Asistente para composición de música minimalista

TRABAJO DE FIN DE GRADO

David Roldán Santos

Doble grado Matemáticas-Ingeniería Informática
Universidad Complutense de Madrid

13 de septiembre de 2016

Directores:

Jaime Sánchez Hernández
Marco Antonio Gómez Martín

Documento maquetado con TEXIS v.1.0.

Este documento está preparado para ser imprimido a doble cara.

Resumen

Podemos definir como música minimalista toda aquella música que es creada a partir de recursos limitados. Algunas de las características más comunes de esta corriente son el uso de armonías o ritmos constantes, reiteración de frases y utilización de transformaciones lentas.

La sencillez que caracteriza a este tipo de música nos permite analizar algunas de estas características computacionalmente. Entre ellas, podemos destacar la utilización de breves figuras melódicas, a las que llamaremos motivos, y el uso de variaciones musicales sobre estos.

Con el fin de analizar estas características, se ha desarrollado un asistente de composición musical con el que, a partir de uno o varios motivos, el usuario podrá crear composiciones musicales de carácter minimalista utilizando variaciones automatizadas. El asistente cuenta con una interfaz que permite gestionar los motivos con facilidad, mostrar su partitura, reproducirlos, aplicar variaciones sobre ellos y editarlos de forma manual.

Para la parte musical de la aplicación se ha utilizado notación `ABC`, una notación musical estandarizada muy completa que permite leer y editar música con facilidad.

Palabras clave

Música minimalista, notación `ABC`, motivo musical, variación musical

Abstract

Minimal music is a form of music that employs limited or minimal musical materials. It is characterised by repeated rhythm and harmony, repeated musical phrases, and the use of slow transformations.

The simplicity that characterizes this musical style allows us to analyze it from a computational point of view. Among these, we may remark the usage of short musical phrases called *motifs*, and the use of musical variations.

Looking towards the analysis of these characteristics, a minimal music assistant has been developed, where the user can apply built-in automatized variations to different motifs. The assistant provides a user interface that allows the user to manage these motifs easily, being able to play, show their music sheets, apply variations and edit them manually.

For the musical part of the application, `ABC` notation has been used. `ABC` notation is a powerful standardized notation designed to easily read and edit music.

Keywords

Minimal music, `ABC` notation, motif, variation

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

La música minimalista es una corriente musical catalogada como experimental. Podemos definirla como aquella música que utiliza una cantidad limitada de recursos musicales, caracterizada por el uso de armonías o ritmos constantes, reiteración de frases o transformaciones lentas [15].

El origen de esta corriente se sitúa en Estados Unidos en la década de los años 60. Es difícil establecer cuál fue la primera composición minimalista, ya que no se utilizó este término hasta que Michael Nyman lo utilizara en un artículo en 1968 [9]. La mayoría de las fuentes [8] [11] [13] consideran que las primeras obras minimalistas fueron *Trío para cuerdas* (1958) de La Mounte Young, y la obra *in C* (1964), de Terry Riley.

Dentro de esta corriene podemos destacar a varios compositores como Philip Glass, Steve Reich, Terry Riley, La Mounte Young y Kyle Gann. Este último identificó nueve características comunes en el diseño de la música minimalista [5]:

1. Armonía estática: tendencia a permanecer en un acorde, o a moverse en un repertorio de acordes.
2. Ritmo estático, ya sea motorizado o restringido a un cierto repertorio de duraciones temporales.
3. Repetición de breves motivos o breves figuras melódicas.
4. Procesos algorítmicos, lineales, geométricos o graduales.
5. Instrumentación estática.
6. *Metamúsica*: crecimiento de detalles sin planificar o amplificados por la percepción.
7. Uso de afinación justa.
8. Influencias no occidentales en las composiciones, como puede ser la música india o la música africana.

9. Esencialidad en la canción, sin detalles ocultos.

Por otro lado, se denomina composición algorítmica a la creación de música a partir de procesos formales o algoritmos. Existen varios modelos para este tipo de composición, como pueden ser modelos matemáticos, sistemas basados en el conocimiento, o sistemas de auto-aprendizaje [14]. La utilización de algoritmos dentro de la música permite crear composiciones de forma automatizada.

Podemos observar que algunas de las características de la música minimalista se asemejan bastante a la música algorítmica, como puede ser la utilización de ritmos motorizados, repetición de motivos, o la transformación de melodías mediante procesos algorítmicos. Esto nos puede llevar a la idea de poder automatizar composiciones de carácter minimalista, partiendo de breves melodías y utilizando transformaciones sobre éstas.

La principal dificultad que surge a la hora de generar música de forma automatizada es evaluar su componente estética, ya que es algo que generalmente no podemos traducir a un algoritmo. En nuestro caso, necesitaremos la intervención humana para decidir si el resultado de aplicar una transformación sobre algunos motivos musicales sigue manteniendo una componente estética satisfactoria.

El objetivo principal de este trabajo es crear un asistente para composición en el que el usuario pueda introducir varios motivos musicales, y a partir de variaciones automatizadas, crear una composición musical de carácter minimalista. Para ello, la herramienta cuenta con una interfaz gráfica donde el usuario puede gestionar los motivos, transformarlos y reproducirlos.

Para el desarrollo de la aplicación se ha utilizado la notación musical `ABC`, un lenguaje estandarizado en ASCII que permite leer y editar música con facilidad. El asistente permitirá leer y guardar archivos que estén en este formato.

No se requieren conocimientos musicales previos para la utilización del asistente musical, exceptuando algunas nociones básicas que serán abordadas en el próximo capítulo. Sin embargo, para entender la implementación de la herramienta, se necesitan conocer algunos conceptos musicales más avanzados, por lo tanto esta memoria será más accesible para aquellas personas que hayan tenido algún tipo de formación musical previa.

En el Capítulo 3 se hará un recorrido sobre el asistente musical, donde se mostrará la interfaz gráfica y cómo utilizar la herramienta. En el Capítulo 4 se explicará como ha sido implementado el asistente. Finalmente, en el último capítulo se comentarán las conclusiones y posible trabajo futuro sobre la aplicación.

Como material complementario a la memoria se incluye un CD de música, en el que se podrán escuchar algunos motivos y composiciones musicales descritas en la memoria. El CD ha sido creado a partir de archivos `MIDI` exportados desde el asistente musical, y posteriormente procesados por un editor de audio externo para mejorar la calidad del sonido.

Capítulo 2

Conceptos musicales

En este capítulo, se introducirán una serie de conceptos musicales que serán útiles para poder entender el funcionamiento de la aplicación desarrollada. La intención es describir estos conceptos desde un punto de vista informático, de modo que el lector pueda entender e identificarlos sin la necesidad de tener un conocimiento musical elevado.

Algunos de estos conceptos no serán necesarios para poder trabajar con la aplicación en modo usuario, pero sí para entender parte de la implementación.

2.1. Música

Es difícil exponer con precisión una definición de música, ya que el concepto de esta ha ido evolucionando con el tiempo. Una posible definición sería una manifestación artística que consiste en dar un conjunto de sonidos y silencios, de forma que el resultado sea coherente y agradable para el oído [16]. El fin de este arte es provocar una experiencia estética en el oyente, ya que la música puede ser utilizada como un medio de expresión de ideas, sentimientos o pensamientos. La música se puede descomponer en dos elementos básicos: los sonidos y los silencios.

- El *sonido* es la sensación percibida por el oído al recibir las variaciones de presión generadas por el movimiento vibratorio de los cuerpos sonoros. El sonido tiene cuatro cualidades básicas, que son las siguientes:
    - La *altura* es la frecuencia a la que se propaga un sonido, es decir, la cantidad de ciclos por segundo que se emiten. Podemos definir los sonidos como graves y agudos, de tal forma que, cuanto mayor sea la frecuencia, más agudo será el sonido.
    - La *duración* se corresponde con el tiempo en el que se mantiene un sonido.

- La *intensidad* es la cualidad que permite distinguir entre un sonido suave y un sonido fuerte.
- El *timbre* es una cualidad del sonido que permite diferenciarlo de otro que tenga la misma altura, duración e intensidad.

Cuando la altura de un sonido es constante, podemos referirnos a ese sonido como una *nota musical*. Si dos o más notas suenan de forma simultánea, estas forman un *acorde*.

- El *silencio* es la ausencia de sonido.

2.2. El sistema musical: escalas y tonalidad

El sistema de afinación musical más utilizado en todo el mundo es el llamado sistema temperado. En este sistema se obtiene una gama total de doce sonidos diferentes uniformemente espaciados. La distancia mínima entre cada uno de estos sonidos se llama *semitono*, y por tanto se llama *tono* al doble de la distancia de un semitono. Definimos como *octava* a una distancia de doce semitonos. Podemos decir que, si dos notas se encuentran a una distancia de doce semitonos o un múltiplo de 12, estas dos notas son la misma nota, pero están en octavas distintas.

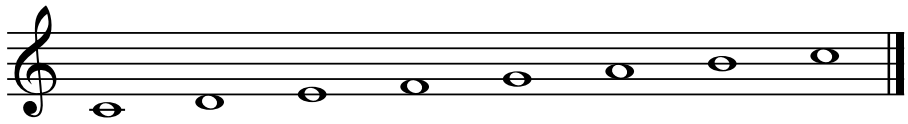
Las notas se agrupan siguiendo un sistema al que se denomina *tonalidad*. Los elementos básicos de la tonalidad son las *escalas*, que están compuestas por siete notas musicales. Existen varios tipos de escalas dependiendo de cómo estén organizadas las notas que las componen. El *modo* de una escala es el encargado de definir la organización de las notas de la escala. Aunque existen muchos modos, nos centraremos principalmente en los dos modos más comunes: modo mayor y modo menor:

- La *escala mayor* está distribuida de la siguiente forma: tono, tono, semitono, tono, tono, tono y semitono. Es decir, la tercera y cuarta nota de la escala están a una distancia de un semitono, y la séptima con la primera están también a una distancia de un semitono. El ejemplo más común es la escala de do mayor, mostrada en la Figura 2.1, donde tanto Mi y Fa como Si y Do están a una distancia de un semitono.

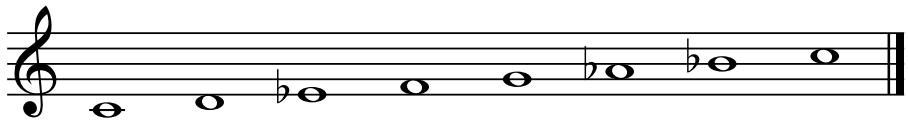
- La *escala menor* está distribuida de la siguiente forma: tono, semitono, tono, tono, semitono, tono, tono.

Se llama *grado* a la posición de cada nota dentro de una escala musical. Por ejemplo, en la escala de do mayor, los distintos grados corresponden a las siguientes notas:

Grado	Nota
I (tónica)	Do
II (supertónica)	Re
III (mediante)	Mi
IV (subdominante)	Fa
V (dominante)	Sol
VI (superdominante)	La
VII	Si



(a) Escala de do mayor, formada por la siguiente secuencia de notas: Do, Re, Mi, Fa, Sol, La, Si y Do.



(b) Escala de do menor, formada por la siguiente secuencia de notas: Do, Re, Mi♭, Fa, Sol, La ♭, Si ♭ y Do.

Figura 2.1: Escalas de do mayor y do menor

Una *triada* es un acorde formado por tres notas. Aunque una triada puede ser un acorde con tres notas cualesquiera, generalmente nos referimos con este nombre al acorde formado por el I, III y V grado. Llamamos triada mayor al acorde formado por el I, III, y V grado de una escala mayor, y triada menor al acorde formado a partir de una escala menor.

2.3. Intervalos

Un *intervalo* mide la distancia o altura entre dos notas musicales. El nombre de un intervalo viene dado por el número de grados que hay entre las dos notas musicales, y por su especie, que puede ser mayor, menor, justa, aumentada o disminuida.

Se denominan intervalos simples a aquellos intervalos no mayores a una octava. En la Tabla 2.1 podemos observar la equivalencia entre los intervalos simples y su distancia en tonos y semitonos.

Intervalo	Distancia en tonos y semitonos
Unísono	0 semitonos (mismo sonido)
Segunda menor	un semitono
Segunda mayor / tercera disminuida	un tono
Tercera menor / segunda aumentada	un tono y un semitono
Tercera mayor / cuarta disminuida	2 tonos
Cuarta justa / tercera aumentada	2 tonos y un semitono
Cuarta aumentada / quinta disminuida	3 tonos
Quinta justa / sexta disminuida	3 tonos y un semitono
Sexta menor / quinta aumentada	4 tonos
Sexta mayor / séptima disminuida	4 tonos un semitono
Séptima menor / sexta aumentada	5 tonos
Séptima mayor	5 tonos un semitono
Octava justa	6 tonos

Tabla 2.1: Equivalencia entre intervalos y distancia en tonos y semitonos.

Por ejemplo, el intervalo entre un do y un la es una sexta mayor, ya que hay una distancia de seis grados (Do-Re-Mi-Fa-Sol-La), y hay una distancia de cuatro tonos y un semitono.

2.4. Motivos musicales y formas musicales construidas a partir de motivos

Un motivo musical es una unidad de construcción breve, fácilmente reconocible debido al impacto temático que tiene en una obra musical. Es utilizado como un punto de partida para la construcción de unidades más extensas y generalmente tiene una formación rítmica y melódica característica.

Una de las primeras formas musicales donde se puede detectar la repetición de motivos es en el *canon*. El canon es una pieza musical basada en la imitación entre dos o más voces separadas por un intervalo temporal. Un ejemplo de canon muy conocido es el *Canon en Re Mayor*, de Johann Pachelbel.

Ya en el siglo XVII aparecieron formas musicales más complejas como la *fuga*, que consiste en el desarrollo de un motivo principal llamado sujeto a partir del uso de transformaciones, y utilizando fragmentos libres en las repeticiones, y las *invenciones*, que fueron introducidas por J.S. Bach, en las cuales todo el material de la composición deriva de un motivo principal. En la Figura 2.2 se puede observar cómo el motivo original sufre una serie de transformaciones a lo largo del tema. A esas transformaciones las llamaremos *variaciones*.

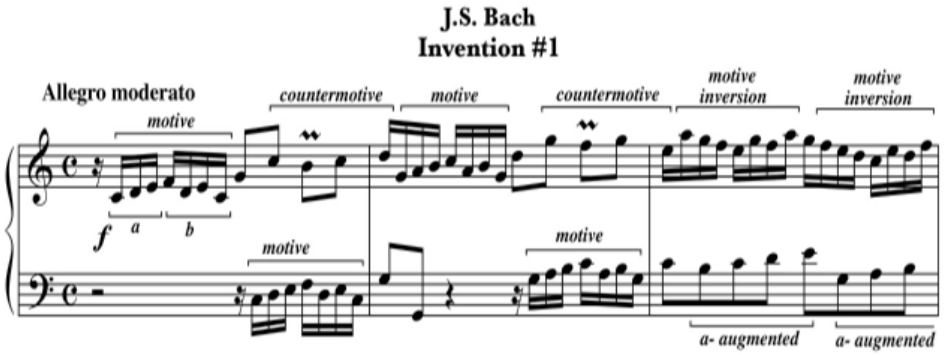


Figura 2.2: Comienzo de la Invención 1, de J.S. Bach.

2.5. Variaciones musicales

Como acabamos de ver, una variación es un recurso musical que consiste en la repetición con ciertas modificaciones de un motivo musical. Podemos clasificar las variaciones en tres tipos [3]:

- Variación por ornamentación o melódica: es aquella en la que el tema sufre transformaciones melódicas, pero sigue conservando toda su esencia melódica y armónica.
- Variación por elaboración o armónico-contrapuntística: es aquella en la que la melodía permanece inalterada o casi inalterada, pero sufre cambios en el ritmo o la armonía.
- Variación por amplificación: el motivo original se utiliza para crear nuevas melodías y armonías a partir de este.

En la Figura 2.3 podemos ver un ejemplo de variación denominado transporte. Esta variación consiste en trasladar la altura de cada una de las notas a la misma distancia, obteniendo como resultado una interpretación del motivo original en una tonalidad distinta.

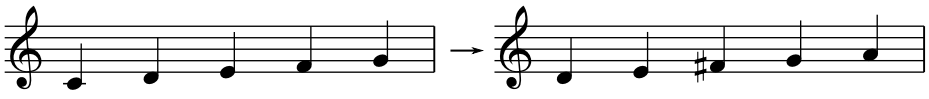


Figura 2.3: Ejemplo de transporte sobre un motivo.

Un ejemplo de una obra creada a partir de variaciones musicales es la obra *Variaciones en Do Mayor (KV265)*, de Wolfgang Amadeus Mozart,

que consta de doce variaciones sobre la canción francesa *Ah! vous dirai-je, Maman*, también conocida como la canción infantil *Campanitas del lugar*.

Para este proyecto, se han seleccionado una serie de variaciones que puedan ser automatizadas. Algunas de estas variaciones serán deterministas, pero otras tendrán una componente aleatoria. En el próximo capítulo veremos qué transformaciones han sido implementadas en la aplicación y un ejemplo de cada una de ellas.

Capítulo 3

Asistente musical

A continuación vamos a introducir la aplicación que ha sido desarrollada para este trabajo: un asistente que permite la creación de composiciones musicales mediante el uso de variaciones automatizadas. En este capítulo haremos un recorrido sobre la aplicación, viendo su interfaz gráfica y las funciones que puede realizar.

3.1. Ejecutando el asistente musical

El asistente de música minimalista sólo es compatible con Windows 7 o superior. Para poder ejecutar la aplicación se debe instalar previamente .NET Framework 4.5, y Ghostscript.

Para ejecutar el asistente musical, simplemente debemos abrir el archivo *MusicaMinimalista.exe*. En el directorio raíz, podemos encontrar varios archivos `.dll` que son necesarios para poder ejecutar la aplicación.

Dentro de este directorio, podemos encontrar las siguientes carpetas:

- *bin*: contiene varios archivos ejecutables que el asistente musical utilizará durante su ejecución.
- *files*: carpeta que se utilizará por defecto para almacenar archivos en formato `ABC`.
- *midi*: carpeta que se utilizará por defecto para guardar archivos de audio en formato `MIDI` que sean exportados por el asistente.
- *pdf*: carpeta que se utilizará por defecto para guardar partituras en formato `PDF` que sean exportadas por el asistente.
- *projects*: carpeta que se utilizará por defecto para guardar los proyectos creados con el asistente.
- *temp*: carpeta que el asistente musical utilizará para crear archivos temporales.

3.2. Interfaz gráfica

Para crear la interfaz de usuario se han tomado principalmente dos referencias: *LMMS* [6] y *FL Studio* [12], que son editores musicales cuyo área de trabajo principal es un *secuenciador basado en patrones*. Un secuenciador basado en patrones es una herramienta que representa una línea temporal, en la que se pueden insertar fragmentos musicales (patrones) de forma ordenada para crear una composición musical.

En nuestro caso, contaremos con un área de trabajo parecida a estos secuenciadores, donde podremos insertar los motivos musicales que vayamos generando para crear una composición musical de carácter minimalista.

Al ejecutar la aplicación, se muestra la pantalla principal, que podemos ver en la Figura 3.1. La pantalla principal cuenta con tres componentes:

- Una *barra de menú principal*, desde donde se pueden cargar y guardar proyectos, cargar y crear motivos, y exportar motivos en distintos formatos.
- Un *árbol de motivos*, que se encuentra en la parte izquierda de la pantalla principal. Este árbol incluye todos los motivos musicales que están cargados en el proyecto.
- En la parte derecha tenemos un *secuenciador*, que es el área principal de edición. El secuenciador está formado por varias pistas en las que se pueden insertar los motivos musicales que están en el árbol de motivos.

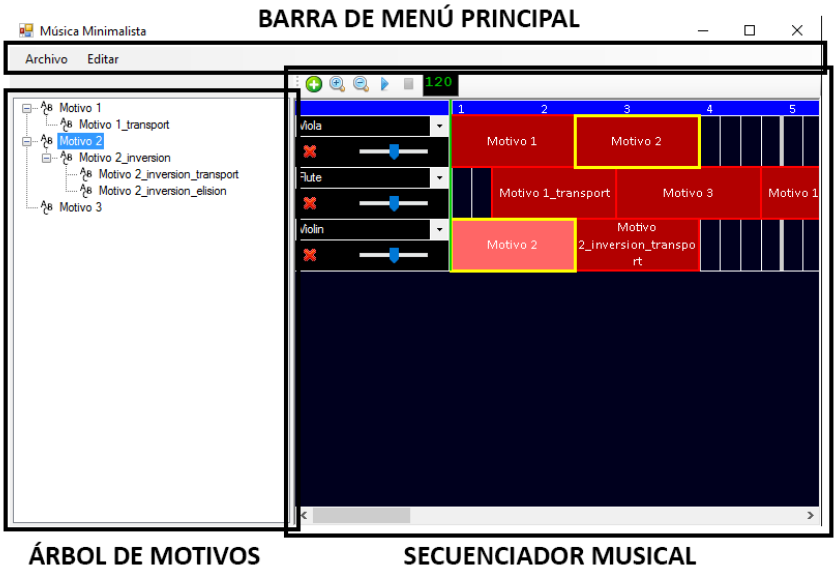


Figura 3.1: Pantalla principal del asistente musical

3.2.1. Creando un nuevo proyecto

Un *archivo de proyecto* es un archivo que contiene toda la información acerca de una composición creada con el asistente musical. A esta información la llamaremos *proyecto*.

Para crear un proyecto nuevo, hacemos click sobre la pestaña de menú de archivo de la barra de menú principal, y elegimos la opción de *Nuevo proyecto*.

En el Apéndice A podemos ver un ejemplo de uso de la aplicación.

3.2.2. Barra de menú principal

La barra de menú principal cuenta con dos menús desplegables: el menú de archivo y el menú de edición.

El *menú de archivo* permite la creación, carga y guardado de archivos proyectos, la carga y la creación de nuevos motivos, y la exportación de proyectos en distintos formatos.

En la Figura 3.2 podemos ver los comandos que tiene el menú de archivo.

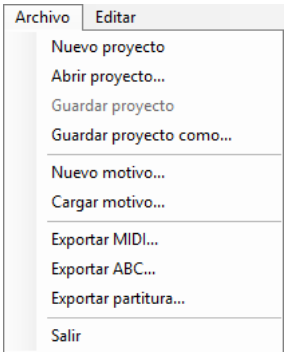


Figura 3.2: Menú de archivo

Por otro lado, el *menú de edición* permite deshacer y rehacer acciones realizadas, como por ejemplo, cargar un motivo, eliminar un motivo, o subir el volumen de una pista del secuenciador.

3.2.3. Árbol de motivos

Como acabamos de ver, el árbol de motivos muestra todos los motivos musicales que están cargados en el proyecto. Se utiliza una representación en forma de árbol para poder diferenciar los motivos que han sido obtenidos a partir de otros motivos mediante variaciones musicales, y los motivos que han sido creados o cargados a partir de un archivo.

En la Figura 3.3 podemos ver un ejemplo de árbol con tres motivos. En este árbol, *Motivo3* ha sido creado a partir de una variación de *Motivo1*.

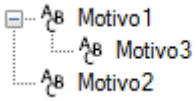


Figura 3.3: Ejemplo de árbol de motivos.

Al hacer click con el botón derecho sobre un motivo, aparece un menú contextual tal y como se muestra en la Figura 3.4. Este menú tiene las siguientes opciones:

- Reproducir: reproduce el motivo seleccionado.
- Renombrar: permite cambiar el nombre del motivo por otro nombre que todavía no esté en uso.
- Editar motivo: abre una ventana de edición en la que se puede editar manualmente el motivo en formato `ABC`. Veremos la ventana de edición de motivos con más detalle en la Subsección 3.2.4.
- Variar: crea un nuevo motivo a partir de una variación del motivo seleccionado.
- Dividir: crea un motivo por cada una de las voces que tiene el motivo seleccionado.
- Duplicar: crea una copia del motivo seleccionado.
- Exportar ABC: exporta el motivo en un archivo en formato `ABC`, que veremos con más detalle en la Sección 3.3.
- Eliminar: elimina el motivo del árbol de motivos. Los hijos del motivo eliminado pasan a ser hijos del padre. En caso de que el motivo eliminado estuviera en la raíz del árbol, los hijos pasarían a la raíz.

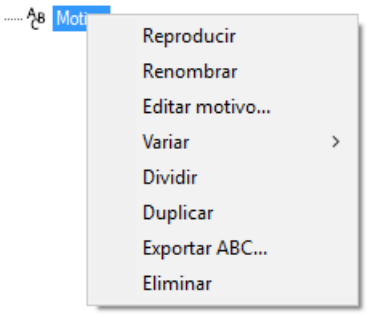
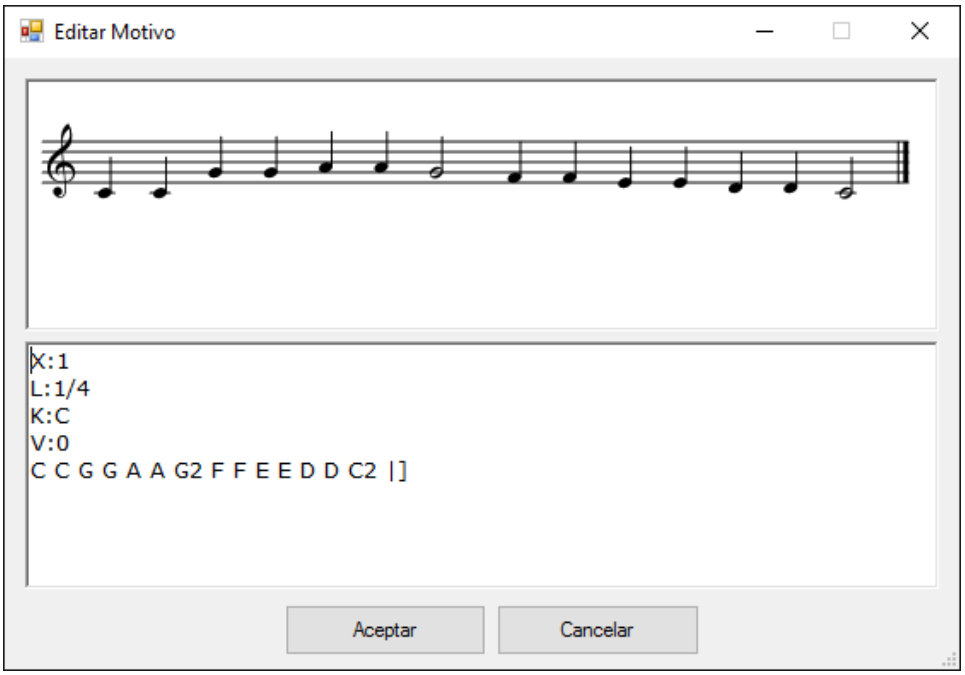


Figura 3.4: Menú contextual de un motivo.

3.2.4. Ventana de creación y edición de motivos

La ventana de creación y edición de motivos es accesible a través del menú contextual del árbol de motivos, o a través de la barra de menú principal. Esta ventana tiene el siguiente aspecto:



Como se puede ver en la imagen, la ventana cuenta con dos paneles: el panel superior muestra una vista previa de la partitura del motivo que se está editando. El panel inferior permite la edición del motivo en formato `ABC`. La vista previa se actualiza en tiempo real mientras se modifica el texto `ABC`. También detecta errores de sintaxis, por lo que el cuadro de edición solo permitirá guardar los cambios del motivo creado si el texto `ABC` está escrito correctamente.

En la Sección 3.3 veremos con más detalle qué es la notación `ABC`.

3.2.5. Secuenciador musical

El secuenciador musical es el componente principal en el que se centra toda la aplicación. El secuenciador estará formado por una o varias pistas, en las que se pueden insertar los motivos que han sido generados previamente. En la parte superior del secuenciador hay una barra de menú que tiene los siguientes elementos:

-  : añade una nueva pista al secuenciador.

-  y  sirven para hacer zoom sobre el secuenciador. También se puede hacer zoom utilizando la rueda del ratón.
-  : reproduce el contenido actual del secuenciador.
-  : pausa la reproducción.
-  : indica el *tempo* o la velocidad de la pieza musical.

La utilización de pistas nos permite poder crear piezas musicales en las que suenen motivos de forma simultánea y con distintos instrumentos. Todos los motivos que estén en la misma pista estarán asociados al mismo instrumento y volumen, que podrá ser configurado en la cabecera de la pista como se muestra en la Figura 3.5.

Figura 3.5: Pista del secuenciador musical.

Para añadir un motivo a una pista, se puede tanto arrastrar desde el árbol de motivos, como copiar y pegar desde el propio secuenciador. La copia y eliminación de motivos del secuenciador se pueden realizar a través de un menú contextual, que además cuenta con las siguientes funciones:

- Desligar: el motivo seleccionado se desliga de la rama del árbol de motivos de donde procede. Para ello, se crea una copia del motivo en la raíz del árbol.
- Dividir: divide un motivo en voces. Para ello, se crea un motivo por cada voz del motivo original, y se colocan en pistas distintas.

Otra funcionalidad importante dentro del secuenciador es la posibilidad de arrastrar motivos a través de la línea temporal, o a otras pistas. La aplicación no permite que haya dos motivos solapados en una misma pista, si se desea escuchar dos motivos de forma simultánea, se necesitará tener dos pistas y situar cada uno de los motivos en una de ellas.

3.3. Notación ABC

La notación `ABC` es una notación musical estandarizada que utiliza únicamente caracteres ASCII, diseñada para que sea fácil de leer y editar para una persona, y a su vez fácil de procesar para un programa.

```
X:1
T:Happy Birthday
L:1/4
M:3/4
K:G
V:O
D3/4D/4   | E D G |
F2 D3/4D/4 | E D A | G3 |]
V:1
z | G,, [G,B,D] [G,B,D] | D,, [D,F,A,C] [D,F,A,C] |
D,, [D,F,A,C] [D,F,A,C] | G,, [G,B,D] [G,B,D] |]
```

Figura 3.6: Ejemplo de archivo en formato ABC.

En la Figura 3.6 podemos ver un ejemplo de archivo en formato ABC. La estructura del archivo está dividida en dos partes:

- La cabecera proporciona información general como por ejemplo el nombre del autor, tempo o tonalidad. La cabecera debe comenzar siempre con un número de referencia (campo X). En el ejemplo anterior podemos ver que el campo T es el título de la canción, el campo M la métrica, y el campo K la tonalidad.

- El cuerpo del archivo contiene todos los elementos que aparecen en la partitura, como por ejemplo notas, silencios, barras o adornos. La notación ABC utiliza el cifrado anglosajón[1] para identificar cada una de las notas, seguido de los símbolos , o ' para indicar la octava de la nota, y seguido de un número para indicar su duración. Los silencios se representan con una z. Para representar un acorde se escriben varias notas entre corchetes.

  El archivo ABC no debe contener ninguna línea en blanco.

  La versión de ABC utilizada en la aplicación cuenta con varias limitaciones, que describiremos en la Subsección 4.2.3. Para más información sobre notación ABC, se puede consultar el estándar de ABC 2.1 [1].

  En la Figura 3.7 podemos ver una representación en forma de partitura del archivo anterior.

[1] El sistema de notación musical anglosajón consiste en identificar cada una de las notas con una letra. Se hace la siguiente asignación: C=Do, D=Re, E=Mi, F=Fa, G=Sol, A=La, B=Si.

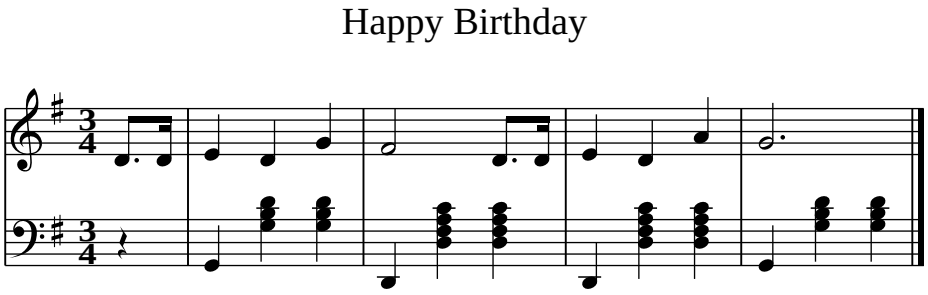
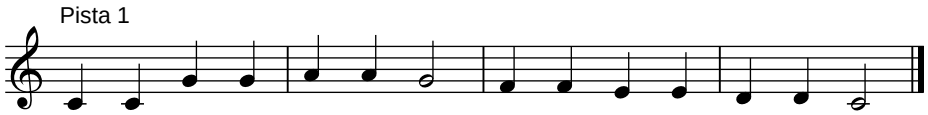


Figura 3.7: Representación en forma de partitura de la Figura 3.6.

3.4. Variaciones de motivos

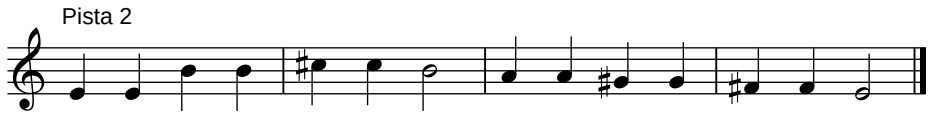
Como vimos en el capítulo anterior, una variación es un recurso musical que consiste en la repetición con ciertas modificaciones de un motivo. Dentro de la aplicación, podemos aplicar variaciones a través del menú contextual del árbol de motivos que hemos visto anteriormente. A continuación vamos a describir cada una de las variaciones que se pueden utilizar. Para ello, vamos utilizar el siguiente ejemplo, que será a partir del cual se obtengan todas las variaciones:



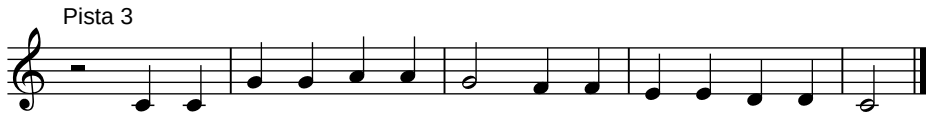
Todos los motivos de está sección están disponibles en el CD de música incluido con la memoria. La pista del CD de cada motivo está indicada en la parte superior izquierda de su partitura.

Las variaciones que el asistente musical puede realizar son las siguientes:

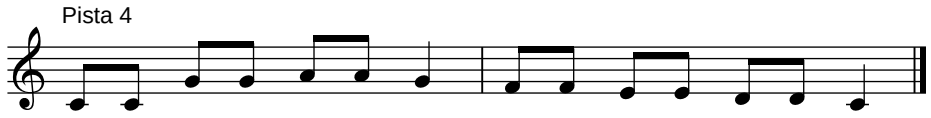
- *Transporte*: consiste en modificar la altura de las notas por igual, de tal forma que el motivo resultante es una interpretación del motivo original en una tonalidad distinta, manteniéndose la relación interválica entre sus notas. Esta variación recibe como parámetro la distancia del transporte, medida en semitonos. A continuación se muestra el motivo originado tras aplicar un transporte de cuatro semitonos sobre el motivo original:



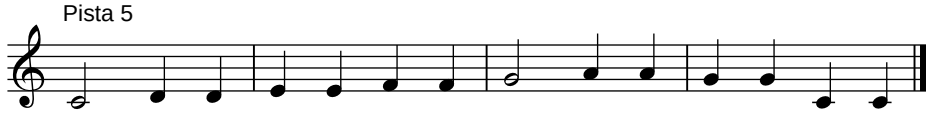
- *Retardo*: imitación de un motivo separado por un intervalo temporal. Esta variación recibe como parámetro la duración del retardo aplicado. Como ejemplo, el motivo originado tras aplicar un retardo con una duración de dos negras sobre el motivo original es el siguiente:



- *Aumentación y disminución*: consiste en multiplicar o dividir los valores de las figuras por una cantidad fija, de modo que se produce una aceleración o ralentización del motivo. A continuación se muestra el motivo originado tras reducir la duración de cada nota del motivo original a la mitad:

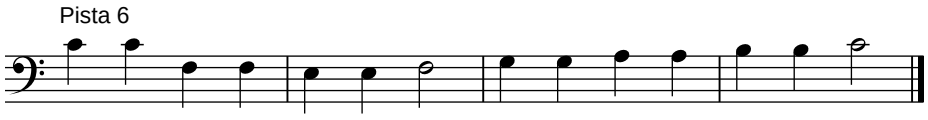


- *Retrogradación*: lectura de un motivo de atrás hacia adelante. Si aplicamos esta transformación sobre el motivo inicial, obtenemos el siguiente motivo:



- *Inversión*: también llamado espejo, consiste en transformar un motivo de tal forma que el motivo resultante sea un reflejo o simetría horizontal del motivo original. La inversión se realiza sobre un eje de simetría

situado en la tónica de la tonalidad, en caso de que la tonalidad sea mayor, y en el tercer grado, en caso de que la tonalidad sea menor. A continuación se muestra el motivo originado tras aplicar una inversión sobre el motivo original:



En este ejemplo, el eje de simetría se sitúa sobre la nota Do, ya que la tonalidad del motivo es Do Mayor.

- *Inversión de Rachmaninoff*: la inversión de Rachmaninoff es un caso especial de inversión donde se conserva la separación de semitonos entre la secuencia de notas. Esta inversión característica fue utilizada por el compositor *Sergei Rachmaninoff* en su obra llamada *Rapsodia sobre un tema de Paganini*, como se muestra en la Figura 3.8. Esta pieza musical es un conjunto de 24 variaciones sobre el *Capricho nº24* de *Niccolo Pagainini*, entre las que se encuentra esta inversión.

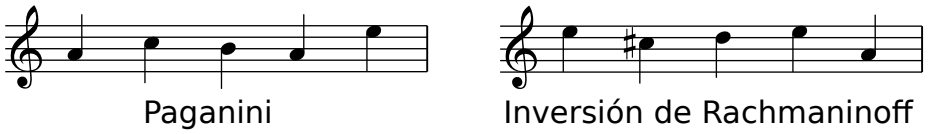
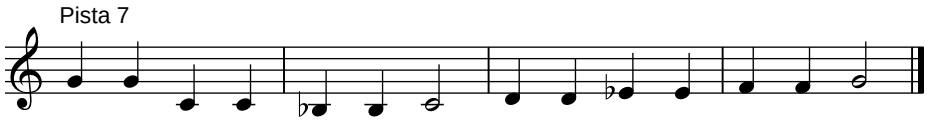


Figura 3.8: Inversión de Rachmaninoff aplicada al motivo original de Paganini.

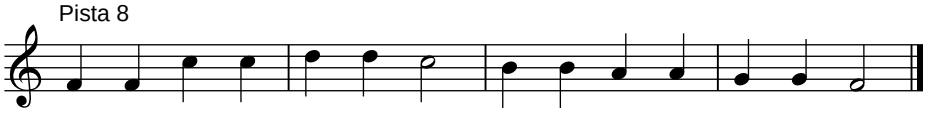
A continuación, se muestra el motivo resultante tras aplicar una inversión de Rachmaninoff sobre el motivo inicial que hemos utilizado de ejemplo:



- *Transporte tonal*: el transporte tonal consiste en modificar la altura de las notas del motivo, de tal forma que las notas que pertenecen a la tonalidad siguen perteneciendo a la tonalidad. Esta variación recibe

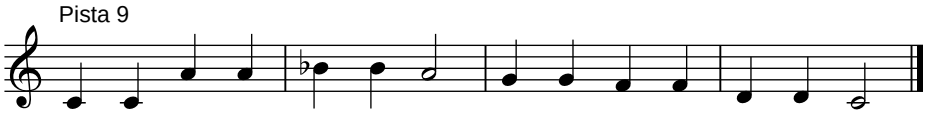
como parámetro el número de grados de la escala que se va a transportar. A continuación, se muestra el motivo originado tras aplicar un transporte tonal de tres grados de la escala sobre el motivo original:

Pista 8

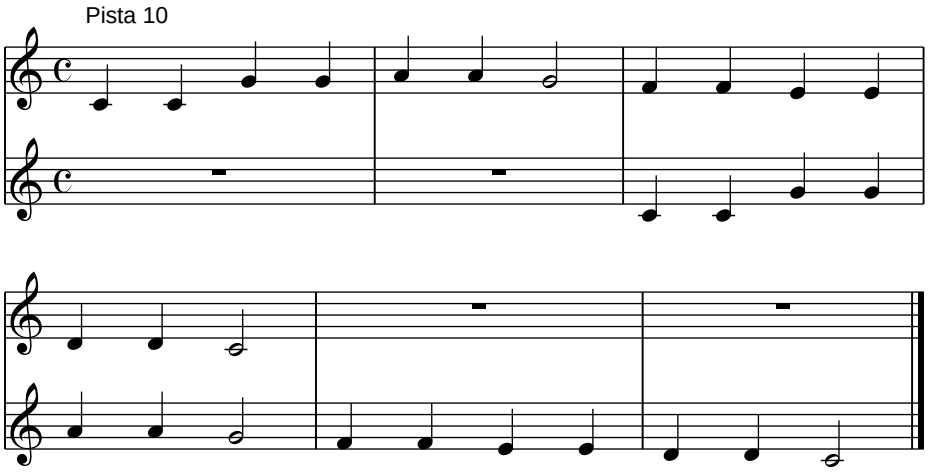


- *Modulación estática*: esta variación consiste en modificar la tonalidad de un motivo, desplazando la altura de las notas del motivo lo mínimo necesario, de tal forma que las notas que pertenecían a la tonalidad original pertenezcan a la nueva tonalidad, y en particular, que las notas que pertenecían a la triada fundamental de la tonalidad original pasen a pertenecer a la triada de la nueva tonalidad. La modulación estática recibe como parámetro la tonalidad a la que se quiere modular. En el siguiente ejemplo se muestra el motivo originado tras aplicar una modulación estática sobre el motivo original a Fa mayor:

Pista 9

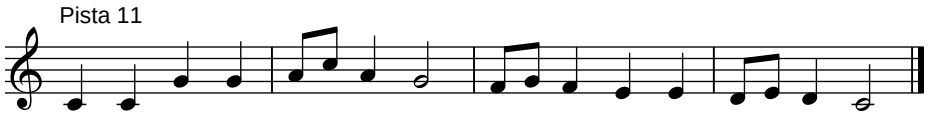


- *Canonización*: se aplica un retardo específico al motivo con el objetivo de minimizar la disonancia entre el motivo original y el motivo retardado. Veamos un ejemplo:

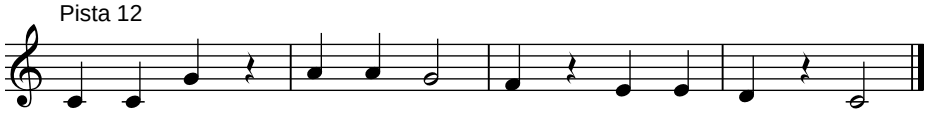


En este ejemplo, la primera voz se corresponde con el motivo original, y la segunda voz el el motivo resultante tras aplicar una canonización sobre el motivo original.

- *Interpolación*: inserción de una o varias notas sobre un motivo. La notas añadidas sobre el motivo resultante son notas pertenecientes a la tonalidad elegidas al azar, dando más peso a las notas que pertenecen a la triada fundamental. En el ejemplo siguiente podemos ver el motivo originado tras aplicar una interpolación sobre el motivo original:

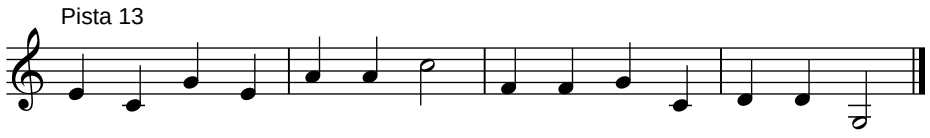


- *Elipsis*: omisión de una o varias notas de un motivo. La elección de las notas a omitir se realiza al azar. A continuación se muestra el resultado tras aplicar una elipsis sobre el motivo original:



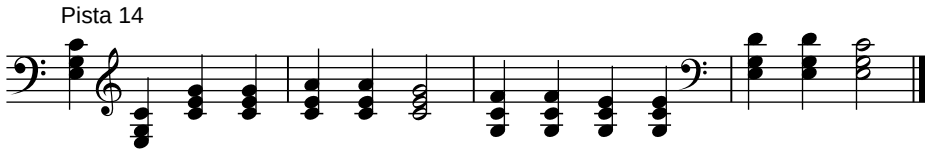
- *Permutación de la triada*: la permutación de triada consiste en identificar cada una de las notas que pertenecen a la triada fundamental de

la tonalidad, e intercambiarlas entre ellas de forma aleatoria. A continuación se muestra un ejemplo de permutación de la triada sobre el motivo original:



Como la tonalidad del motivo original es do mayor, la triada fundamental está formada por Do, Mi y Sol. Cada una de estas notas ha sido transformadas por otra nota perteneciente a la triada.

- *Armonización*: la armonización es una variación experimental que consiste en convertir las notas del motivo en acordes, utilizando notas de la triada fundamental de la tonalidad para completar dichos acordes. El funcionamiento de esta transformación mejorará si en un futuro se añade un análisis armónico sobre los motivos. A continuación podemos ver el resultado tras aplicar una armonización sobre el motivo original:



Capítulo 4

Desarrollo de la aplicación

4.1. Lenguaje de programación y tecnologías utilizadas

En este capítulo se abordará tanto la implementación del asistente musical como la parte algorítmica utilizada para crear las variaciones musicales.

4.1.1. C# y .NET Framework

El proyecto ha sido desarrollado en C# , un lenguaje de programación orientado a objetos desarrollado por Microsoft y estandarizado como parte de su plataforma *.NET*. El entorno de desarrollo escogido has sido *Microsoft Visual Studio*, ya que es el más utilizado para este lenguaje.

El único sistema operativo compatible con esta aplicación es Windows, ya que para la interfaz gráfica se ha utilizado Windows Forms, que proporciona acceso a los elementos de la interfaz de Windows, y para la reproducción de música se ha utilizado Windows Media Player, un componente exclusivo de Windows que permite ser añadido como control directamente a la interfaz de usuario de una aplicación Windows Forms.

Se ha utilizado una librería de C# para poder visualizar imágenes en formato `SVG`, obtenida a través del gestor de paquetes NuGet que está incluido dentro de Visual Studio. Esta librería ha sido útil para poder mostrar partituras dentro de la aplicación.

4.1.2. Notación ABC

Como ya hemos visto en el capítulo anterior, se ha utilizado la notación `ABC` para la parte musical de la aplicación. La principal ventaja a la hora de utilizar esta notación musical es que existen varios paquetes de software que permiten procesar y convertir estos archivos en archivos de audio, o representarlos en forma de partitura. Para esta aplicación, se han utilizado

dos de ellos: `abc2midi`, que es un conversor de `ABC` a `MIDI`, y `abcm2ps`, que convierte los archivos en partituras tanto en formato *PostScript* como en `SVG`.

4.1.3. ANTLR

Como acabamos de ver, se han utilizado varios paquetes de software para convertir archivos en formato `ABC` a archivos de audio y a partituras. Sin embargo, ninguno de estos paquetes nos permite convertir este tipo de archivo en la estructura interna de datos utilizada dentro de la aplicación. Para realizar esta tarea se ha utilizado `ANTLR` [10], una herramienta que permite crear compiladores y traductores de lenguaje a partir de una descripción gramatical del lenguaje de partida, y que tiene una versión disponible para C# . La versión de `ANTLR` utilizada para este proyecto es `ANTLR4`.

En la Subsección 4.2.3 veremos con más detalle cómo se ha realizado la importación y exportación de archivos `ABC` en la aplicación.

4.2. Estructura de datos interna de la aplicación

Existen múltiples posibilidades a la hora de crear una estructura de datos para representar música. Para poder elegir una estructura que se ajuste lo mejor posible a esta aplicación debemos recordar cuál es el principal objetivo del asistente: crear composiciones de música minimalista. Esto quiere decir que no nos bastará con crear una simple lista de notas para crear una pieza musical, sino que debemos agrupar esas notas en motivos, e incorporar esos motivos a otra estructura mayor para crear dicha pieza.

4.2.1. Motivos musicales

Como hemos visto en capítulos anteriores, un motivo musical es esencialmente un fragmento musical breve que puede derivar en otros motivos mediante variaciones. Debemos tener en cuenta las siguientes consideraciones:

1. Un motivo puede tener varias voces, es decir, un motivo puede estar formado por varias melodías que suenan a la vez.

2. Cada una de estas melodías o voces puede contener acordes.

Por lo tanto, se han creado los siguientes objetos para la representación de los motivos:

- El objeto `Note` servirá para representar las notas musicales. Este objeto tiene dos atributos: la duración, que está representada en forma de

fracción[1], y la altura (`pitch`), que es un entero. La representación de la altura es la misma que en la de los archivos MIDI, tomando cada nota un valor entre 0 y 127, siendo 127 el valor correspondiente a la nota más aguda. Adicionalmente se utiliza el valor -1 para representar los silencios.

- Los acordes se representan mediante el objeto `Chord`, que contiene una lista de notas. Tanto `Note` como `Chord` heredan de una clase abstracta llamada `MusicItem`.

- Para agrupar tanto notas como acordes en una melodía o voz se utiliza la clase `Voice`, que contiene una lista de `MusicItem`. Esta clase representa cada una de las voces que contiene un motivo musical.

- Por último, la clase `Motif` contiene los siguientes atributos:

  - Un entero `id`, que servirá para identificar de forma única a cada uno de los motivos.
  - El identificador del motivo padre `parentId`. Si el motivo ha sido cargado de archivo o creado de cero, el valor de `parentId` será -1.
  - Una lista de voces.
  - La variación que se ha utilizado para crear el motivo. Veremos la implementación de la clase `Variation` más adelante.
  - La tonalidad del motivo, que servirá como información adicional para aplicar ciertas variaciones. Aunque es posible determinar la tonalidad de un motivo mediante análisis armónico, realizar dicho análisis es complejo y queda fuera del alcance de este proyecto.

    Es posible determinar la tonalidad de un motivo mediante análisis armónico, sin embargo, realizar dicho análisis es complejo y se queda fuera del alcance de este proyecto. Como alternativa, la tonalidad del motivo vendrá dado y servirá como información adicional para aplicar ciertas variaciones.

En la Figura 4.1 podemos ver un esquema general de la representación interna de motivos en la aplicación.

En la sección de variaciones musicales, veremos algunos de los métodos que contiene esta clase y cómo aplicar variaciones a los motivos.

[1]El asistente musical utiliza como medida de referencia la negra. Por lo tanto, las duraciones de las distintas figuras son las siguientes: redonda = 4, blanca = 2, negra = 1, corchea = 1/2, semicorchea = 1/4, fusa = 1/8 y semifusa = 1/16.

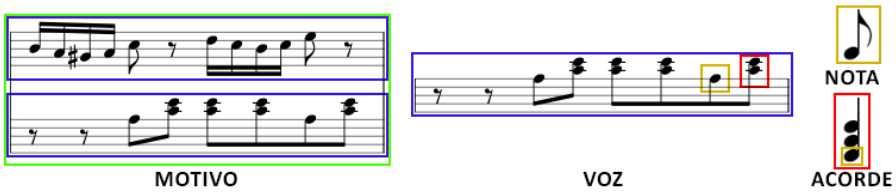


Figura 4.1: Esquema general de la representación interna de un motivo.

4.2.2. Composiciones musicales

Una vez creada la estructura para almacenar motivos, debemos crear otra estructura para poder crear composiciones musicales. Para ello, se debe tener en cuenta que la aplicación está basada en un secuenciador basado en patrones, que ya definimos en la Sección 3.2. Por lo tanto, se ha de crear una estructura que encaje con dicho diseño. A continuación vamos a ver las dos clases principales que se han creado para crear la estructura, la clase `Track` y la clase `Tune`:

- La clase `Track` sirve para representar cada una de las pistas del secuenciador. Dentro de esta clase tenemos los siguientes atributos:
  - El instrumento de la pista: es un tipo enumerado con el que podemos seleccionar cualquier instrumento compatible con `abc2midi`, que es el programa utilizado para traducir partituras en formato ABC a archivos MIDI.
  - El volumen de la pista, representado por un entero.
  - Una lista ordenada `SortedList` donde se guarda qué motivos tiene la pista y su posición. Para almacenar esta información se define un par clave-valor, donde la clave es la posición de inicio del motivo, y el valor es el identificador del motivo.
- La clase `Tune` es la clase principal del proyecto. Es aquí donde guardaremos toda la información acerca de los motivos que han sido creados y la posición de los motivos en las pistas del secuenciador. La clase contiene los siguientes atributos a destacar:
  - Un entero `tempo`, que indica el tempo o la velocidad a la que se va a reproducir la pieza musical. Cuanto mayor sea este valor, más rápida será la velocidad de reproducción.
  - Una lista de `Track` que sirve para representar las pistas del secuenciador.
  - Un diccionario[2] `Dictionary<int, Motif> motifList` donde se almacenan todos los motivos creados en el proyecto, siendo la clave

[2]Un diccionario es una representación de una colección de claves y valores.

del diccionario el identificador del motivo. No necesitamos guardar los motivos en forma de árbol, ya que cada uno de los motivos ya guarda el identificador del motivo padre, y en la mayoría de las ocasiones nos va a interesar más hacer una búsqueda sobre el identificador del motivo.

4.2.3. Importación y exportación archivos ABC

La aplicación permite importar y exportar archivos en formato ABC.

Para importar archivos `ABC` se ha utilizado `ANTLR` para crear un analizador sintáctico que es capaz de traducir estos archivos a la implementación interna de la aplicación. Por simplicidad en la implementación del analizador, los archivos `ABC` que vayan a ser importados utilizando el asistente musical deben cumplir las siguientes restricciones:

- La cabecera del archivo `ABC` debe tener únicamente los siguientes campos:
  - `X:` contiene el número de referencia, que es obligatorio en cualquier archivo `ABC`. Este campo debe ser el primer campo de la cabecera.
  - `T:` contiene el título. Este campo debe ser el segundo campo de la cabecera.
  - `M:` indica la métrica de la melodía. Este campo es opcional.
  - `L:` especifica la figura musical que se utiliza como unidad. Por ejemplo, si el valor es `L:1/4`, entonces todas las notas que aparezcan en el cuerpo del archivo que no tengan especificadas una duración serán negras. Este campo es opcional.
  - `Q:` define el tempo de la melodía. Este campo es opcional.
  - `K:` especifica la tonalidad. Este campo debe ser el último campo de la cabecera.
- Dentro del cuerpo del archivo, el analizador ignora cualquier información que no es relevante para la aplicación, como por ejemplo adornos en las notas o la letra de una partitura. Por último, tampoco permite especificar campos de la cabecera dentro del cuerpo del archivo utilizando la sintáxis entre corchetes especificada en el estándar de `ABC`.

Durante la importación de archivos en formato `ABC`, cada figura musical representada en el archivo se traduce a la representación utilizada internamente en la aplicación, que como recordamos utiliza un entero para representar la altura de la nota, y una fracción para representar la duración. La tradución de la duración es sencilla, ya que la duración también se representa en forma de fracción en los archivos `ABC`. Para traducir las notas musicales hay que recordar algunas reglas básicas a la hora de escribir música en una partitura:

- Las alteraciones de la armadura están activas durante toda la pieza musical, salvo que haya un cambio en la armadura.

- Si se añade un sostenido o bemol sobre una nota, se altera tanto esa nota como todas las notas que tengan el mismo nombre y altura hasta que acabe el compás, delimitado por una barra.

- Si se añade un becuadro sobre una nota, se elimina la alteración de esa nota, incluyendo las alteraciones de la armadura, hasta que acabe el compás.

La exportación de archivos `ABC` se realiza a través de la clase `ABCFileWriter`. La aplicación permite exportar motivos y proyectos completos.

Cuando se exporta un motivo, se crea un fichero `ABC` que contiene la tonalidad del motivo en su cabecera, y una lista de voces en el cuerpo del fichero correspondiente a la lista de voces del motivo. Por otro lado, cuando se exporta un proyecto, se crea un fichero que contiene el tempo en la cabecera, y una lista de voces que se corresponde con el número de pistas utilizadas en el secuenciador.

Cuando se exporta un motivo a un archivo `MIDI`, primero se crea un archivo `ABC` temporal, y posteriormente se llama a `abc2midi` para obtener el archivo de audio. El archivo `ABC` temporal necesita información adicional acerca del timbre y volumen de cada pista. Para ello, se utiliza la sintáxis especificada en el manual del programa `abc2midi` [2].

4.2.4. Archivos de proyecto

Como vimos en el capítulo anterior, un archivo de proyecto es un archivo que contiene toda la información acerca de una composición creada con el asistente musical. Desde el punto de vista de la implementación, un archivo de proyecto es un archivo en formato `XML`, donde se guarda la información del árbol de motivos, y el estado del secuenciador, incluyendo cada una de las pistas y la colocación de los motivos en estas.

Para la carga y el guardado de estos archivos se ha utilizado un serializador `XML`.

4.3. Interfaz gráfica

La interfaz gráfica se ha implementado con Windows Forms, que proporciona acceso a los elementos de la interfaz de Windows. En esta sección se explicará cómo se han implementado los componentes visuales principales de la aplicación.

4.3.1. Árbol de motivos

Para crear el árbol de motivos se ha utilizado el componente `TreeView`, un componente de Windows Forms que muestra una colección jerárquica de elementos con etiquetas, llamados `TreeNode`. Este componente es el más indicado para visualizar elementos en forma de árbol, y permite añadir un menú contextual (`MenuItem`) para poder programar acciones adicionales sobre cada uno de los elementos.

4.3.2. Ventana de edición de motivos

Como vimos en el capítulo anterior, la ventana de edición de motivos permite editar cualquier motivo que haya sido añadido al árbol de motivos, o crear un motivo desde cero. El panel inferior de la ventana es un cuadro de texto `RichTextBox`, que permite visualizar el contenido del motivo en formato `ABC` y editarlo.

El panel superior es un cuadro de imagen `PictureBox`, donde se muestra la partitura del motivo actualizada con los cambios realizados en el cuadro de texto `ABC`.

El proceso para actualizar la partitura es el siguiente: primero se debe copiar el contenido del cuadro de texto en un archivo `ABC`. Después, se llama a `abcm2ps` para convertir el archivo en una partitura en formato `SVG`. Una vez creada la partitura, se utiliza la librería de SVG para abrir el archivo y convertirla a un mapa de bits. Finalmente, el mapa de bits es añadido al cuadro de imagen.

Aunque el proceso completo de creación de la partitura no es demasiado largo (menos de un segundo de duración), sí es lo suficiente largo como para actualizarla cada vez que haya un cambio en el cuadro de texto. Para ello, se ha creado un hilo que se encarga de realizar este proceso, que se activa cuando detecta un cambio en el cuadro de texto y espera como mínimo medio segundo entre ejecución y ejecución. Este hilo se ha implementado mediante la clase `BackgroundWorker`, que permite ejecutar operaciones en un subproceso dedicado independiente.

4.3.3. Secuenciador y reproductor de música

El secuenciador no ha podido ser implementado a partir de ningún control existente de Windows Forms, ya que ninguno de ellos es realmente útil para representar líneas temporales. Para implementar el secuenciador se ha tenido que crear un control básico llamado `PlayList`, que herede de la clase `Control` y por lo tanto tenga las propiedades fundamentales de un control, como el tamaño, posición o la posibilidad de ser añadido a un formulario.

El control `PlayList` recibe la información del secuenciador, incluyendo las pistas que contiene, los motivos, o el estado de reproducción de la pieza

musical. Estos datos se utilizan para pintarlo utilizando la función `OnPaint`. El control también actúa como contenedor de otros controles, ya que las cabeceras de las pistas tienen un `ComboBox` para seleccionar el instrumento musical, un botón para eliminar la pista, y una barra de seguimiento para cambiar el volumen.

También se han implementado los eventos Drag and Drop para poder tanto recibir motivos desde el árbol de motivos, como para arrastrarlos dentro del secuenciador, evitando el solapamiento entre ellos.

Por último, el secuenciador permite la reproducción de la pieza musical que se está creando, mostrando una barra que indica el progreso de la reproducción. La reproducción de audio se lleva a cabo mediante un componente de Windows Media Player oculto en el formulario, que lee un archivo `MIDI` temporal creado tras pulsar el botón de reproducción en la barra de iconos del secuenciador. Este archivo `MIDI` es creado a su vez a partir de un archivo `ABC`, utilizando `abc2midi`.

Una vez iniciada la reproducción del archivo, se crea un subproceso que se encarga de actualizar la barra de progreso del secuenciador. Este subproceso también se encarga de parar la reproducción en caso de que se pulse el botón de pausa. La información de la barra de progreso se adquiere directamente del componente de Windows Media Player.

El asistente permite comenzar la reproducción en cualquier punto de la pieza musical. Para ello, simplemente se debe mover la barra de progreso hasta el punto donde se desea iniciar la reproducción. Aunque la reproducción funciona correctamente, el componente de Windows Media Player no muestra correctamente su barra de progreso cuando no se inicia la reproducción desde el principio, y por lo tanto es posible que en este caso, la barra de progreso del secuenciador se muestre ligeramente desplazada durante la reproducción.

4.4. Variaciones musicales

En el Capítulo 2 se introdujo el concepto de variación musical, que consiste en realizar ciertas modificaciones a un motivo musical, obteniendo como resultado un motivo parecido al original. A continuación se explicará como se ha implementado cada una de las variaciones que se pueden utilizar en el asistente musical.

4.4.1. La clase `Variation`

La clase `Variation` es una clase abstracta cuyo único método es `public Motif variate(Motif motif)`, es decir, este método recibirá un motivo y devolverá un nuevo motivo con la variación ya aplicada. Como una variación necesitará tener acceso al motivo para poder modificar sus elementos, por cada variación implementada se necesitará crear una clase que herede de la

clase `Variation`, y también se necesitará crear un método en la clase `Motif`, que será el método al que accederá la clase anterior y que se encargará de modificar el motivo. En la figura Figura 4.2 podemos ver la clase que ha sido creada para el transporte.

```
public class TransportVariation : Variation
{
   public TransportVariation(int transport)
   {
      this.transport = transport;
   }

   public override Motif variate(Motif motif)
   {
      Motif m = motif.Clone();
      m.transport(this.transport);
      return m;
   }

   private int transport;
}
```

Figura 4.2: Representación interna de motivos.

Recordamos que un transporte consiste en modificar la altura de las notas de un motivo por igual, por lo tanto esta variación necesitará recibir como parámetro la diferencia de altura que habrá entre el motivo original y el motivo transportado. Por otro lado, en la clase `Motif` se ha creado el método `transport`, que será el encargado de modificar la altura de las notas del motivo.

4.4.2. Implementación de las variaciones musicales

Debido a la estructura interna creada para representar motivos, algunas de las variaciones tienen una implementación trivial, como por ejemplo el transporte, que simplemente consiste en sumar o restar el mismo valor a cada nota del motivo. Por lo tanto, vamos a centrarnos en ver la implementación de algunas de ellas:

- *Retardo*: podemos interpretar un motivo retardado como un motivo que comienza con un silencio de igual duración al retardo aplicado. De esta forma, la implementación de esta variación simplemente consiste en añadir un silencio al comienzo de cada voz del motivo.

- *Inversión*: el objetivo de esta transformación es generar una simetría horizontal del motivo original.

  Para comenzar debemos definir un eje de simetría, pero para ello necesitamos conocer la tonalidad del motivo. El objetivo es que la tonalidad

del motivo no varíe tras aplicar la transformación. Para ello, se definirá el eje sobre la tónica, en caso de que la tonalidad sea mayor, y en la tercera, en caso de que la tonalidad sea menor.

Vamos a ver un ejemplo. Como podemos observar en la Figura 4.3, la tonalidad del motivo es Do Mayor, por lo tanto el eje de simetría estará sobre la nota Do. Para decidir exactamente en que nota se sitúa el eje, simplemente se realiza una media de las alturas de las notas de cada voz del motivo y se tomará como eje de simetría la tónica que esté más cerca de esa media.

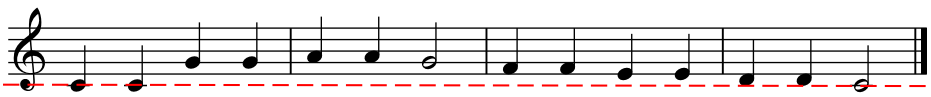


Figura 4.3: Eje de simetría sobre la nota Do.

Tras fijar el eje de simetría, se calculan los intervalos que hay entre cada una de las notas del motivo y el eje. Como vimos en el Capítulo 2, los intervalos se definen por:

- El grado, que determina la distancia de las notas del intervalo en notas naturales.
- La especie, que puede ser aumentada, mayor, justa, menor o disminuida.
- La dirección, que puede ser ascendente o descendente.

Por lo tanto, para realizar la inversión, invertiremos tanto la dirección como la especie de los intervalos, de tal forma que un intervalo aumentado pasará a ser disminuido, un intervalo mayor pasará a ser menor, y un intervalo justo seguirá siendo justo.

En el ejemplo anterior, las dos primeras notas están situadas sobre el eje de simetría y por lo tanto no cambiarán. Para la tercera nota, calculamos el intervalo entre Do y Sol, que es una quinta justa ascendente. Tras aplicar la inversión, el intervalo resultante es una quinta justa descendente, por lo tanto la nota obtenida será un Fa. En la Figura 4.4 podemos ver el motivo resultante tras aplicar la inversión junto al motivo original.

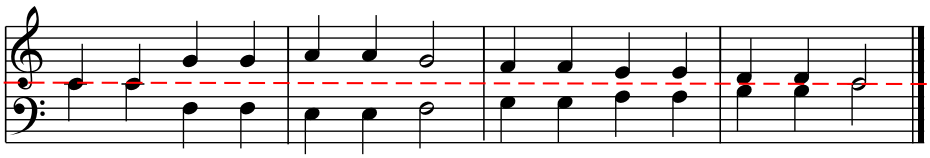
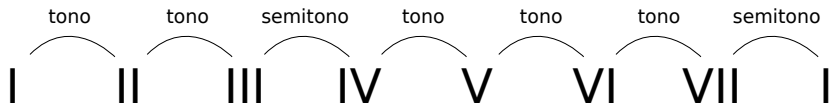


Figura 4.4: Motivo resultante tras aplicar la inversión.

- *Inversión de Rachmaninoff*: la inversión de Rachmaninoff sitúa el eje de simetría sobre el tercer grado de la tonalidad, de tal forma que la nota original y la nota invertida son equidistantes con respecto al eje. Las propiedades que se cumplen al aplicar esta transformación son las siguientes:

  - Una tonalidad mayor se transforma en una tonalidad menor, y viceversa.
  - Las notas que pertenecen a la tonalidad siguen perteneciendo a la tonalidad.
  - Las notas que pertenecen originalmente a la triada siguen perteneciendo a la triada tras la transformación.

Vamos a comprobar las propiedades anteriores. Supongamos que tenemos un motivo con una tonalidad mayor. Como vimos en el Capítulo 2, una escala mayor se construye de la siguiente forma:

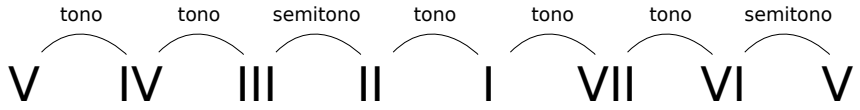


Ahora aplicamos la inversión de Rachmaninoff sobre las notas de la escala. Para que las notas de la triada de la tonalidad original sigan perteneciendo a las notas de la triada de la tonalidad resultante, necesitamos que el primer grado pase a ser el quinto grado. De esta forma, el tercer grado pasará a ser el tercer grado, y el quinto grado pasará a ser el primer grado. El resto de grados se transforman como se muestra en la Tabla 4.1.

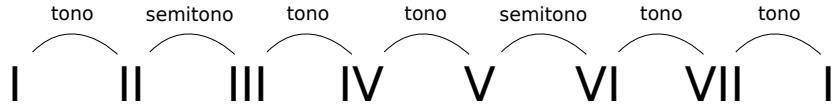
Grado en la tonalidad original	Grado en la tonalidad resultante
I	V
II	IV
III	III
IV	II
V	I
VI	VII
VII	VI

Tabla 4.1: Grados resultantes tras aplicar la inversión de Rachmaninoff

Como las notas invertidas son equidistantes a las notas originales con respecto al eje de simetría, se mantiene la distancia en semitonos entre ellas. Por lo tanto, tras aplicar la inversión, la distancia entre las notas es la siguiente:



y si ordenamos las notas en grados, obtenemos la siguiente escala:



que es una escala menor.

De forma equivalente, podemos comprobar que una escala menor se transforma en una escala mayor tras aplicar la inversión de Rachmaninoff.

En ningún momento hemos utilizado el hecho de que el eje de simetría esté situado a una distancia de tres tonos y medio con respecto a la tónica para comprobar las propiedades anteriores. Esto es porque realmente las propiedades se cumplen sin importar dónde situemos el eje. Simplemente se ha elegido esa distancia porque es la distancia utilizada originalmente en la obra *Rapsodia sobre un tema de Paganini*.

- *Transporte tonal*: el transporte tonal consiste en modificar la altura de las notas del motivo, de tal forma que las notas que pertenecen a

la tonalidad siguen perteneciendo a la tonalidad. El transporte tendrá como parámetro el grado que se va a desplazar.

En la Tabla 4.2 podemos observar como cambiarían las notas pertenecientes a la tonalidad tras aplicar un transporte tonal de tres grados en do mayor. La nota Do, que es la tónica, pasará a ser el Fa, que es el cuarto grado. La nota Re, que es la segunda, pasará a ser el Sol que es la quinta, y así sucesivamente.

Nota original	Do	Re	Mi	Fa	Sol	La	Si
Nota tras aplicar el transporte	Fa	Sol	La	Si	Do	Re	Mi

Tabla 4.2: Transporte tonal de tres grados en do mayor.

Ahora sólo necesitamos ver como transportar aquellas notas que no pertenezcan a la tonalidad. En el Capítulo 2 vimos como se forman tanto la escala mayor como la escala menor, y en ambos casos, la distancia máxima entre dos notas de la escala es de un tono[3]. Por lo tanto, si se suma un semitono a la nota, se puede asegurar que esta pertenecerá a la tonalidad. Una vez sumado el semitono, podemos aplicar el transporte tonal que hemos aplicado al resto de notas. Tras haber aplicado el transporte, restamos un semitono a la nota resultante.

- *Modulación estática*: la modulación estática transforma la tonalidad del motivo, desplazando sus notas lo mínimo posible. Esta transformación deberá cumplir los siguientes requisitos:

  - Todas las notas que pertenecen a la tonalidad original deben pertenecer a la nueva tonalidad tras la transformación.
  - Todas las notas pertenecientes a la triada de la tonalidad original (primer, tercer y quinto grado de la escala) deben pertenecer a la triada de la tonalidad resultante tras la transformación.

  Por lo tanto, la transformación consiste en asignar a cada una de las notas pertenecientes a la tonalidad original una de las notas pertenecientes a la tonalidad final.

  Primero asignamos cada nota de la triada original a una nota de la triada resultante, minimizando el desplazamiento de las notas. Tendremos tres casos posibles:

[3]La distancia máxima entre dos notas consecutivas de una escala mayor y menor es de un tono. Sin embargo, existen otras escalas donde la distancia máxima entre dos notas consecutivas puede ser mayor.

- La tónica de la triada original se asigna a la tónica de la triada resultante, la tercera a la tercera, y la quinta a la quinta. En este caso la triada resultante sería un acorde en estado fundamental.
- La tónica se asigna a la tercera, la tercera a la quinta, y la quinta a la tónica. En este caso la triada resultante sería un acorde en primera inversión.
- La tónica se asigna a la quinta, la tercera a la tónica, y la quinta a la tercera. En este caso la triada resultante sería un acorde en segunda inversión.

Una vez asignada la triada, podemos pasar a asignar el resto de notas. Como la asignación del resto de notas depende la asignación de la triada anterior, tendremos de nuevo tres casos, como se muestra en la Tabla 4.3.

	I	II	**III**	IV	**V**	VI	VII
Triada en estado fundamental	**I**	II	**III**	IV	**V**	VI	VII
Triada en primera inversión	**III**	IV	**V**	VI / VII	**I**	II	II
Triada en segunda inversión	**V**	VI / VII	**I**	II	**III**	IV	IV

Tabla 4.3: Asignación de grados tras aplicar la modulación estática.

En los dos últimos casos, para decidir cuando se asigna al VI o VII grado, simplemente asignaremos a aquel grado que minimice la distancia, y en caso de empate, al VI grado.

Podemos observar que la asignación que estamos realizando no es inyectiva, ya que notas distintas pueden ser asignadas a una misma nota. Esto puede llegar a suponer una pérdida de información con respecto al motivo original.

Vamos a ver un ejemplo: supongamos que queremos hacer una modulación estática de do mayor a fa mayor. Primero asignaríamos la triada como podemos ver en la Figura 4.5. Después, haríamos la asignación de cada nota. Como la triada resultante esta en segunda inversión, haremos la asignación siguiendo el tercer caso:

Notas de la escala original	Do	Re	Mi	Fa	Sol	La	Si
Notas asignadas	Do	Re	Fa	Sol	La	Si ♭	Si ♭

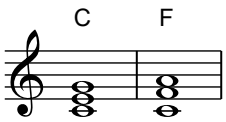


Figura 4.5: Modulación estática de Do Mayor a Fa Mayor. La triada resultante es un acorde en segunda inversión (V-I-III).

- *Canonización*: el objetivo de la canonización es aplicar un retardo al motivo original de tal forma que la disonancia entre el motivo original y el motivo resultante sea mínimo.

  Entendemos como disonancia al conjunto de sonidos que el oído percibe con tensión, y por lo tanto tiende a rechazarlos. La disonancia depende del intervalo o la distancia entre las notas que suenan de forma simultánea. Utilizando como referencia el algoritmo de consonancia-disonancia de Foster [4], se han establecido los siguientes valores de disonancia dependiendo de la distancia entre las notas:

Intervalo	Distancia (semitonos)	Valor de disonancia
2^a menor	1	240
2^a mayor	2	72
3^a menor	3	30
3^a mayor	4	20
4^a justa	5	12
4^a aumentada / 5^a disminuida	6	1440

  Por otro lado, aquellos intervalos que se puedan obtener a partir de una inversión de los anteriores, tendrán su mismo valor de disonancia. Por ejemplo, una quinta justa tendrá un valor de 12, ya que una quinta justa es una cuarta justa invertida.

  Podemos observar que los intervalos que generalmente se evitan más a la hora de componer música tienen un mayor valor de disonancia, como por ejemplo las segundas menores o las cuartas aumentadas.

  Por último necesitamos dar un valor de disonancia al unísono. El valor que podríamos darle inicialmente es 1, ya que como es lógico, la disonancia entre dos notas iguales es mínima. Sin embargo, no nos interesa que el valor del unísono sea tan bajo, ya que el objetivo de la canonización es escuchar dos motivos de forma simultánea con la mínima disonancia posible, y el unísono nos haría perder esa percepción de dos motivos sonando simultáneamente, ya que el oído no puede distinguir dos notas iguales que estén sonando a la vez. Como solución, podemos

aumentar el valor de disonancia del unísono con el objetivo de penalizarlo sobre otros intervalos. El valor que se ha decidido dar al unísono es de 25.

Una vez definidos los valores de disonancia, podemos pasar a aplicarlo sobre el algoritmo de canonización. El algoritmo funcionará de la siguiente forma:

1. Se escoge una nota del motivo. Se calcula la duración entre el comienzo de dicha nota y el comienzo del motivo.
2. Se aplica un retardo al motivo de dicha duración.
3. Se calcula la disonancia entre el motivo original y el motivo retardado. La disonancia total será la media entre la disonancia de todas las notas que suenan al unísono, dividido por unidad de tiempo.
4. Se vuelven a realizar los pasos 1, 2 y 3 con todas las notas del motivo, exceptuando aquellas notas que comiencen después de la mitad del motivo, ya que queremos que la canonización dure por lo menos la mitad del motivo.
5. El retardo que se aplica finalmente al motivo original y por lo tanto el resultado de la canonización será aquel cuyo valor de disonancia sea mínimo.

En el ejemplo de la Figura 4.6, tomaríamos las ocho primeras notas como punto de partida para los retardos. El último retardo de la figura nos da un valor de disonancia de 16,5, por lo tanto ese retardo será el elegido tras aplicar la transformación.

- *Interpolación*: la interpolación añade varias notas al motivo de forma aleatoria. La transformación utiliza dos métodos para añadir notas: reemplazar un silencio por una nota, o dividir el valor de una nota ya existente por la mitad y añadir una nota de su mismo valor.

  Cada nota añadida tendrá una probabilidad mayor de ser una nota de la triada de la tonalidad, ya que una nota de la triada tiene menos probabilidad de generar disonancias dentro del motivo.

- *Elipsis*: esta transformación elimina varias notas del motivo de manera aleatoria. Cada una de las notas eliminadas es reemplazada por un silencio.

- *Permutación de triada*: la permutación de la triada consiste en transformar al azar cada una de las notas de la triada de la tonalidad en otra de ellas. La idea de esta variación es poder modificar algunas de las notas del motivo de manera aleatoria, aunque se hace de una forma muy

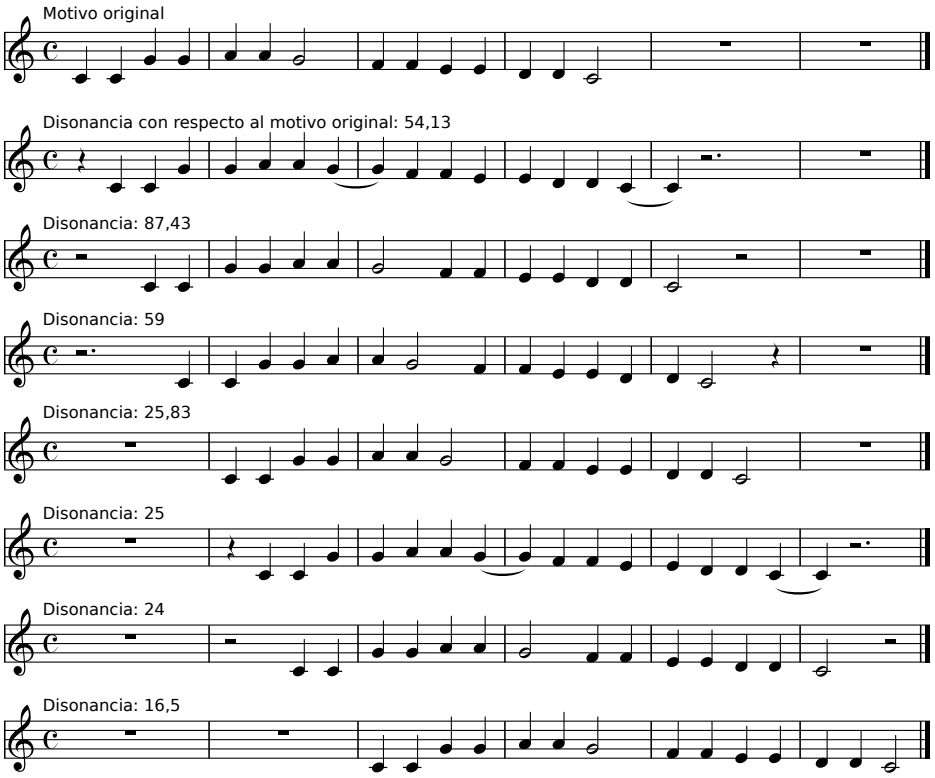


Figura 4.6: Retardos aplicados sobre el motivo durante el proceso de canonización.

restrictiva. El motivo de que únicamente se transformen las notas de la triada es evitar disonancias mayores tras aplicar la transformación.

Se han dado una serie de pesos a cada nota de la triada de forma experimental: cada nota tendrá una probabilidad de un 50 % de convertirse en la tónica, un 20 % de convertirse en el tercer grado, y un 30 % en convertirse en el quinto.

- *Armonización*: como la única información que tenemos acerca de la armonía es la tonalidad del motivo, la armonización simplemente añadirá acordes correspondientes al primer grado de la tonalidad por cada nota del motivo. Esta variación solo será conveniente utilizarla en casos excepcionales, y mejorará su funcionamiento si en un futuro se añade análisis armónico sobre los motivos.

  El objetivo de esta transformación es crear cada uno de los acordes lo más cercano posible a la nota sobre la que se crea, siempre por debajo de esta para que la nota original siga teniendo más peso.

Capítulo 5

Conclusiones

En este proyecto hemos visto qué es la música minimalista y cuáles son las principales características de esta corriente. Dentro de estas características, nos hemos centrado en analizar aquellas que son más fáciles de abordar computacionalmente, entre las que podemos destacar la repetición de motivos y el uso de variaciones musicales.

Para realizar este análisis, se ha creado un asistente para composición musical, donde un usuario puede cargar o crear varios motivos musicales, y a partir de variaciones automatizadas, crear una composición musical de carácter minimalista.

Para la creación de la interfaz gráfica se han tomado como referencia otras herramientas de composición ya conocidas, que utilizan como área principal de trabajo un secuenciador musical basado en patrones, como se ha descrito en la Sección 3.2. La utilización de este área de trabajo ha facilitado el manejo de motivos en la aplicación.

Durante la implementación de las variaciones musicales, hemos visto que algunas de ellas pueden aplicarse directamente sobre las notas del motivo, mientras que otras necesitan conocer información adicional acerca de este, como puede ser la tonalidad. Por lo tanto, hemos necesitado proporcionar esta información adicional a cada uno de los motivos para que algunas de las variaciones funcionen correctamente. Por otro lado, el asistente musical está diseñado para que en un futuro se puedan añadir más variaciones musicales con facilidad.

En los apéndices, podemos comprobar algunos de los resultados obtenidos tras la utilización de esta aplicación. En el Apéndice A se ha mostrado un ejemplo de utilización del asistente, construyendo un ejemplo de composición minimalista a partir de un único motivo musical. Por otro lado, en el Apéndice B se ha logrado replicar una composición minimalista conocida a partir de las variaciones musicales implementadas en la aplicación, utilizando de forma puntual alguna edición manual sobre los motivos.

Los resultados utilizando la aplicación han sido positivos, no sólo hemos

conseguido crear composiciones de música minimalista de forma sencilla, sino que además hemos conseguido crear composiciones que sean estéticamente satisfactorias.

5.1. Trabajo futuro

Algunas posibles mejoras que se podrían realizar sobre la aplicación son las siguientes:

- Añadir análisis armónico al asistente musical. La información que nos proporciona la tonalidad del motivo acerca de la armonía es limitada, por lo tanto realizar un análisis armónico sobre los motivos podría mejorar potencialmente el funcionamiento de algunas variaciones musicales, y permitiría la creación de otras nuevas.
- Convertir el asistente musical en una aplicación multiplataforma. Para ello, sería necesario buscar alternativas para aquellos componentes que se han utilizado en la aplicación y son nativos de Windows, como por ejemplo Windows Media Player.
- Añadir la posibilidad de propagar la edición de un motivo por las ramas del árbol de motivos. De esta forma, cuando se edita un motivo, el usuario podría actualizar todos los motivos originados a partir de variaciones de este motivo.
- Permitir la posibilidad de crear variaciones customizadas. Una posibilidad sería implementar un lenguaje pseudocódigo con el que el usuario pudiera escribir sus propias variaciones, y posteriormente utilizarlas dentro de la aplicación.

Apéndice A

Ejemplo de uso: creando una composición minimalista

A continuación vamos a ver un ejemplo de uso de la aplicación. Vamos a crear una pieza musical breve utilizando variaciones sobre el motivo de la Figura A.1, al que hemos hecho referencia en capítulos anteriores:

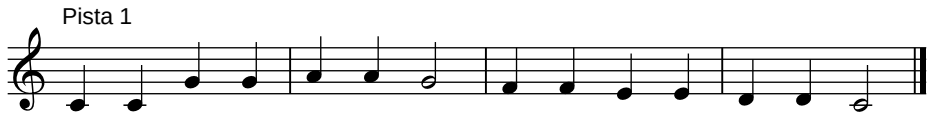


Figura A.1: Motivo inicial de la canción *Campanitas del lugar*.

Abrimos la aplicación, y dentro del menú de archivo pulsamos en nuevo motivo. Introducimos el siguiente código en el cuadro de texto:

```
X:1
L:1/4
K:C
C C G G | A A G2 | F F E E | D D C2 |]
```

Podemos verificar que tras escribir dicho código, en el panel superior debería aparecer la partitura de la Figura A.1. Tras pulsar en el botón de aceptar, aparecerá un elemento nuevo en el árbol de motivos, con el nombre de *motif*. Podemos cambiar el nombre de este haciendo click con el botón derecho y seleccionando la opción de renombrar. Vamos a renombrarlo a *campanitas*, y después vamos a arrastrar el motivo sobre la pista del secuenciador. En este momento aparecerá un cuadrado rojo en el secuenciador, como se muestra en la Figura A.2, que se corresponde con el motivo que se acaba de añadir.

Si pulsamos sobre el botón de reproducir, podremos escuchar el motivo que acabamos de añadir. Vamos a cambiar el tempo a 200 para que la canción

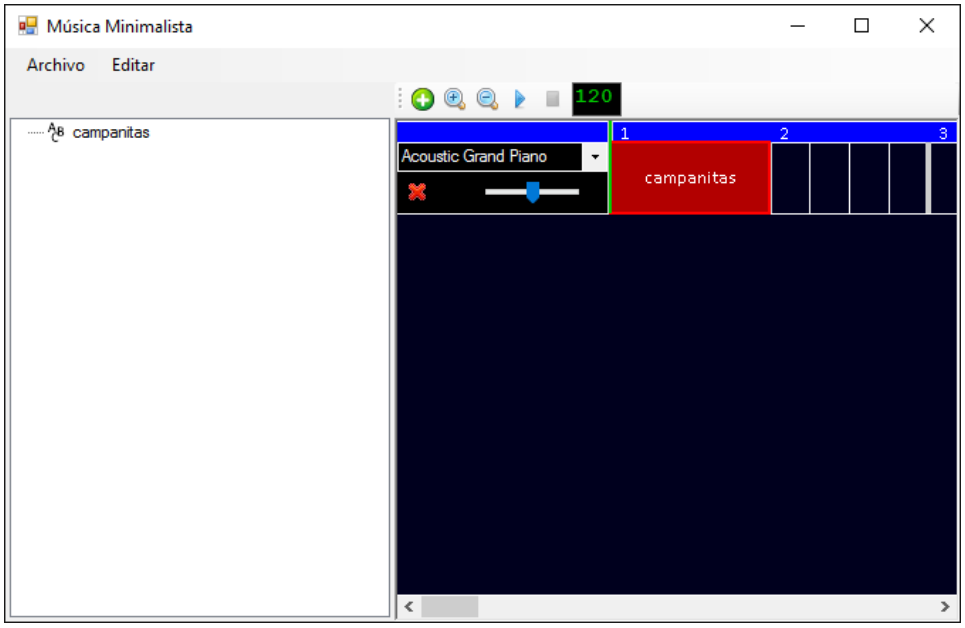


Figura A.2: Estado de la aplicación tras haber creado el motivo inicial y arrastrado al secuenciador.

sea más rápida.

Ahora vamos a crear una serie de variaciones sobre el motivo original. Para crear una variación, hacemos click derecho sobre el motivo y elegimos la opción de *Variar*. Vamos a comenzar eligiendo un transporte de cinco semitonos, una modulación estática a Sol Mayor y una retrogradación. Una vez creados, añadimos los motivos en ese orden a la pista del secuenciador. En la Figura A.3 se muestra el estado de la aplicación tras haber añadido estos tres nuevos motivos.

Podemos hacer que suenen varias melodías simultáneas. Para ello, vamos a crear una nueva pista pulsando el botón de añadir situado en la barra de iconos. Para poder diferenciar cada una de las pistas, cambiaremos el instrumento de la primera pista por una flauta, y le subiremos un poco el volumen.

Ahora, vamos a crear dos variaciones nuevas. Primero, crearemos una armonización sobre el motivo original (*campanitas*), y renombraremos el nuevo motivo como *armonia_C*. Crearemos también una armonización sobre los motivos *campanitas_transport* y *campanitas_modulation*, y renombraremos a los motivos resultantes *armonia_F* y *armonia_G* respectivamente. Arrastraremos *armonia_C* al comienzo de la segunda pista, después *armonia_F* y después armonia_G. Por último, haremos botón derecho sobre el motivo *armonia_C* situado en la pista, y lo pegaremos tras *armonia_G*. En la Figura A.4 se muestra el estado del asistente tras haber añadido estos últimos

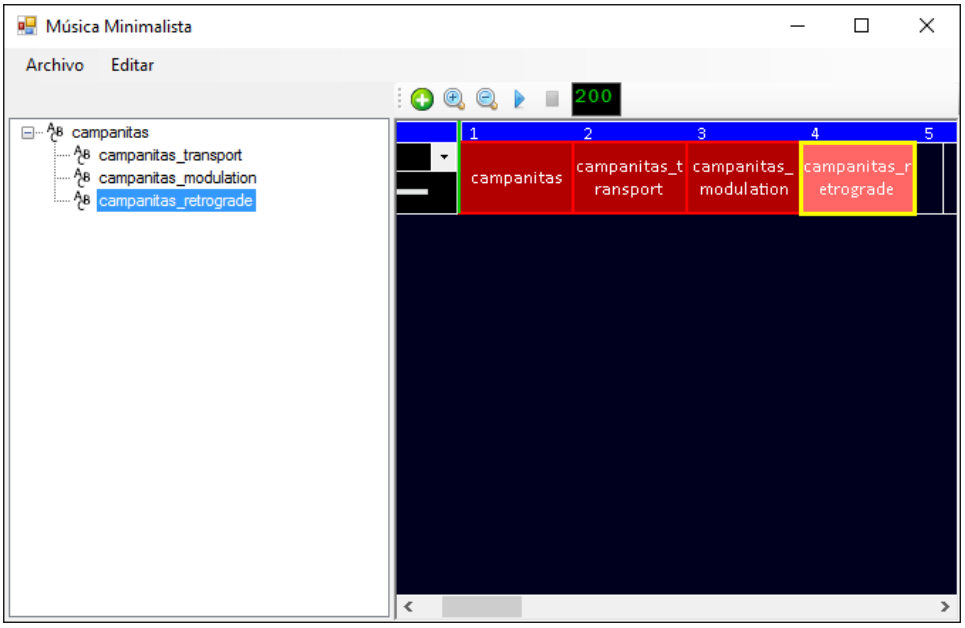


Figura A.3: Estado de la aplicación tras haber añadido las tres primeras variaciones del motivo original.

motivos.

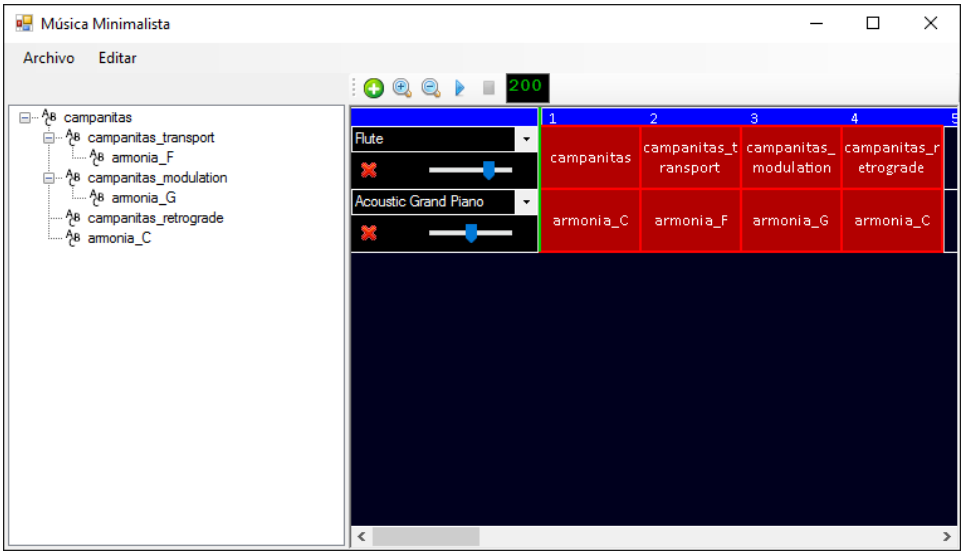


Figura A.4: Estado de la aplicación tras añadir las variaciones de armonización al secuenciador.

Para finalizar, vamos a crear una interpolación sobre el motivo *campanitas_ modulation*. Se creará el motivo *campanitas_ modulation_ interpolate*.

Recordamos que la interpolación es aleatoria, por lo tanto es recomendable comprobar si hemos obtenido una variación que nos guste. Podemos editar el motivo si es necesario desde el menú contextual del árbol de motivos, como se muestra en la Figura A.5. Finalmente, podemos eliminar el motivo *campanitas_modulation* que estaba en la primera pista, y reemplazarlo por el motivo que acabamos de crear. En la Figura A.6 podemos observar el resultado final.

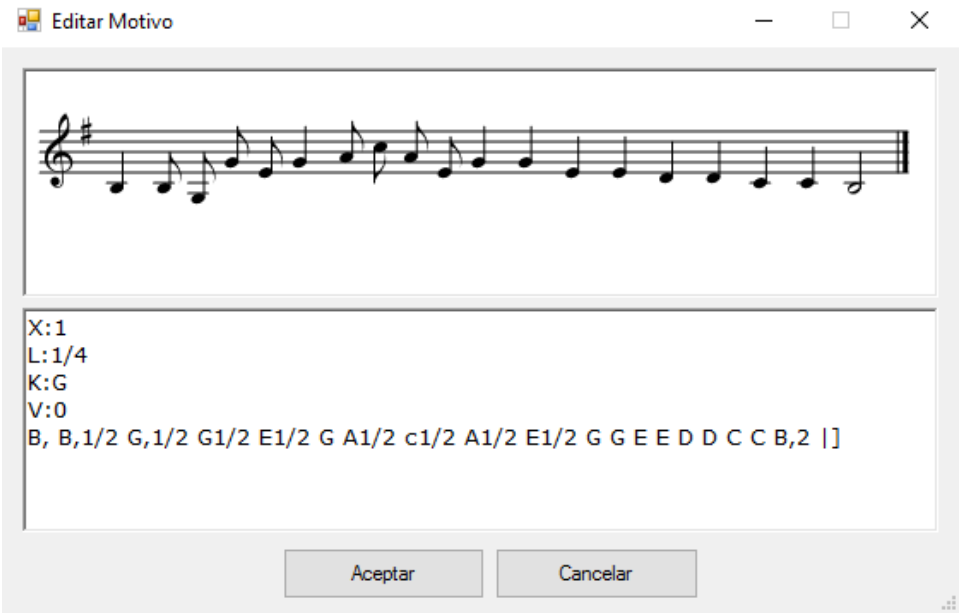


Figura A.5: Motivo *campanitas_modulation_interpolate* visto desde la ventana de edición.

Podemos escuchar el resultado final en la *Pista 15* del CD de música incluido con la memoria.

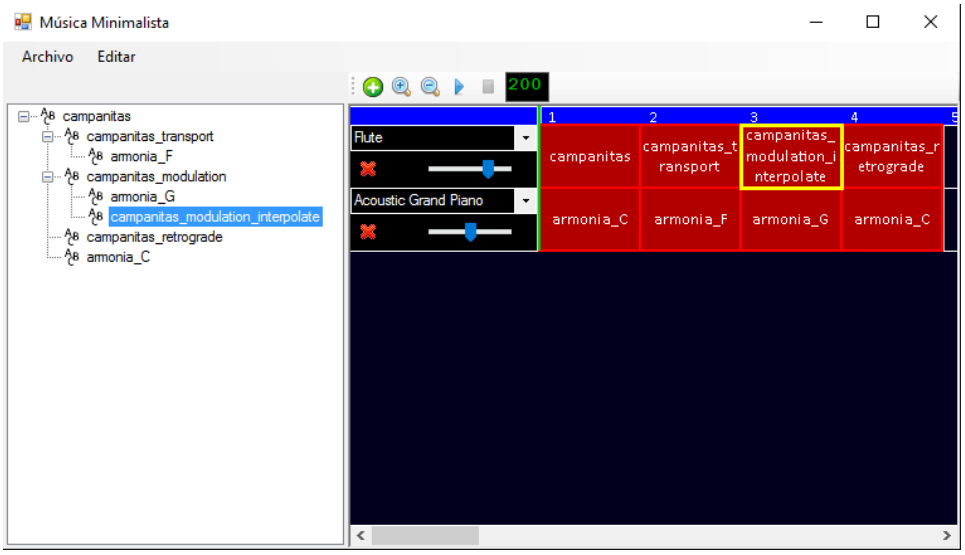


Figura A.6: Resultado final.

Apéndice B

Análisis de una composición minimalista

El objetivo de este apéndice es comprobar que es posible replicar algunas composiciones de carácter minimalista conocidas, utilizando las variaciones implementadas en el asistente, recurriendo a la edición manual de motivos solo en casos excepcionales.

Abrimos la aplicación, y en el menú de archivo elegimos la opción de *Abrir Proyecto*. Se abrirá un explorador de archivos, en donde se mostrará la carpeta *projects* del asistente. Dentro de esta carpeta abrimos el archivo llamado *amelie.mm*. En la Figura B.1 podemos ver el estado del asistente musical tras cargar el archivo de proyecto.

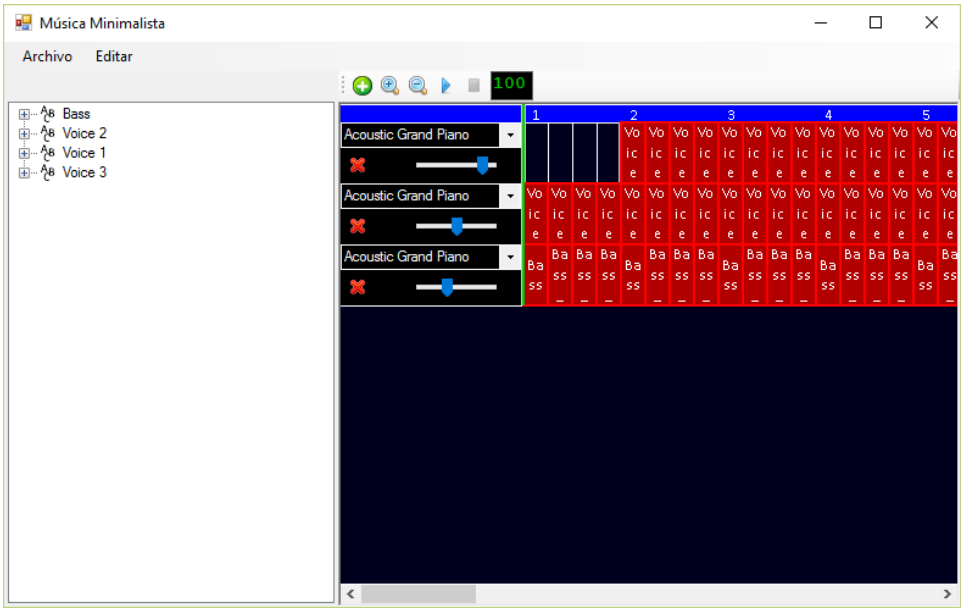


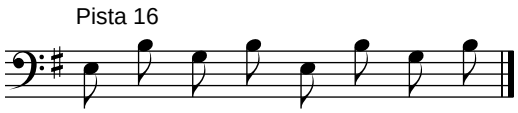
Figura B.1: Asistente musical con el archivo *amelie.mm* cargado.

Este proyecto contiene un fragmento de la obra *Comptine d'Un Autre Été*, una obra de carácter minimalista perteneciente a la banda sonora de la película *Amelie*. Podemos escuchar el contenido de este proyecto en la *Pista 20* del CD incluido con la memoria.

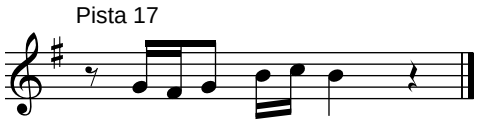
Podemos observar que se han utilizado tres pistas en el secuenciador, la primera se corresponde con la melodía principal, la segunda con la melodía secundaria, y la tercera con el bajo.

Si desplegamos el árbol de motivos, podemos ver que la composición está formada por cuatro motivos principales, sobre los que se han utilizado modulaciones estáticas y transportes, como se muestra en la Figura B.2:

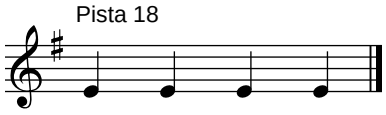
- El motivo *Bass*, que se corresponde con el motivo inicial del bajo:



- El motivo *Voice 1*, que se corresponde con el motivo inicial de una de las melodías principales:



- El motivo *Voice 2*, que se corresponde con el motivo inicial de la melodía secundaria:



- El motivo *Voice 3*, que se corresponde con el motivo inicial de otra de las melodías principales:

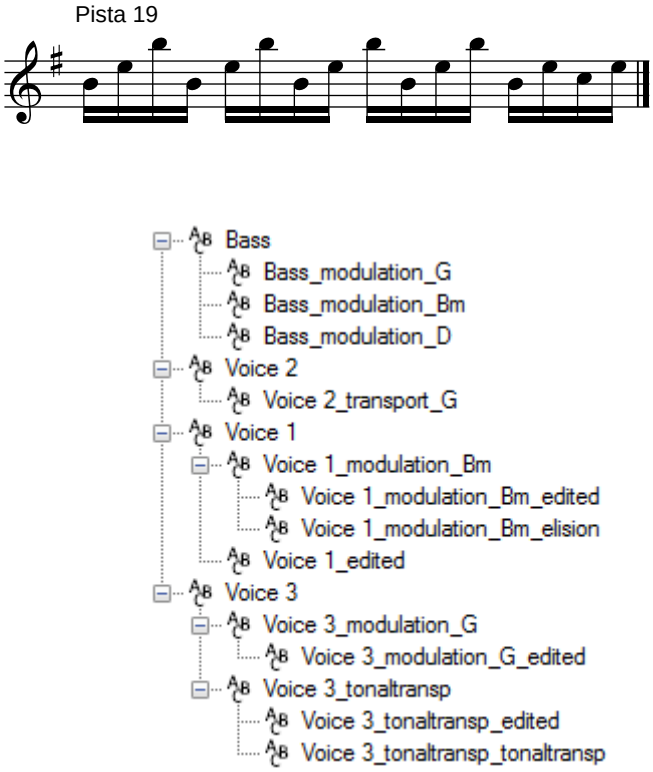


Figura B.2: Árbol de motivos del archivo *amelie.mm*.

Como podemos observar en el árbol de motivos y en el secuenciador, se ha podido construir este fragmento musical utilizando simplemente transportes y modulaciones estáticas sobre estos cuatro motivos, una elipsis y sólo en algunos casos puntuales se ha necesitado editar manualmente algunas notas del motivo.

Bibliografía

[1] The abc music standard 2.1, 2011. URL `http://abcnotation.com/wiki/abc:standard:v2.1`.

[2] J. Allwright. *Guide to writing abc for abc2midi*. URL `http://abc.sourceforge.net/standard/abc2midi.txt`.

[3] I. E. de Musicología. *Anuario musical*. El Instituto, 2007.

[4] C. D. Foster. A consonance dissonance algorithm for intervals. In *Proceedings of International Computer Music Conference*, 1995.

[5] K. Gann. Thankless attempts at a definition of minimalism. *Audio Culture: Readings in Modern Music*, 2004.

[6] P. Giblock and T. Doerffel. Lmms. URL `https://lmms.io/`.

[7] T. Halbert. Algorithmic minimalist music inspired by human intervention. 2010.

[8] B. Leach. *Looking and Listening: Conversations Between Modern Art and Music*. Rowman & Littlefield, 2015. ISBN 9780810883468.

[9] M. Nyman. Minimal music. *The Spectator*, 221(7320), 1968.

[10] T. Parr. Antlr. URL `http://www.antlr.org/`.

[11] K. Potter. *Four Musical Minimalists: La Monte Young, Terry Riley, Steve Reich, Philip Glass*. Music in the Twentieth Century. Cambridge University Press, 2002. ISBN 9780521015011.

[12] I.-L. Software. Fl studio. URL `https://www.image-line.com/flstudio/`.

[13] E. Strickland. *Minimalism–origins*. Indiana University Press, 1993. ISBN 9780253213884.

[14] Wikipedia. Composición algorítmica, .

[15] Wikipedia. Música minimalista, .

[16] Wikipedia. Música, .