se puede observar en la Figura 5.1, donde se muestra la imagen de entrada a la izquierda y la composición que se pretende reconocer a la derecha. Imágenes con una disposición geométrica de figuras muy acentuada puede producir composiciones musicales cuya organización global sea equivalente a esta geometría detectada.

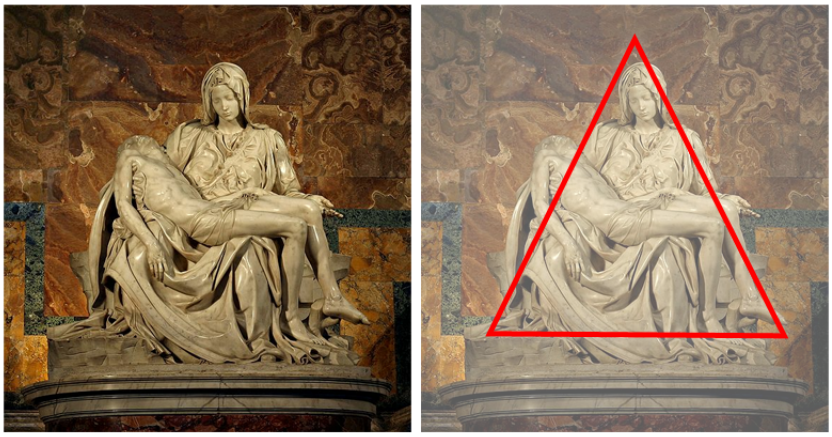


Figura 5.1: Ejemplo de reconocimiento de una composición triangular.

- *Reconocimiento de estructuras fractales:* existen actualmente compositores algorítmicos que toman como entrada los parámetros de una estructura fractal para generar piezas musicales acordes a ella. De forma más ambiciosa, el proceso de análisis de este sistema podría distinguir aquellos casos en los que las figuras de una imagen forman entre sí estructuras fractales, es decir, donde formas básicas se repiten en diferentes dimensiones, como se da en la Figura 5.2. Posteriormente, esta información se pasaría a un compositor que tenga en cuenta estas estructuras.
- *Ampliación de descripción del color de una figura:* la aplicación actual, con el objetivo de simplificar el proceso de análisis, sólo reconoce un color para cada figura reconocida. Sin embargo, la figura puede experimentar dentro de ella una variación de colores (aunque sea dentro de pequeños rangos), hecho que el sistema simplifica calculando el color medio. Esta ampliación consistiría

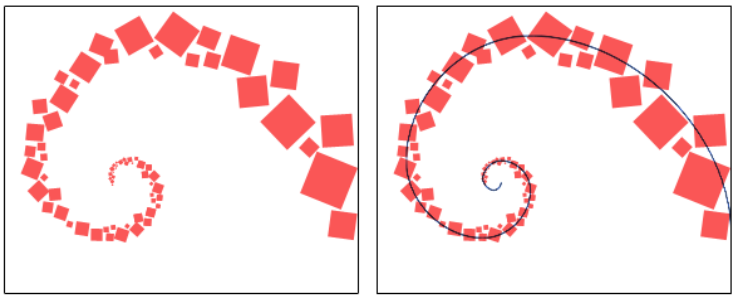


Figura 5.2: Ejemplo de reconocimiento de una estructura fractal.

en investigar una manera de detectar esa variación interna de color de forma que se consiga una representación más fiel de la figura.

- **Portar la aplicación a sistemas Apple.** Con esto se conseguiría expandir el número de posibles usuarios de la aplicación. Dado que las máquinas con sistema operativo de Apple funcionan con UNIX como base, esta ampliación puede resultar muy sencilla si se parte de la versión para Linux de la aplicación.

- **Portar la aplicación a sistemas móviles.** Una de las ideas iniciales que se debatieron antes de comenzar el desarrollo de la aplicación fue la de implementar el sistema como una aplicación móvil. Sin embargo, al realizar el estudio de alcance del proyecto y el tiempo de desarrollo del mismo, se desechó la idea. Sin embargo, una aplicación de esta índole puede alcanzar un gran éxito en estos dispositivos ya que las cámaras integradas de los mismos proporcionan una entrada gráfica fácil y sencilla, lo cual sumado a la facilidad de uso del sistema proporcionan a la aplicación un gran atractivo.

  El sistema se ha diseñado para facilitar la implementación de dicha ampliación, eligiendo librerías disponibles tanto en sistemas operativos de ordenadores personales como de dispositivos móviles (ver Sección A).

5.3. Seguimiento del proyecto

La aplicación, para cada una de las plataformas que ha sido desarrollada, se puede descargar desde aquí:

http://code.google.com/p/muphic/downloads/list

Algunas piezas musicales compuestas a partir de imágenes por el software desarrollado están disponibles en este canal de *youtube*:

http://www.youtube.com/user/mrmuphic

Con el tiempo se irán añadiendo más resultados de las diferentes composiciones relevantes o especiales que se vean convenientes.

Para contactar con el grupo de proyecto se puede hacer a través de la siguiente dirección:

imagebasedcomposing@gmail.com

Apéndices

Apéndice A

Tecnologías utilizadas

A.1. Notación ABC

En 1993, Chris Walshaw introdujo una nueva notación musical de texto plano, junto con abc2mtex (programa que traduce la notación ABC en notación que usa MusicTeX y TeX), [17]. Como resultado se obtuvo un lenguaje para escribir partituras de música. Poco después, Michael Methfessel con abc2ps y James Allwright followed con abcMIDI, empezaron a exportar la notación ABC en diferentes formatos.

Inicialmente ABC se creó para escribir música tradicional. Gradualmente se fueron añadiendo nuevas funcionalidades expandiendo sus posibilidades, pero ante todo se ha seguido preservando la mayor ventaja de esta notación: es fácil de crear, es ligera en almacenamiento (archivos de tamaño muy pequeño) y se puede compartir y transformar fácilmente, ya que es texto plano. Además, si está bien formateada la información se puede leer tras adquirir cierta experiencia.

Hoy en día hay un gran número de aplicaciones software que son compatibles con la notación ABC. Algunos programas comerciales permiten cargar o guardar la música producida en formato ABC. También existe la posibilidad de transformar del formato ABC a formato MIDI y viceversa. La oferta de software gratuito que maneja esta notación es amplia y variada.

Gracias a todas estas características, la notación ABC se ha adoptado como solución al problema que surge de traducir la representación interna de la música dentro del sistema a formato de audio MIDI. Para conseguirlo se transforma de la notación musical interna a notación ABC y de ésta a un archivo MIDI gracias a abc2midi, que forma parte del proyecto abcMIDI.

No es el propósito de ese documento realizar un tutorial exhaustivo de dicha notación, sin embargo se muestran en las Figura A.1 y A.2 dos ejemplos de lo fácil que es escribir notación musical usando este estándar.

Código:

```
L:1/64
CDEFGABcdefgabc'
```

Partitura asociada:

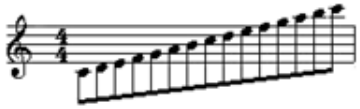


Figura A.1: Ejemplo de notación ABC

Código:

```
X:1
T:Paddy O'Rafferty
C:Trad.
M:6/8
L:1/8
K:D
dff cee|def gfe|dff cee|dfe dBA|dff cee|def gfe|faf gfe|
1 dfe dBA:|2 dfe dcB||
~A3 B3|gfe fdB|AFA B2c|dfe dcB|~A3 ~B3|efe efg|faf gfe|
1 dfe dcB:|2 dfe dBA||
fAA eAA|def gfe|fAA eAA|dfe dBA|fAA eAA|def gfe|faf gfe|
dfe dBA:|
```

Partitura asociada:

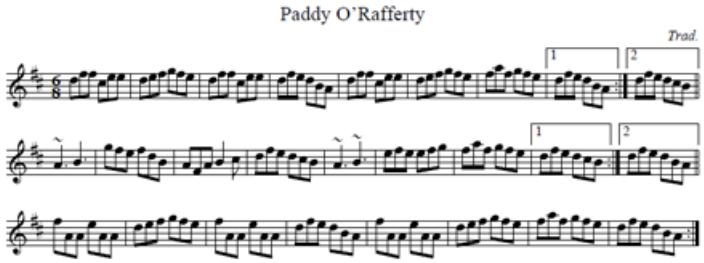


Figura A.2: Ejemplo de notación ABC

A.2. abcMIDI

abcMIDI consiste un paquete de programas desarrollado por James Allwright, [29]. Su cometido principal es procesar archivos con formato notación ABC. Este paquete contiene los siguientes programas: abc2midi, abc2abc, yaps, abcm2ps y midi2abc. En el proyecto se usan tanto abc2midi (para producir archivos de audio), como abcm2ps (para producir partituras).

abc2midi sirve para convertir archivos con formato notación ABC en archivos de sonido MIDI. Es probablemente el programa más avanzado y maduro de las diferentes posibilidades de software libre disponible en la red. Además contiene funcionalidades extra como el manejo de archivos con múltiples voces, trasponer voces individuales y añadir un acompañamiento de percusión entre otros.

abcm2ps tiene como funcionalidad producir una partitura musical partiendo del archivo en notación ABC de entrada. Esta partitura puede darse en numerosos formados, siendo dos de ellos ps y xhtml. A diferencia de su versión previa, abcm2ps, permite trabajar con varias voces; aunque a día de hoy existen unas pocas características de la notación ABC que no es capaz de transformar a la correspondiente partitura.

A.3. OpenCV

OpenCV es una de las librerías más importantes de visión por computador en tiempo real que existen por el momento. Fue desarrollada por Intel en 1999, y desde entonces se ha extendido su uso en numerosos sectores tecnológicos, [24].

Es una librería multiplataforma que ofrece funcionalidad para todo tipo de análisis y reconocimiento de imagenes, razón por la cual se ha elegido como base para la implementación del módulo de análisis de imágenes del proyecto.

A.4. TiMidity

Se trata de una herramienta de sintetización musical que puede reproducir archivos de audio en formato MIDI sin necesidad de un hardware sintetizador. La versión usada en este proyecto se denomina *TiMidity++*, y ha sido desarrollada por Masanao Izumo et al. basándose en *TiMidity 0.2i*, escrito por Tuuka Toivonen en 1995, [22].

Este software es capaz de convertir archivos MIDI a otros formatos, como pueden ser .wav, .au o .ogg si previamente se le proporciona información

sobre instrumentos los digitales que debe usar.

Es de gran utilidad en este proyecto para la creación de archivos de audio en formato .wav, ya que son los únicos que la interfaz gráfica puede reproducir. De esta forma el usuario puede escuchar de forma directa los resultados del proceso de composición sin tener que cambiar de aplicación.

A.5. TinyXML

TinyXML es un parser para C++ del lenguaje de etiquetas XML, el mismo lenguaje en el que están escritos los archivos de configuración del proyecto. Permite leer, manipular y escribir archivos XML de forma rápida y sencilla, [21].

A.6. Qt

Es un framework multiplataforma para el desarrollo de interfaces gráficas, creado inicialmente por Haavard Nord y Eirik Chambe-Eng, [28]. Esta librería usa C++ como lenguaje base, pero soporta programación con otros lenguajes.

Qt funciona sobre varias plaformas, incluídas entre ellas las máquinas portátiles y móviles. Es por esta razón que, de entre todas las tecnologías disponibles para el desarrollo de interfaces gráficas, se haya elegido *Qt* para el proyecto: el proceso de portar la aplicación a dispositivos móviles se simplifica enormemente si se usan las mismas herramientas que en la versión para ordenadores personales.

A.7. Phonon Multimedia Framework

Se trata de un framework que proporciona una API multimedia eficaz y fácil de usar, [27]. Se puede conectar fácilmente con Qt, proporcionando funcionalidades de reproducción de audio y video a las interfaces gráficas desarrolladas en esta librería.

Esta librería permite reproducir archivos de audio en formato .wav en la interfaz desarrollada para este proyecto. Esta característica, unida a la posibilidad de generar archivos MIDI gracias al paquete *abcMIDI* (ver Sección A.2) y la de transformar estos archivos MIDI en archivos de audio .wav gracias a *TiMidity* (ver Sección A.4), hace posible que la aplicación pueda ofrecer al usuario la opción de reproducir las composiciones musicales generadas.

Bibliografía

[1] André Pintado Jorge Gonçalves. *Sense*^{2} *A Music System based on Paintings*. Instituto Superior Técnico: Universidade Técnica de Lisboa Octubre 2009.
`https://dspace.ist.utl.pt/bitstream/2295/570110/1/dissertacao.pdf`

[2] Andrew J. Elliot (University of Rochester), Markus A. Maier (University of Munich), Arlen C. Moller and Ron Friedman (University of Rochester), and Jörg Meinhardt (Univeristy of Munich). *Color and Psychological Functioning: The Effect of Red on Performance Attainment*. Journal of Experimental Psychology: General, 2007, Vol. 136, No. 1, pp. 154 –168.

[3] Aristóteles. *De Sensu et Sensato*. Traducción Kevin White, 2005.
`http://www.josephkenny.joyeurs.com/CDtexts/SensuSensato.htm`

[4] *Brian Eno's Thoughts On Ambient Music*. Trouser Press interview, 1982.
`http://www.synthtopia.com/content/2009/09/17/brian-enos-thoughts-on-ambient-music/`

[5] Arshia Cont. *Artificial Intelligence and Music: A critical survey and proposal*. May 2005, University of California at San Diego.
`http://cosmal.ucsd.edu/arshia/papers/AI_Music_survey.pdf`

[6] David Eagleman. *Redes: Flipar en Colores*. Emisión Redes cap 22, 9 Febrero 2009.
`http://www.redesparalaciencia.com/249/redes/redes-22-flipar-en-colores-29-minutos`

[7] Dimitrios Margounakis, Dionysios Politis. *Converting Images to Music using their Colour Properties*. Proceedings of the 12th International Conference on Auditory Display, London, UK, June 20-23, 2006.
`http://www.dcs.qmul.ac.uk/research/imc/icad2006/proceedings/papers/f66.pdf`

[8] F. Pereira, C. Grilo, L. Macedo, and A. Cardoso. *Composing Music with Case-Based Reasoning*. International Conference on Computational Models of Creative Cognition, Dublin, 1997.
`http://iconline.ipleiria.pt/bitstream/10400.8/94/1/ICCMCC1997.pdf`

[9] G. Papadopoulos and G. Wiggins. *AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospect*. Proceedings of the AISB'99 Symposium on Musical Creativity, Edinburgh, UK, 1999.
`http://qmul.academia.edu/GeraintWiggins/Papers/201960/AI_Methods_for_Algorithmic_Composition_A_Survey_a_Critical_View_and_Future_Prospects`

[10] Gonzalo Pajares, Jesús M. de la Cruz. *Visión por Computador: Imágenes digitales y aplicaciones*. RA-MA, 2001.

[11] Guido d'Arezzo. *Micrologus* (traducido al inglés por Warren Babb). 1027.

[12] Ingrid Calvo Ivanovic. *Sinestesia Cromática*.
`http://www.proyectacolor.cl/significados-del-color/sinestesia-cromatica/`

[13] Isaac Newton. *Opticks*. 1704.
`http://www.gutenberg.org/ebooks/33504`

[14] Jack Sisson.*Creating Music from Chaos*. 12 de Junio de 2007, Math 53.
`http://www.math.dartmouth.edu/archive/m53f07/public_html/proj/Sisson.doc`

[15] Jhon A. Maurer. *A Brief History of Algorithmic Composition*. March, 1999.
`https://ccrma.stanford.edu/~blackrse/algorithm.html`

[16] John Biles. *GenJam: A Genetic Algorithm for Generating Jazz Solos*. 1994, pp. 131-137.
`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.6146`

[17] John Chambers. *ABC Music Notation*. 2009.
`http://trillian.mit.edu/~jc/music/abc/doc/ABCtut_Intro.html`

[18] K. Giannakis and M. Smith. *Imaging Soundscapes: Identifying Cognitive Associations between Auditory and Visual Dimensions*. Godoy, R. I., Jorgensen, H. (eds.): Musical Imagery, Swets & Zeitlinger, 2001, pp. 161-179.
`http://quod.lib.umich.edu/cgi/p/pod/dod-idx?c=icmc;idno=bbp2372.2000.167`

[19] Lauri Gröhn. *Sound of Paintings*.
`http://www.synestesia.fi/`

[20] Lawrence E. Marks. *The Unity Of The Senses*. 1978, pp. 93.

[21] Lee Thomason. *TinyXML: Main Page*. 2011.
`http://www.grinninglizard.com/tinyxmldocs/index.html`

[22] Masanao Izumo. *TiMidity++*. 2003.
`http://timidity.sourceforge.net/`

[23] Miguel Angel Mateu. *ARMONÍA PRÁCTICA vol.1*. Ab Música Ediciones Musicales, 2004. Valencia, España.
`http://es.scribd.com/arturo_vazquez_20/d/46628426-Armonia-Practica-Mateu-Vol-1`

[24] *OpenCV 2.1 C++ Reference*. 2010.
`http://opencv.willowgarage.com/documentation/cpp/index.html`

[25] KSAN. *Programa Web - Música a través de un dibujo*
`http://drawmusic.ksan.ru/`

[26] Nokia Corporation. *Qt Development Network: Phonon Module*. 2012.
`http://qt-project.org/doc/qt-4.8/phonon-module.html`

[27] Nokia Corporation. *Qt Development Network: Phonon Overview*. 2008.
`http://doc.trolltech.com/4.4/phonon-overview.html`

[28] Nokia Corporation. *Qt, the cross-platform application framework*. 2012.
`http://qt.nokia.com/downloads`

[29] Seymour Shlien. *abcMIDI*. 14 de Julio de 2003.
`http://abc.sourceforge.net/abcMIDI/`

[30] V. Lesbros. *From Images to Sounds: A Dual Representation*. Computer Music Journal 20 (3), 1996, pp. 59-69.
`http://www.jstor.org/discover/10.2307/3680824?uid=3737952&uid=2129&uid=2&uid=70&uid=4&sid=56229244313`

[31] V.S. Ramachandran & E.M. Hubbard. *Synaesthesia - A Window Into Perception, Thought and Language*. Journal of Conscousness Studies, 8, No.12, 2001, pp.3-34.

[32] Wassily Wassilyevich Kandinsky. *Concerning the Spiritual in Art*. 1911. Traducción Sadler, Michael T. H. Editorial Booksurge Llc, 2009.

[33] William Moritz. *The Dream of Color Music, and Machines that made it possible*. Animation World, Issue 2.1, April 1997,
`http://www.awn.com/mag/issue2.1/articles/moritz2.1.html`

[34] Wolfgang Köhler. *Gestalt Psychology.* Nueva York: Liveright, 1929.

[35] Xiaoying Wu, Ze-Nian Li. *A Study of Image-based Music Composition.* Multimedia and Expo, 2008 IEEE International Conference on.Hannover, June 23 2008-April 26 2008, pp. 1345-1348. `https://www.cs.sfu.ca/research/groups/VML/image_based_music/wu_icme08.pdf`