UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

DESARROLLO DE UN CÓDEC IM AF E INTEGRACIÓN EN SONIC VISUALISER

INGENIERÍA DE TELECOMUNICACIÓN

JESÚS MARÍA CORRAL GARCÍA
MÁLAGA, 2014

Agradecimientos

Para comenzar, quisiera expresar mis agradecimientos a la tutora del presente Proyecto Fin de Carrera, Isabel Barbancho Pérez, por ofrecerme la oportunidad de realizar este interesante trabajo, y por su ayuda y buena disposición en todo momento. A Panos Kudumakis, profesor en *Queen Mary University of London*, por su gran dedicación y apoyo.

A nivel personal, quiero agradecer a mis amigos y compañeros que me han acompañado en distintas etapas de este viaje, Paco, Manu, Carlos, Plácido, Amando, y algunos otros. Gracias por hacer mucho más amenos los días en la escuela.

A mis amigos de la infancia, en especial, Miguel, Berni y Sergio, por estar siempre ahí.

A mis padres, hermanos y el tío Paco, por todo.

A María del Mar, por su amor, compresión y apoyo incondicional siempre.

A mis padres

con cariño

Contenido

Lista de figuras **v**

Lista de tablas **vii**

Lista de Acrónimos **ix**

1. Introducción **1**

1.1. Motivación del Proyecto Fin de Carrera 1

1.2. Objetivos del Proyecto Fin de Carrera 3

1.3. Metodología empleada 4

1.4. Estructura de la documentación 6

1.5. Formatos de audio interactivos previos a IM AF 7

1.5.1. iKlax 7

1.5.2. MT9 8

1.5.3. MOGG 9

1.5.4. Resumen comparativo de los formatos de audio interactivos tratados 10

2. El estándar IM AF **11**

2.1. Características generales 11

2.2. Modos de uso 12

2.3. Comportamiento general del Analizador de Reglas 14

2.4. Componentes soportados en IM AF 14

2.5. Grupos 16

2.6. Reglas 17
2.6.1. Reglas de selección 18
2.6.2. Reglas de mezcla 20
2.7. Presets 21
2.8. Pista de texto sincronizada con el audio 23
2.9. Metadatos 24
2.10. *Brand identification* 24
2.11. El formato MP3 26

3. Desarrollo del codificador IM AF **33**
3.1. Punto de partida 33
3.2. Especificación de requisitos de nuestro codificador IM AF 34
3.2.1. Requisitos funcionales 34
3.2.2. Requisitos técnicos 35
3.3. Implementación del codificador 36
3.3.1. Estructura del fichero IM AF 36
3.3.2. Definición del tipo de fichero 38
3.3.3. Pistas de audio 40
3.3.4. Pista de texto 45
3.3.5. Imagen JPEG 52
3.3.6. Metadatos 53
3.3.7. Grupos 55
3.3.8. *Presets* 56
3.3.9. Reglas 59

4. Integración del codificador IMAF en Sonic Visualiser **65**
4.1. Introducción a Sonic Visualiser 66
4.1.1. Herramientas de visualización 66
4.1.2. Capas de anotación 68
4.1.3. *Plugins* 69

4.1.4. Sumario de clases de Sonic Visualiser 69
4.2. La libreria Qt 70
4.2.1. Señales y *slots* en Qt 71
4.2.2. Clases de Qt 71
4.3. Integración del codificador IM AF 72
4.3.1. Introducción 72
4.3.2. Desarrollo de la interfaz gráfica e integración 73

5. Desarrollo del decodificador IM AF en Sonic Visualiser 79
5.1. Pistas de audio 80
5.2. Pista de texto 81
5.3. Interfaz Gráfica 82

6. Pruebas 87
6.1. Puntos de conformidad *(Conformance points)* 87
6.2. Creación de ficheros de verificación 89
6.3. Verificación de los ficheros 91
6.3.1. Fichero *test1.ima* 91
6.3.2. *Fichero test2.ima* 92
6.3.3. *Fichero test3.ima* 93
6.3.4. *Fichero test4.ima* 93
6.4. Verificación del decodificador IM AF en Sonic Visualiser 100
6.4.1. *Fichero test1.ima* 101
6.4.2. *Fichero test2.ima* 101
6.4.3. *Fichero test3.ima* 101
6.4.4. *Fichero test4.ima* 103

7. Conclusiones y líneas futuras 105
7.1. Conclusiones 105
7.2. Desarrollos futuros 107

A. Artículo publicado **109**

B. Capítulo publicado **117**

Referencias **143**

Índice de figuras

1.1. Interfaz gráfica del reproductor iKlax 9
1.2. Interfaz gráfica del reproductor MT9 10

2.1. Modos de uso en IM AF 13
2.2. Comportamiento general del Analizador de Reglas 15
2.3. Estructura de grupos de una canción IM AF 17
2.4. Estructura de la cabecera de una trama MP3 27

3.1. Interfaz de usuario del codificador de partida 34
3.2. Estructura del fichero IM AF 39
3.3. Creación de *Sample Table Box* 45
3.4. Cadena de texto en reproductor IM AF 46
3.5. Proceso de codificación de las letras en *Media Data Box* 49
3.6. JPEG en fichero IM AF 53

4.1. Aspecto de la interfaz de Sonic Visualiser 67
4.2. Panel con cuatro capas 68
4.3. Posibles salidas de un *plugin* VAMP en Sonic Visualiser 69
4.4. Menú que permite invocar el codificador IM AF en Sonic Visualiser 74
4.5. Interfaz del codificador IM AF 76

5.1. Proceso de decodificación de las pistas de audio 81
5.2. Proceso de decodificación de la pista de texto 83
5.3. Menú que permite la ejecución del decodificador IM AF 84

5.4. Interfaz del decodificador IM AF 85

6.1. Puntos de conformidad de los ficheros creados 89
6.2. Detalles de los ficheros creados 90
6.3. Controles de volumen de las cuatro pistas de audio en el modo de usuario (test1.ima) 92
6.4. Codificación del audio y *brand* implementada (test1.ima) 92
6.5. *Preset* donde el volumen permanece estático (test1.ima) 93
6.6. Grupo *"Ritmo"* compuesto por la base rítmica de la canción (test2.ima) . . 94
6.7. Reglas Min/Max e Inferior (test2.ima) 94
6.8. *Preset* donde el volumen permanece estático (test2.ima) 95
6.9. Controles de volumen de las cinco pistas de audio en el modo de usuario (test3.ima) 95
6.10. Letras de la canción y carátula del disco (test3.ima) 96
6.11. Metadatos (test3.ima) 96
6.12. *Preset* con volumen dinámico (test3.ima) 97
6.13. Controles de volumen de las seis pistas de audio en el modo de usuario (test4.ima) 97
6.14. Letras de la canción y carátula del disco (test4.ima) 98
6.15. *Preset* donde el volumen permanece estático (test4.ima) 98
6.16. *Preset* con volumen dinámico (efecto *fade*) (test4.ima) 99
6.17. Dos grupos definidos : Ritmo y armonía (test4.ima) 99
6.18. Regla de Implicación y Regla de Límites (test4.ima) 100
6.19. Fichero test1.ima 101
6.20. Fichero test2.ima 102
6.21. Fichero test3.ima 102
6.22. Fichero test4.ima 103

7.1. Acordes alineados con la letra 107

Índice de tablas

1.1. Tabla comparativa de los formatos de audio interactivos 10

2.1. Tabla resumen de los componentes soportados en IM AF 16
2.2. *Presets* definidos en el estándar IM AF y disponibilidad en software de referencia. 23
2.3. Información almacenada por los metadatos 25
2.4. *Brands* de IM AF 26
2.5. Valores del campo B del fichero MP3 28
2.6. Valores del campo C del fichero MP3 28
2.7. Valores del campo D del fichero MP3 28
2.8. Valores del campo E del fichero MP3 29
2.9. Valores del campo F del fichero MP3 30
2.10. Valores del campo I del fichero MP3 30
2.11. Valores del campo J del fichero MP3 31
2.12. Valores del campo K del fichero MP3 31
2.13. Valores del campo L del fichero MP3 31
2.14. Valores del campo M del fichero MP3 31

3.1. *Boxes* empleadas en el codificador 37
3.2. *Boxes* en *Sample Description Box* 42
3.3. Contenido de *Sample Size Box* 43
3.4. Contenido de *Time To Sample Box* 43
3.5. Contenido de *Sample To Chunk Box* 44

3.6. Contenido de *Chunk Offset Box* 44
3.7. Contenido de *Sample Size Box* 50
3.8. Contenido de *Time To Sample Box* 51
3.9. Contenido de *Sample To Chunk Box* 51
3.10. Contenido de *Chunk Offset Box* 52
3.11. *Boxes* usadas para los metadatos 53
3.12. *Boxes* usadas para los grupos 55
3.13. *Boxes* usadas para los *presets* 57
3.14. *Boxes* usadas para definir las reglas 59
3.15. Tipos de reglas de selección 60
3.16. Parámetros en reglas de selección 61
3.17. Tipos de reglas de mezcla 62
3.18. Parámetros en reglas de mezcla 63

Acrónimos

MP3	MPEG-1 Audio Layer III
FLAC	Free Lossless Audio Codec
AAC	Advanced Audio Coding
MOGG	Multitrack Ogg File
IM AF	Interactive Music Application Format
MPEG	Moving Picture Experts Group
ISO-BMFF	ISO Base Media File Format
3GPP	3rd Generation Partnership Project
JPEG	Joint Photographic Experts Group
WAV	Waveform Audio File Format
WMA	Windows Media Audio
SAOC	MPEG-D Spatial Audio Object Coding Baseline profile
PCM	Pulse-code modulation
MDS	MPEG-7 Multimedia Description Scheme
3GPP TT	3GPP Timed Text
URL	Uniform Resource Locator

XML	Extensible Markup Language
API	Application Programming Interface
MIDI	Musical Instrument Digital Interface

Capítulo 1. Introducción

Este primer capítulo expone la motivación del presente Proyecto Fin de Carrera. En este contexto, se presenta la relevancia de los formatos de música interactivos y la necesidad de definir un nuevo formato de audio interactivo estandarizado.

A continuación, se desarrollan los objetivos del proyecto y se presenta la metodología empleada para la consecución de dichos objetivos. En la siguiente sección, se explica la estructura de este documento, describiendo de forma somera el contenido de cada capítulo. Para finalizar el capítulo, se muestran algunos de los formatos interactivos de mayor relevancia previos a IM AF.

1.1. Motivación del Proyecto Fin de Carrera

La industria musical ha experimentado notables cambios en las últimas dos décadas, especialmente desde la popularización del estándar MPEG-1 Audio Layer III (MP3) y la llegada de Internet a una gran cantidad de usuarios. Pero dichos cambios están relacionados sobre todo con la distribución de los contenidos musicales, pero no con el modo en el que los usuarios interaccionan con la música.

A pesar del éxito de dispositivos como *smartphones* o *tablets*, en los cuales hay cierta tendencia a desarrollar aplicaciones interactivas, los usuarios de formatos como MP3, Free Lossless Audio Codec (FLAC), Advanced Audio Coding (AAC), etc solo pueden interaccionar con la música que escuchan de forma limitada. En este sentido, están surgiendo nuevos servicios interactivos musicales con el fin de mejorar la experiencia del oyente. De este modo, se pretende que el oyente deje de ser un "usuario pasivo" y se convierta en

"usuario activo", teniendo la posibilidad de crear una mezcla a su gusto, entre otras cosas [1]. En los últimos años se han comercializado distintos servicios de música interactivos como iKlax [2], MT9 [3] o Multitrack Ogg File (MOGG) [4]. Sin embargo, dichos servicios interactivos usan sus propios formatos propietarios. Para garantizar la interoperabilidad entre los distintos reproductores interactivos de música y las canciones interactivas, se requiere un formato de fichero estandarizado [5].

Para solventar dicha carencia, el grupo ISO/IEC Moving Picture Experts Group (MPEG) publicó recientemente el estándar MPEG-A: Interactive Music Application Format (IM AF) [6]. Dicho estándar define un formato de fichero diseñado para proporcionar un servicio de música interactiva. Se trata de un formato multipistas, lo que significa que puede contener pistas de audio individuales para los distintos instrumentos (guitarra, percusión, piano, voz, etc). Esto permite que el usuario pueda por ejemplo modificar la mezcla de la canción, alterando el volumen de cada instrumento por separado. Así, los oyentes tienen la posibilidad de escuchar con más detalle un instrumento concreto, aislándolo de los otros instrumentos. En el caso de músicos, puede ser de gran ayuda a la hora de realizar transcripciones, o de mejorar la técnica acompañando a sus artistas favoritos (disminuyendo total o parcialmente el volumen del instrumento deseado).

La novedad del formato Interactive Music Application Format (IM AF) frente a otros formatos multipistas como iKlax o MOGG reside en la adición de contenido multimedia para enriquecer la experiencia de usuario. De este modo, es posible la inclusión de texto sincronizado con audio, que permita la representación de las letras de la canción o los acordes que está ejecutando un determinado instrumentista. También puede contener alguna imagen relacionada con la canción, disco o artista. Con el fin de evitar que se haga irreconocible una determinada canción, se da la posibilidad al artista de definir ciertas restricciones denominadas reglas, que eviten que por ejemplo determinados instrumentos puedan ser silenciados completamente o por debajo de determinado nivel de volumen. Asimismo, el artista o productor musical dispone de la posibilidad de incluir distintos *presets* de la canción, cada uno con una mezcla distinta (por ejemplo, una versión karaoke de la canción). Otro concepto que utiliza IM AF es el de grupos. De este modo, es posible

reunir un conjunto de pistas de audio en un grupo (por ejemplo, un grupo con distintas tomas de la guitarra en una canción).

Las especificaciones del formato IM AF se aportan junto con un reproductor IM AF no optimizado y varios ficheros para verificar dicho reproductor. De este modo, Moving Picture Experts Group (MPEG) permite que se desarrollen codificadores IM AF acordes al estándar, así como diversos servicios asociados. Sin embargo, a día de hoy nadie ha hecho pública una implementación de dicho codificador.

El presente proyecto viene a cubrir dicho vacío, proponiendo una implementación de un códec (codificador-decodificador) IM AF acorde el estándar.

La integración de este códec en el software de código abierto Sonic Visualiser [7] (desarrollado para visualizar y analizar el contenido de los ficheros de audio), puede ser de interés para investigadores del área de procesado de señal, bibliotecarios musicales, o en general, para aquellos que deseen conocer con detalle el contenido de los ficheros de audio mediante una interfaz agradable.

La elección de Sonic Visualiser para realizar la integración del códec IM AF (frente a otras herramientas) es debido principalmente a su uso ampliamente extendido (más de 10000 usuarios activos en Abril de 2014) y al soporte de *plug-ins* de terceros. De este modo, es posible el empleo de *plug-ins* ya desarrollados para extracción de acordes, extracción de la melodía de la canción, o alineación entre distintas pistas de audio, entre otros.

1.2. Objetivos del Proyecto Fin de Carrera

En el contexto de los formatos de audio (como es el caso del formato IM AF), hay que hacer una distinción entre el codificador y el decodificador. El fichero de audio es creado en el codificador, de acuerdo al estándar, de este modo se puede asegurar que dicho archivo podrá ser decodificado sin ambigüedades por el decodificador, que es el encargado de leer el fichero y reproducirlo.

Con el fin de que las empresas desarrollen su propio codificador, el grupo *ISO/IEC Moving Picture Experts Group*, hace público junto con el estándar un decodificador no optimizado. Una primera versión muy limitada de un codificador IM AF se ha desarrollado en el *Centre*

for Digital Music (C4DM), Queen Mary University of London. Dicha versión permite únicamente la inclusión de pistas de audio en el archivo IM AF resultante.
Los objetivos de este Proyecto Fin de Carrera son los siguientes:

1. Desarrollo de un codificador IM AF a partir de dicha primera versión simplificada, dotando al codificador de las siguientes funcionalidades:

   a) Inclusión de la letra de la canción en perfecta sincronía con la música, resaltando las palabras en el instante de tiempo adecuado.

   b) Inclusión de imágenes que puedan ser mostradas durante la reproducción de la canción (carátula del disco, fotos del grupo, etc).

   c) Inclusión de reglas, *presets*, grupos y metadatos, de acuerdo el estándar.

2. Desarrollo de un decodificador de IM AF con las siguientes características:

   a) Decodificación de las pistas de audio.

   b) Decodificación de la pista de texto en sincronía con el audio.

3. Integración del códec IM AF en Sonic Visualiser. De este modo el usuario puede crear archivos IM AF de una forma sencilla y flexible, mediante una intuitiva interfaz gráfica. Asimismo, el usuario puede reproducir archivos IM AF en Sonic Visualiser con un fácil control de los volúmenes de las pistas de audio, al haberse desarrollado un panel con los controles de volumen de cada una de las pistas.

El desarrollo de un códec IM AF en Sonic Visualiser permite abrir nuevas posibilidades en el ámbito del procesado de señal, mediante la creación de novedosos *plug-ins*, aprovechando las características de IM AF.

1.3. Metodología empleada

Las fases realizadas en nuestro proyecto son las siguientes:

1. *Análisis del estándar ISO/IEC 23000-12 (Interactive Music Application Format).*

   Mediante dicho estudio se pretende conocer en profundidad las posibilidades del formato IM AF, así como la estructura del fichero que deberá generar el codificador.

2. *Análisis y compilación del codificador básico existente.*

   En esta fase se evalúan las funcionalidades ya implementadas con el fin de detectar las nuevas funcionalidades a desarrollar.

3. *Diseño y desarrollo de nuevas funcionalidades en el codificador IMAF.*

   Para ello se seguirá el ciclo de vida clásico del software, comprendiendo las etapas de análisis de requisitos, diseño del sistema, implementación, verificación y documentación. En este punto hay que realizar el estudio de otros estándares a los que IM AF hace referencia, ya que dicho formato toma como partida el estándar ISO Base Media File Format (ISO-BMFF) [8]. Asimismo, para implementar las letras sincronizadas con el audio hay que hacer uso del estándar 3GPP Timed Text Format [9] y el estándar Joint Photographic Experts Group (JPEG) para la imagen [10]. El conocimiento del formato MP3 es también de gran relevancia. Hay que señalar que el codificador desarrollado en esta fase únicamente permite la interacción con el usuario mediante línea de comandos, dejando el desarrollo de la interfaz gráfica para la siguiente fase. De este modo, conseguimos dividir la dificultad.

   A destacar que en la fase de pruebas será de gran utilidad el software de referencia IM AF Player, mediante el cual podemos verificar el correcto funcionamiento del fichero generado con nuestro codificador. También será util en nuestro desarrollo la aplicación MP4 Browser [11], la cual permite analizar la estructura de ficheros basados en ISO-BMFF.

4. *Integración del codificador IM AF en Sonic Visualiser.*

   El primer paso es la familiarización con la herramienta Qt [12]. Se trata de un sistema integral de desarrollo para aplicaciones multiplataforma, con el cual Sonic Visualiser se ha desarrollado. Una vez conocida la estructuración del código de Sonic

Visualiser se procede a su compilación. A continuación, se realiza la integración en Sonic Visualiser del codificador previamente desarrollado, haciendo uso de las librerías Qt.

5. *Diseño y desarrollo de un decodificador IM AF en Sonic Visualiser.*

   En esta fase se seguirán unos pasos similares a la fase 3, aunque el decodificador ya se integra directamente en Sonic Visualiser, ya que en este punto nos encontramos familiarizados con el entorno Qt. El decodificador desarrollado debe permitir la decodificación de las pistas de audio, mostrar las letras en perfecta sincronía con el audio y crear un panel con el control de volumen de cada una de las pistas.

6. *Verificación del códec IM AF desarrollado,extracción de las conclusiones finales del proyecto y posibles líneas futuras de continuación del presente proyecto.*

   Con el fin de verificar nuestra implementación del códec IM AF en Sonic Visualiser, se procede a la creación de diversos ficheros IM AF con determinadas características. Dichos ficheros deben ser perfectamente reproducidos por el IM AF Player y por el decodificador IM AF desarrollado en Sonic Visualiser.

   Las conclusiones finales del proyecto y las posibles líneas futuras serán presentadas al final.

1.4. Estructura de la documentación

La memoria del proyecto que nos ocupa, se divide en siete capítulos. Después de este capítulo introductorio, el capítulo dos hace una presentación del estándar IM AF, detallando las posibilidades que nos ofrece este novedoso formato. Asimismo, se incluye una sección dedicada al formato MP3, de gran importancia para el desarrollo de nuestro códec. El tercer capítulo se centra en el desarrollo del codificador, explicando los algoritmos implementados.
En el cuarto capítulo se expone la integración del codificador IM AF en Sonic Visualiser. Se incluyen dos apartados introductorios sobre Sonic Visualiser y la librería Qt .

El capítulo cinco se centra en el decodificador IM AF que hemos desarrollado en Sonic Visualiser.
El sexto capítulo muestra los resultados del presente proyecto, donde se expone la creación de ficheros de prueba y su posterior verificación.
El último capítulo desarrolla las conclusiones del proyecto, así como diversos comentarios sobre posibles líneas futuras que continúen el proyecto.
Como anexos, se incluyen dos publicaciones internacionales realizadas recientemente sobre la temática del proyecto y en las que participa el autor del presente proyecto. En concreto se ha realizado un artículo para el *AES 53rd International Conference on Semantic Audio*, y un capítulo del libro *Springer Handbook for Systematic Musicology*.
Para finalizar el capítulo introductorio, vamos a presentar los formatos interactivos de audio previos a IM AF de mayor relevancia, realizando una comparación con las características de IM AF.

1.5. Formatos de audio interactivos previos a IM AF

A continuación se mostrarán diversos servicios de música interactivos previos a IM AF, con el fin de comprender el contexto en el que nace IM AF.
Se describirán someramente algunos de estos formatos, estableciendo las principales diferencias y similitudes con respecto a IM AF. Hay que señalar que los formatos que se mencionarán a continuación han tenido escasa repercusión, siendo solamente conocidos en un círculo limitado de usuarios.

1.5.1. iKlax

La compañía *iKlax Media* se fundó en 2008 en Francia. Esta compañía es copropietaria del estándar IM AF, ya que la tecnología de iKlax se tomó como referencia para la creación del estándar. Las características más importantes del formato iKlax son las siguientes:

- Se trata de un formato multipista, esto es, puede contener en pistas separadas los diferentes instrumentos que componen la canción. De este modo, el usuario puede

alterar el volumen de cada uno de los instrumentos.

- Posibilidad de incluir diferentes mezclas de la misma canción dentro del mismo fichero, gracias a la inclusión de *presets*.
- La cantidad de pistas de audio que puede contener un fichero es ilimitada. Su correcta reproducción dependerá de la capacidad de proceso del computador que ejecute el reproductor.
- Los formatos de audio que soporta son los siguientes: Waveform Audio File Format (WAV), MP3, Ogg Vorbis, Windows Media Audio (WMA).
- Permite organizar diferentes pistas de audio en grupos, siendo posible crear por ejemplo, un grupo para todas las pistas de guitarra de una canción.
- Permite definir reglas, que por ejemplo evite que dos pistas estén simultáneamente activas.

Hay que resaltar que el formato iKlax no es compatible con el estándar IM AF, ya que utiliza su propio formato. La contribución de ellos al estándar IM AF fue la introducción de los conceptos de *presets*, reglas y grupos. En la Figura 1.1 podemos ver el aspecto del reproductor iKlax.

1.5.2. MT9

Se trata de un formato de audio multipista, desarrollado por *Korea's Electronics and Telecommunications Research Institute* (ETRI). El formato está siendo promovido actualmente por la compañía *Audizen* bajo el nombre *Music 2.0*.
MT9 permite un máximo de seis pistas de audio. La idea de esta limitación es que los ficheros puedan ser reproducidos en dispositivos con capacidades de procesamiento limitadas, como los teléfonos móviles. De hecho, *Samsung Electronics* y *LG Electronics* estuvieron interesados en equipar sus teléfonos móviles con un reproductor de MT9. Al final esto no fue así, probablemente debido a la aparición del estándar IM AF.

Figura 1.1: Interfaz gráfica del reproductor iKlax

El formato MT9 permite almacenar imágenes relacionadas con el artista, así como la visualización de las letras. Estas dos características fueron tomadas como referencia a la hora de definir el estándar IM AF. La Figura 1.2 muestra el reproductor MT9.

1.5.3. MOGG

El formato MOGG (*Multitrack OGG*) es esencialmente una variante multipista del formato OggVorbis, conteniendo múltiples archivos Ogg. Ogg es un contenedor que envuelve audio, video y subtitulos dentro de un mismo paquete.

Este tipo de ficheros multipista ha tenido cierta repercusión gracias al popular videojuego *Rock Band*, pudiéndose adquirir en Internet múltiples canciones de artistas de éxito en éste formato.

Existen programas gratuitos que permiten reproducir ficheros MOGG, entre ellos podríamos nombrar a *Audacity* y *Free Audio Editor 2012*.

Figura 1.2: Interfaz gráfica del reproductor MT9

1.5.4. Resumen comparativo de los formatos de audio interactivos tratados

A continuación mostraremos una tabla comparativa de los distintos formatos previos a IM AF y éste (Tabla 1.1). En dicha tabla se muestran las características más destacables de dichos formatos.

	iKlax	**MT9**	**MOGG**	**IM AF**
Multipistas	Si	Si	Si	Si
Imágenes	No	Si	No	Si
Texto	No	Si	No	Si
Presets	Si	No	No	Si
Reglas	Si	No	No	Si
Grupos	Si	No	No	Si

Tabla 1.1: Tabla comparativa de los formatos de audio interactivos

Podemos observar que el estándar IM AF aúna las características de iKlax y MT9.

Capítulo 2. El estándar IM AF

En este capítulo se introducen las características más importantes del estándar IM AF, de cara a una mejor compresión del diseño e implementación del codificador/decodificador. También se incluye un apartado dedicado al formato MP3, en el cual mostramos su estructura.

2.1. Características generales

El estándar IM AF define un formato de fichero diseñado para servicios de música interactivos. Especifica como combinar las múltiples pistas de audio con información asociada. Se trata de un formato bien definido que facilita el almacenamiento, intercambio, edición y presentación de contenidos de música interactivos.
Un archivo IM AF está compuesto de :

- Múltiples pistas de audio que representan las distintas partes (instrumentos y/o voces) de una canción;
- grupos de pistas de audio que definen una estructura jerárquica (por ejemplo todas las guitarras de una canción pueden ser reunidas en un mismo grupo);
- *presets*, que contienen información predefinida sobre la mezcla de múltiples pistas de audio (por ejemplo, una versión karaoke de la canción o una versión con la base rítmica);
- reglas que introducen datos específicos relacionados con la interacción con el usuario (definición de acciones permitidas en relación con las pistas de audio, selección de

grupos y el control del volumen);

- información multimedia adicional para enriquecer la interacción con el usuario (texto sincronizado con el audio para representar las letras de una canción e imágenes relacionadas con la canción, disco o artista);

- metadatos para describir la canción, álbum o artista;

Toda esta información se almacena en un formato de fichero ISO-BMFF.

2.2. Modos de uso

Los usuarios pueden escuchar y manipular las pistas de audio (y eventualmente otro tipo de información multimedia y metadatos) de dos modos:

- *Modo Preset*;

- *Modo Usuario*;

En el *Modo Preset*, el usuario elige un *preset* entre los que hay almacenados dentro del fichero IM AF, y se reproduce la mezcla con los parámetros del *preset* seleccionado.
En el *Modo Usuario*, el usuario puede habilitar/deshabilitar las pistas de audio y los grupos de pistas. Asimismo, permite controlar el volumen de cada pista. Las acciones llevadas a cabo por el usuario son analizadas en todo momento por el reproductor, garantizando la compatibilidad con las reglas definidas por el artista o productor. Esta compatibilidad es verificada por el analizador de reglas.
La Figura 2.1 muestra el diagrama de flujo para el *Modo Preset* y *Modo Usuario*. Observamos que la posible interacción con el usuario reside en la selección de *preset*, selección de elemento (pista/grupo) y cambios de volumen.

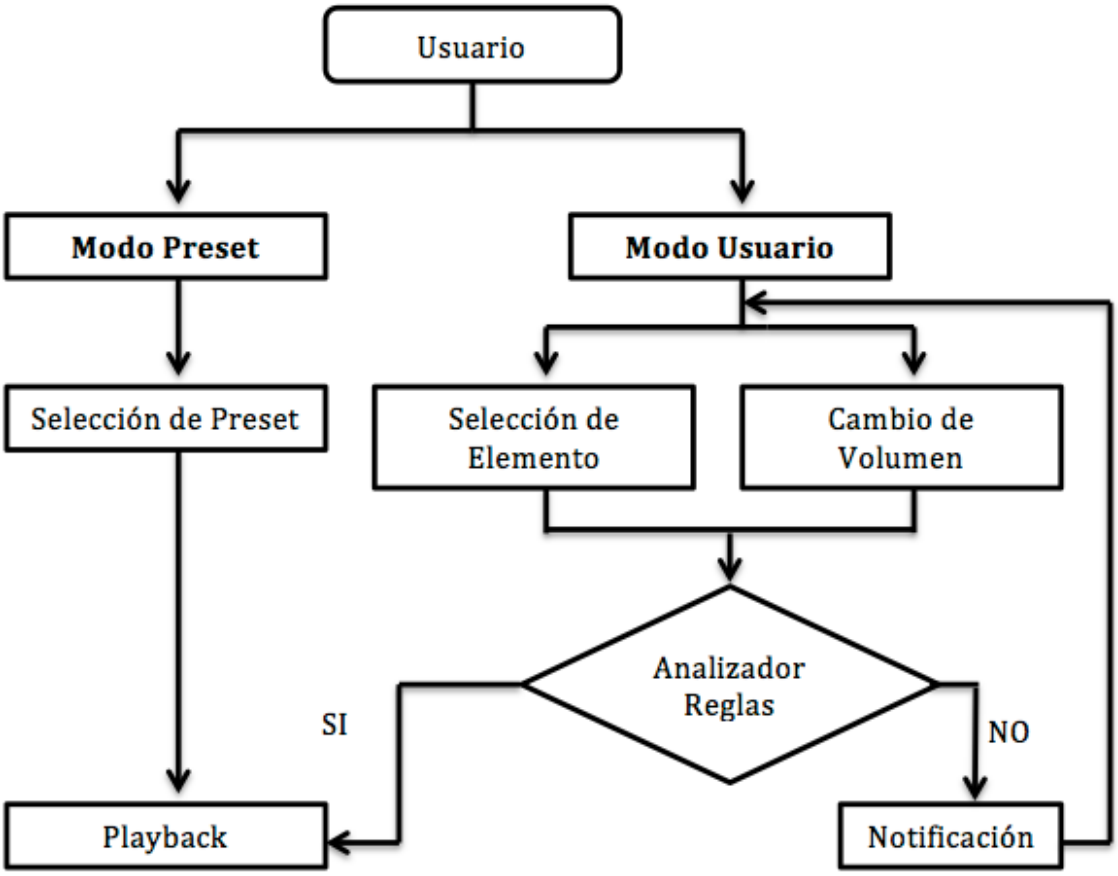

Figura 2.1: Modos de uso en IM AF

2.3. Comportamiento general del Analizador de Reglas

A continuación, se describirá el comportamiento general del Analizador de Reglas, usado en el modo de mezcla de usuario por el reproductor IM AF. Se muestra el efecto de la interacción del usuario con las reglas definidas. De hecho, se describe el modo en el que se verifica la compatibilidad entre la interacción del usuario y las reglas. El proceso que representa este comportamiento general se muestra en la Figura 2.2.

Los usuarios pueden realizar acciones sobre los elementos, como cambiar su estado (por ejemplo, habilitar o deshabitar un elemento) y cambiar su volumen. Una vez el usuario ha realizado la acción, se inicia un tiempo de espera, el cual representa el intervalo de tiempo dado al proceso para ejecutar el cambio o encontrar la solución más cercana del cambio solicitado. Se verifica la compatibilidad entre el cambio solicitado y las reglas definidas. Si se cumple la compatibilidad, entonces el mezclador ejecuta las acciones correspondientes al cambio. En otro caso, el Analizador de Reglas debería encontrar una configuración válida dentro del intervalo de tiempo definido por el tiempo de espera. El Analizador puede proporcionar diversas soluciones, donde cada una se denomina "mezcla generada". Para cada solución, se verifica la compatibilidad con las reglas para saber si la solución proporcionada es compatible. Si ninguna solución alternativa se encuentra al final del tiempo de espera, el cambio solicitado se considera imposible de ejecutar y se le notifica al usuario (mensaje de tiempo de espera). Otro caso de cambio considerado imposible puede ocurrir cuando el Analizador de Reglas no encuentra ninguna solución (mensaje de mezcla). Si se verifica la compatibilidad, se despliega la mezcla generada como una respuesta a la acción del usuario.

2.4. Componentes soportados en IM AF

El formato IM AF permite la utilización de flujos de audio de distintos tipos, como MP3, AAC, MPEG-D Spatial Audio Object Coding Baseline profile (SAOC) [13] y Pulse-code

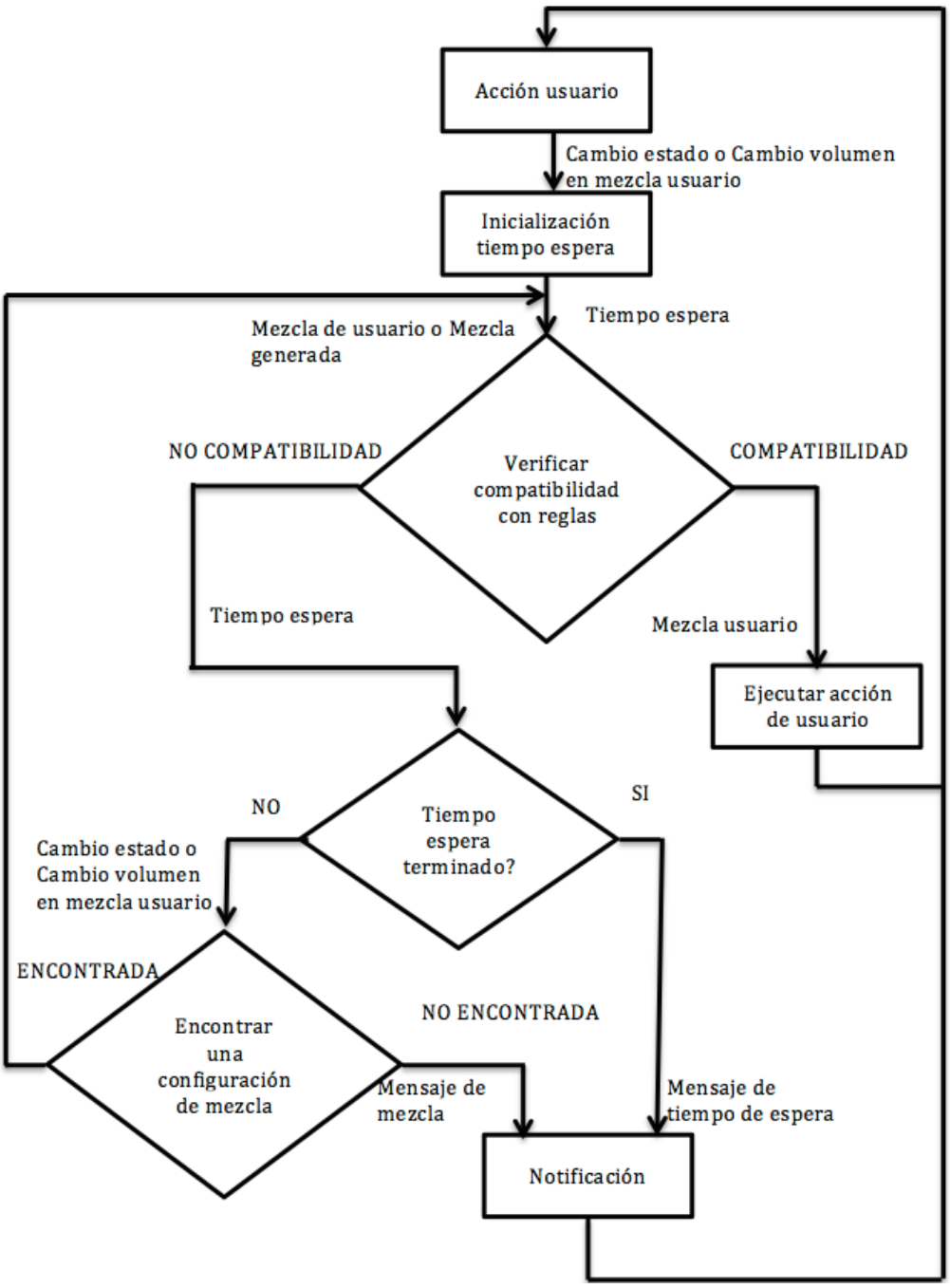

Figura 2.2: Comportamiento general del Analizador de Reglas

modulation (PCM). Para el texto utiliza 3GPP *Timed Text*, para imágenes JPEG y MPEG-7 Multimedia Description Scheme (MDS) para los metadatos [14]. La estructura del formato de fichero se basa en ISO-BMFF. La Tabla 2.1 muestra los componentes soportados en IM AF.

Tipo	**Nombre del componente**	**Abreviación**	**Especificación**
Formato fichero	ISO Base Media File Format	ISO-BMFF	ISO/IEC 14496-12:2008
Audio	MPEG-1 Audio Layer III	MP3	ISO/IEC 11172-3:1993
	MPEG-4 Audio AAC profile	AAC	ISO/IEC 14496-3:2005
	MPEG-D SAOC Baseline profile	SAOC	ISO/IEC 2003-2:200x
	PCM	PCM	-
Imagen	JPEG Image	JPEG	ISO/IEC 10918-1
Texto	3GPP Timed Text	3GPP TT	3GPP TS 26.245
Metadatos	MPEG-7 Multimedia Description Scheme	MDS	ISO/IEC 15938-5:2003

Tabla 2.1: Tabla resumen de los componentes soportados en IM AF

2.5. Grupos

La pista de audio es el elemento atómico de un archivo IM AF y puede representar un instrumento, una voz, varios intrumentos o voces mezclados juntos.

Los grupos pueden estar compuestos por un conjunto de pistas de audio y/o grupos. De este modo, los grupos permiten definir estructuras jerárquicas, donde los elementos están ordenados. El nombre, el número de grupos y el orden de los elementos los determina el productor.

Cada elemento del grupo puede tener dos posibles estados : activo o inactivo. Cuando una pista de audio se reproduce está activa, en caso contrario está inactiva.

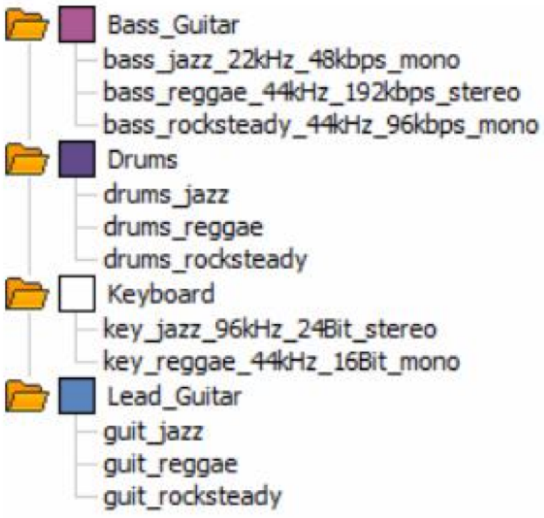

Figura 2.3: Estructura de grupos de una canción IM AF

Un grupo está en estado activo cuando al menos uno de los subelementos del primer nivel jerárquico está activo. Un grupo está inactivo cuando todos los subelementos del primer nivel jerárquico están inactivos.

Además, cada elemento del grupo tiene asignado un volumen de referencia, siendo dicho volumen el aumento de volumen que debe ser usado cuando el elemento se activa.

La Figura 2.3 muestra las pistas de audio de una canción IM AF agrupadas en cuatro grupos: Bajo (*Bass_Guitar*), Batería (*Drums*), Teclado (*Keyboard*) y Guitarra Solista (*Lead_Guitar*). Cada uno de estos grupos contiene distintas versiones de la pista de audio del instrumento correspondiente. Por ejemplo, en el grupo Bajo se encuentran tres versiones posibles del bajo de la canción, en estilo jazz, reggae y rock, respectivamente.

2.6. Reglas

El productor o artista tiene la opción de imponer ciertas restricciones al usuario para cuando se produzca la reproducción del archivo IM AF. De este modo, puede asegurarse que se preserve la creación artística. Estas restricciones se denominan reglas. El estándar IM AF define dos tipos de reglas:

- Reglas de selección;
- Reglas de mezcla;

2.6.1. Reglas de selección

Este primer tipo de reglas afectan a la selección de pistas de audio y grupos en el momento de la reproducción. A continuación se van a introducir los cuatro tipos de reglas de selección disponibles: *Regla Min/Max*, *Regla de exclusión*, *Regla de no silencio*, *Regla de implicación*.

1. *Regla Min/Max*

   Permite especificar el número máximo y mínimo de elementos del grupo que pueden estar en estado activo. Se usan dos parámetros:

   - número mínimo de elementos de el grupo que pueden estar en estado activo en el mismo momento;
   - número máximo de elementos de el grupo que pueden estar en estado activo en el mismo momento;

   Esta regla pone su atención en los estados de los elementos del primer nivel jerárquico (hijos directos del grupo).

   Se debe cumplir en todo momento la siguiente desigualdad:

   $$max(Gi) \geq \sum hijos\ directos\ del\ grupo\ Gi \geq min(Gi) \tag{2.1}$$

   Es decir, el número máximo de elementos de un determinado grupo Gi debe ser mayor o igual a la suma de los hijos directos del grupo Gi que están en estado activo. A su vez, estos deben ser mayor o igual que el mínimo número de elementos del grupo Gi.

2. *Regla de exclusión*

   Esta regla permite especificar los elementos de una canción que nunca estarán en estado activo al mismo tiempo. Se utilizarán dos parámetros, donde cada parámetro define un elemento del archivo IM AF.

   De este modo, A θ B (donde A y B son elementos y θ representa exclusión) significa que el elemento A en estado activo implica la exclusión del elemento B. Recíprocamente, cuando el elemento B está activo, el elemento A se pondrá en estado inactivo.

   La exclusión se puede aplicar a pistas de audio y a grupos, pudiendo estar aplicada sobre elementos de un mismo grupo o sobre elementos de diferentes grupos. En el contexto de los grupos, exclusión significa que los elementos de un mismo grupo nunca serán reproducidos en el mismo instante que los elementos de otro grupo. En el contexto de las pistas de audio, exclusión significa que dos pistas de audio nunca se reproducirán en el mismo momento.

3. *Regla de no silencio*

   Esta regla permite definir que un elemento esté en estado activo siempre. Se puede aplicar tanto a pistas de audio como a grupos.

   Cuando se aplica a grupos significa que al menos un elemento contenido en el grupo siempre será reproducido de acuerdo a otras reglas de selección.

4. *Regla de implicación*

   Permite establecer que la activación de un elemento implique la activación de otro elemento. Se utilizan para ello dos parámetros:

   - el primer elemento define el elemento que implica la activación;
   - el segundo define el elemento implicado;

   La implicación puede ser aplicada a pistas de audio y a grupos. En lo referente a los grupos, la implicación significa que elementos de un mismo grupo serán activados en el mismo instante o implican la activación elementos de otro grupo. En el contexto

de las pistas de audio, la implicación significa que la activación de una pista de audio implica la activación de otra pista de audio. Las pistas de audio pueden pertenecer al mismo grupo o a grupos distintos.

La implicación inversa no es implícita, si se necesita habría que definirla. A $\Rightarrow$ B significa que si el elemento A está activo, el elemento B estará también en estado activo. Si el elemento B está en el estado inactivo, entonces el elemento A estará en este estado. Por último, si el elemento A está en estado inactivo entonces el elemento B podría estar en el estado activo o inactivo.

2.6.2. Reglas de mezcla

Esta segunda categoría de reglas está relacionada con la mezcla de audio. Permite definir la forma en la que el usuario interacciona con los volúmenes de los grupos y pistas de audio en el momento de la reproducción. El volumen asociado a cada elemento podrá ser relativo o absoluto.
El volumen relativo es el valor actual de cada elemento de la estructura que puede ser modificado por el oyente en cualquier momento durante la reproducción.
El volumen absoluto es un valor real aplicado a la señal de audio de cada elemento de la canción. Este valor se calcula mediante la media de cada valor relativo dado por el oyente. Cuando el usuario cambia el volumen relativo de un elemento, el volumen absoluto es calculado y actualizado por el sistema. Si este elemento tiene hijos en la estructura, todos los volúmenes absolutos son calculados y actualizados.
Hay cuatro tipos de reglas de mezcla: *Regla de límites*, *Regla de equivalencia*, *Regla superior* y *Regla inferior*. Las reglas de mezcla son consideradas sólo para los elementos que está en estado activo en el momento de la reproducción.

1. *Regla de límites*

   Permite fijar al productor/artista los valores máximos/mínimos de los volúmenes relativos de cada elemento. De esta manera, prohibe al usuario mover los volúmenes fuera de esos límites.

2. *Regla de equivalencia*

   Esta regla se aplica a los volúmenes relativos de dos elementos de la canción. Dichos volúmenes deben respetar una relación de equivalencia estricta definida de la siguiente manera: la equivalencia entre los elementos A y B viene dada por la relación:

$$VrA/VrB = VdA/VdB \tag{2.2}$$

   donde VdA y VdB son los volúmenes de referencia de los elementos A y B.

   Esta regla puede ser aplicada a volúmenes de elementos que pertenecen al mismo grupo o a volúmenes de elementos que pertenecen a diferentes grupos.

3. *Regla superior/inferior*

   Se aplica a los volúmenes relativos de dos elementos de la canción. Estos volúmenes deben respetar una relación no estricta de superioridad/inferioridad definida respectivamente de la siguiente manera:

$$VrA/VrB \geq VdA/VdB \tag{2.3}$$

$$VrA/VrB \leq VdA/VdB \tag{2.4}$$

   donde VdA y VdB son los volúmenes de referencia de los elementos A y B.

   Esta regla se puede aplicar a los volúmenes de los elementos que pertenecen al mismo grupo o a los volúmenes de los elementos que pertenecen a diferentes grupos.

2.7. Presets

El formato IM AF permite la adición de uno o más *presets*, de forma que permita al usuario disfrutar de la música de un modo sencillo. Pueden ser de utilidad especialmente

para dispositivos portátiles (con una interfaz limitada en tamaño) como teléfonos móviles o reproductores MP3, ya que el usuario se limita a elegir uno y disfrutar de la música.

Algunos ejemplos de *presets*:

- preset general: compuesto de múltiples pistas de audio elegidas por el productor musical.
- preset karaoke: compuesto de múltiples pistas de audio excepto las vocales.
- preset a capela: compuesto por las pistas vocales.

El estándar IM AF define doce tipos de *presets* [15], pero solo cinco de ellos se encuentran implementados en el reproductor proporcionado por MPEG. De este modo, no es posible comprobar la funcionalidad de los *presets* no implementados en el software. La Tabla 2.2 muestra los *presets* definidos en el estándar y su disponibilidad en el software de referencia.

Se puede distinguir entre *presets* estáticos y *presets* dinámicos. En los *presets* estáticos, el codificador establece un volumen fijo a cada elemento que conforma la canción. En el caso de los *presets* dinámicos, el volumen de uno o varios elementos puede variar en el tiempo. Cuando en la descripción del *preset* se habla de objeto, se refiere a un canal de una determinada pista de audio. El *preset* con volumen dinámico aproximado puede ser utilizado para realizar un efecto *fade-in* o *fade-out*.

Asimismo, el estándar IM AF permite definir varios tipos de ecualización, mediante la implementación de diversos filtros en el decodificador: Filtro Paso Bajo, Filtro Paso Alto, Filtro *Shelf* de Baja Frecuencia (*Low Shelf Filter*), Filtro *Shelf* de Alta Frecuencia (*High Shelf Filter*) y Filtro *Peaking* (*Peaking Filter*).

Es posible utilizar más de un filtro sobre cada pista de audio, y el uso de la ecualización es muy sencillo: cada filtro solamente requiere una serie de parámetros (frecuencia central, ancho de banda y ganancia).

Preset		**Descripción**
0	Si	Pista con volumen estático
1	Si	Objeto con volumen estático
2	Si	Pista con volumen dinámico
3	Si	Objeto con volumen dinámico
4	Si	Pista con volumen dinámico aproximado
5	No	Objeto con volumen dinámico aproximado
6	-	Valor reservado
7	-	Valor reservado
8	No	Pista con volumen estático y EQ
9	No	Objeto con volumen estático y EQ
10	No	Pista con volumen dinámico y EQ
11	No	Objeto con volumen dinámico y EQ
12	No	Pista con volumen dinámico aproximado y EQ
13	No	Objeto con volumen dinámico aproximado y EQ

Tabla 2.2: *Presets* definidos en el estándar IM AF y disponibilidad en software de referencia.

2.8. Pista de texto sincronizada con el audio

El estándar IM AF permite acompañar la música con texto sincronizado al ritmo de la música, pudiendo representar dicho texto las letras de la canción o los acordes que están interpretando los músicos. Para ello se adopta el formato *3GPP Timed Text Format*, el cual consiste en cadenas de texto, seguidas por modificadores de texto (opcionalmente) y la información de temporización.

Las cadenas de texto son una secuencia de caracteres que representan la información textual a mostrar (por ejemplo, letras o acordes). Por otro lado, los modificadores de texto describen el modo en el cual la cadena de texto deberá ser presentada por pantalla, como por ejemplo, tamaño y el tipo de fuente o el color de fondo del texto .

La información de temporización sirve para indicar al decodificador los instantes de tiempo en los que debe mostrar las diferentes cadenas de texto.

2.9. Metadatos

Los metadatos proporcionan la información relativa al título de la canción, género, compositor, etc. Podemos encontrar los metadatos en tres niveles distintos:

- Álbum
- Canción
- Pista

La Tabla 2.3 muestra la información almacenada por los metadatos en cada uno de los tres niveles.
Los metadatos se almacenan en lenguaje XML (*Extensible Markup Language*).

2.10. *Brand identification*

Los archivos IM AF contienen un identificador formado por cuatro caracteres, que indica el tipo de codificación utilizada y restricciones del fichero. Este identificador se denomina *Brand identification.*
La Tabla 2.4 define las posibles *brands* que podemos encontrar en los archivos IM AF. En todos los casos puede existir una pista de texto 3GPP, imagen JPEG y metadatos. Los identificadores que comienzan por 'im0' están pensados para aplicaciones en dispositivos como teléfonos móviles o reproductores de audio portátiles. Los identificadores que comienzan por 'im1' se usan para aplicaciones en dispositivos con capacidades medias de procesamiento. Por último, el identificador 'im21' está pensada para uso profesional, permitiendo hasta 32 pistas de audio y frecuencia de muestreo de 96 kHz con 24 bits.

Descripción	**Nivel**		
	Álbum	Canción	Pista
Título	•	•	•
Cantante	•	•	-
Compositor	-	•	-
Letrista	-	•	-
Músico intérprete	-	-	•
Género	•	•	-
Fecha del fichero	•	•	•
Número de pista de la canción en el CD	-	•	-
Producción	•	•	-
Editor	•	•	-
Información de derechos de autor	•	•	-
ISRC (International Standard Recording Code)	-	•	-
Imagen	•	•	-
URL (dirección web relacionada con el artista)	•	•	-

Tabla 2.3: Información almacenada por los metadatos

	Audio				Max. número de pistas de audio decodificadas simultáneamente	Max. frecuencia muestreo/bits
Brands	**AAC**	**MP3**	**SAOC**	**PCM**		
'im01'	•	•			4	48kHz/16bits
'im02'	•	•			6	
'im03'	•	•			8	
'im04'	•	•	•		2	
'im11'	•	•		•	16	
'im12'	•	•	•		2	
'im21'	•			•	32	96kHz/24bits

Tabla 2.4: *Brands* de IM AF

2.11. El formato MP3

A continuación se procederá a detalladar la estructura de un fichero MP3, de cara a facilitar una mayor comprensión de los siguientes capítulos.
La estructura de un fichero MP3 es la siguiente :

[*TAG v2*] Trama1 Trama2 Trama3... TramaN [*TAG v1*]

TAG o etiqueta es la parte del fichero MP3 donde se puede almacenar información relativa a la canción, como el nombre de la canción, nombre del grupo, año de grabación, género musical, etc. Se pueden distinguir dos versiones de *TAGs*:

- *TAG v1*: ocupa 128 bytes y se encuentra al final del archivo MP3, después de la última trama de audio.
- *TAG v2*: se encuentra al principio del archivo MP3, antes de todas las tramas de audio.

Un fichero MP3 está compuesto por pequeños bloques de bits, denominados tramas. Cada trama tiene una duración constante, 0,026 segundos. Pero el número de bytes de una trama varía en función del *bitrate*. Así, para un *bitrate* de 128 kbps, una trama ocupa 417 bytes (o 418 bytes si hay bit de *padding*).

Los primeros 4 bytes de cada trama forman la cabecera de la trama, y el resto de bytes de la trama son las muestras de audio. La cabecera de la trama contiene información sobre las muestras de audio de esas trama (*bitrate*, frecuencia de muestreo, etc). Cada trama puede tener sus propias características y por tanto, variar la cabecera de una trama a otra. Esto es usado en MP3 con *bitrates* variables.

La Figura 2.4 muestra la estructura de la cabecera de trama. Cada letra se corresponde a un bit, y todos los bits con la misma letra forman parte del mismo campo. Se han asignado distintos colores a cada uno de los campos y se han agrupado los bits en grupos de 8 bits, para facilitar la comprensión.

Figura 2.4: Estructura de la cabecera de una trama MP3

A continuación vamos a proceder a comentar el significado de los distintos bits de cabecera de la trama :

- A: se corresponde con la sincronización de trama. Todos los bits de este campo toman el valor '1' y se usa para encontrar el inicio de una trama.
- B: identifica la versión de MPEG, tal y como se muestra en la Tabla 2.5.
- C: se muestran sus valores en Tabla 2.6.
- D: las tramas pueden estar protegidas con un código CRC. Este bit indica si la trama está protegida con dicho código o no (ver Tabla 2.7).
- E: indica el *bitrate* de la trama. Los valores de la Tabla 2.8 están expresados en kbps.

00	MPEG v.2.5
01	reservado
10	MPEG v.2
11	MPEG v.1

Tabla 2.5: Valores del campo B del fichero MP3

00	reservado
01	Layer III
10	Layer II
11	Layer I

Tabla 2.6: Valores del campo C del fichero MP3

0	Protegida por CRC
1	No protegida

Tabla 2.7: Valores del campo D del fichero MP3

0000	libre
0001	32
0010	40
0011	48
0100	56
0101	64
0110	80
0111	96
1000	112
1001	128
1010	160
1011	192
1100	224
1101	256
1110	320
1111	no válido

Tabla 2.8: Valores del campo E del fichero MP3

- F: indica la frecuencia de muestro. Los valores de la Tabla 2.9 están expresados en Herzios.

00	44100
01	48000
10	32000
11	reservado

Tabla 2.9: Valores del campo F del fichero MP3

- G: Bit de Padding. Sirve para acomodar el *bitrate* exactamente. Si se usan tramas de 417 bytes (para *bitrates* de 128 kbps) no dan exactamente un flujo de 128 kbps. Añadiendo el bit de padding puedes conseguir flujos de 128 kbps.
- H: Bit privado. Sirven para ser usados libremente para necesidades específicas de una aplicación.
- I: indica el número de canales de audio y su modo (Tabla 2.10).

00	Stereo
01	Joint Stereo
10	Dual
11	Mono

Tabla 2.10: Valores del campo I del fichero MP3

- J: Establece el modo en el caso *Joint Stereo* (Tabla 2.11).
- K: indica si el audio tiene copyright o no (Tabla 2.12).
- L: sus valores se muestran en Tabla 2.13.
- M: indica si frecuencias por encima de 3.2 kHz han sido enfatizadas (Tabla 2.14).

	Intensity Stereo	MS Stereo
00	off	off
01	on	off
10	off	on
11	on	on

Tabla 2.11: Valores del campo J del fichero MP3

0	Audio sin copyright
1	Audio con copyright

Tabla 2.12: Valores del campo K del fichero MP3

0	Copia de media original
1	Media original

Tabla 2.13: Valores del campo L del fichero MP3

00	No
01	50/15
10	reservado
11	CCIT J.17

Tabla 2.14: Valores del campo M del fichero MP3

Capítulo 3. Desarrollo del codificador IM AF

En este capítulo se presenta el codificador IM AF desarrollado en este proyecto. En primer lugar, se describirá el codificador del cual partimos. A continuación, se expondrán los requisitos funcionales y técnicos de nuestro codificador.
El apartado mayor relevancia en este capítulo será en el que describamos la implementación del codificador, abordando en distintas secciones cada una de las funciones desarrolladas.

3.1. Punto de partida

Para el desarrollo de nuestro codificador partimos de una primera versión muy simple de un codificador IM AF. Esta primera versión fue desarrollada en el *Centre for Digital Music, Queen Mary University of London*. Dicha versión permite la generación de un fichero con formato IM AF que incluya distintas pistas de audio codificadas en formato MP3. De este modo, al reproducir el archivo, el usuario puede alterar el volumen de cada una de las pistas de audio que conforman la canción. El resto de las características que incluye el estándar IM AF no fueron desarrolladas y se dejaron para futuros desarrollos. Este proyecto viene a cubrir esa necesidad, añadiendo nuevas funcionalidades al codificador e implementándolas de acuerdo al estándar.
Características generales del codificador de partida :

- El codificador está implementado en lenguaje C.

- Permite la inclusión de pistas de audio únicamente.
- Las pistas de audio deben estar codificadas en formato MP3 a frecuencia de muestreo de 44.1 kHz.
- La interfaz de usuario es una simple línea de comandos a través de la cual se introduce el número de pistas de audio y sus respectivos nombres.

La Figura 3.1 muestra una captura de pantalla de la interfaz de usuario del codificador del que partimos. Como vemos es una simple línea de comandos en la cual se nos pregunta únicamente el número de pistas de audio que contendrá el archivo IM AF y los nombres de dichas pistas de audio. El directorio en el cual deben estar situados los archivos MP3 viene establecido por defecto en el código.

```
How many tracks there will be in your IMAF file?
3
Name of the track number: 1
bass.mp3
Name of the track number: 2
drums.mp3
Name of the track number: 3
synth.mp3
```

Figura 3.1: Interfaz de usuario del codificador de partida

3.2. Especificación de requisitos de nuestro codificador IM AF

Tras la descripción de las características generales del codificador del que partimos, se presentarán los requisitos funcionales y técnicos del presente proyecto.

3.2.1. Requisitos funcionales

Los requerimientos funcionales del codificador son los siguientes:

- Creación de archivo IM AF (extensión .ima) de acuerdo al estándar.

- Inclusión de archivos de audio en formato MP3, permitiendo un máximo de seis pistas.

- Inclusión de texto sincronizado con el audio, que puede representar la letra de la canción, u otra información como pueden ser los acordes de dicha canción. Para ello se emplea el estándar 3GPP TT.

- Inclusión de imágenes que pueden representar la carátula del disco en cuestión o imágenes relacionadas con el artista/álbum. Se emplea el estándar JPEG.

- Creación de reglas, tanto reglas de selección como reglas de mezcla. Dichas reglas definen las acciones permitidas al usuario en relación con la activación/desactivación de pistas de audio, volumen de dichas pistas y selección de grupos.

- Creación de grupos, para poder agrupar varias pistas de audio. De este modo el usuario puede actuar sobre varias pistas a la vez.

- Creación de *presets*, pudiendo incluir *preset* con volumen estático o dinámico. En el caso de volumen dinámico, se debe poder crear un archivo IM AF con un efecto *fade-in* o *fade-out*.

- Incorporación de metadatos, según el estándar MDS. Dichos metadatos deben contener información sobre la canción, pista y album.

3.2.2. Requisitos técnicos

El codificador a desarrollar se implementa en lenguaje C. Microsoft Visual Studio 2012 es el entorno de desarrollo integrado elegido para tal fin. El computador utilizado en la realización del presente proyecto posee un procesador Intel Dual Core a 2 GHz y 2 GB de memoria RAM. El sistema operativo empleado es Microsoft Windows 7.

3.3. Implementación del codificador

En esta sección desarrollamos la implementación del codificador IM AF, dedicando un primer subapartado a la estructura que tendrá el fichero generado por el codificador. El resto de subapartados detallan cada una de las funcionalidades que presenta el codificador.

3.3.1. Estructura del fichero IM AF

El codificador es el encargado de crear el archivo IM AF, en base al estándar. La estructura de formato de fichero IM AF se deriva del estándar ISO-BMFF.

Los ficheros IM AF están formados por una serie de objetos denominados *boxes*, donde están contenidos todos los datos del fichero. Se pueden diferenciar dos tipos de *boxes*, por un lado las que contienen otras *boxes* dentro (llamadas *Boxes*), y por otro lado las *boxes* que solo contienen datos (llamadas *FullBoxes*). Ambos tipos de *boxes* empiezan por una cabecera, la cual proporciona su tamaño y tipo.

El tamaño viene expresado por el número de bytes total de dicha *box*, utilizándose 32 bits para representarlo. Si el tamaño fuera mayor a 32 bits se podrían utilizar 64 bits. El tipo también se expresa con 32 bits y sirve para identificar la *box*. Ejemplo de tipos de *boxes*: *File Type Box, Media Data Box, Movie Box,* etc.

Las *FullBoxes* además cuentan con dos campos extra en la cabecera: versión y *flag*. El campo versión es un entero formado por 8 bits y especifica la versión de la *box*. Por otro lado, el campo *flag*, es un entero formado por 24 bits y su uso depende la *box* en concreto.

El orden de almacenamiento de los bytes sigue el formato *big-endian*, es decir, primero se almacena el byte más significativo.

La Tabla 3.1 muestra la estructura de *boxes* empleada en nuestro codificador IM AF.

La columna situada a la izquierda muestra las *boxes* de más alto nivel. El sangrado refleja las *boxes* que se localizan dentro de otras *boxes*. Asi, por ejemplo, *mvhd* está contenida dentro de *moov*. La última columna contiene el nombre la *box* en cuestión.

Con el fin de mejorar la interoperabilidad, hay una serie de recomendaciones en cuanto al orden de aparición de las *boxes*:

ftyp						File Type Box
mdat						Media Data Box
moov						Movie Box
	mvhd					Movie Header Box
	trak					Track Box
		tkhd				Track Header Box
		mdia				Media Box
			mdhd			Media Header Box
			hdlr			Handler Reference Box
			minf			Media Information Box
				smhd		Sound Media Header Box
				dinf		Data Information Box
					dref	Data Reference Box
				stbl		Sample Table Box
					stsd	Sample Description box
					stts	Time to Sample Box
					stsc	Sample to Chunk Box
					stsz	Sample Size Box
					stco	Chunk Offset Box
	grco					Group Container Box
		grup				Group Box
	prco					Preset Container Box
		prst				Preset Box
	ruco					Rule Container Box
		rusc				Selection Rule Box
		rumx				Mixing Rule Box
meta						Metadata Box

Tabla 3.1: *Boxes* empleadas en el codificador

- *File Type Box* debe aparecer al principio del fichero, ya que define el tipo de fichero.
- Solo puede aparecer en el fichero una única *Movie Box.*
- Las *boxes* contenidas dentro *Sample Table Box* deben seguir el siguiente orden : *Sample Description, Time to Sample, Sample to Chunk, Sample Size, Chunk Offset.*

A continuación, describiremos el contenido de las principales *boxes* que conforman la estructura de un fichero IM AF.
File Type Box (ftyp) define el tipo de fichero, tal y como hemos comentado anteriormente. Entre otras cosas indica el formato de las pistas de audio y el número máximo de pistas de audio.
El contenido multimedia (audio, texto, imágenes) se almacena dentro de *Media Data Box (mdat).*
La información necesaria para la decodificación de dicho contenido multimedia se encuentra en *Movie Box (moov)*, la cual puede contener una o varias *Track Box (trak).* Cada *Track Box* contiene la descripción de una sola pista de audio o texto. También es posible que *Track Box* incluya una Uniform Resource Locator (URL), que indica la localización en la que se encuentra almacenado el contenido multimedia (en lugar de almacenarlo en Media Data Box). De este modo podemos conseguir que el archivo IM AF ocupe menos espacio.
En *Group Container Box (grco)*, *Preset Container Box (prco)* y *Rule Container Box (ruco)* está contenida la información relativa a los grupos, *presets* y reglas, respectivamente.
Metadata Box (meta) incluye los metadatos, como por ejemplo el título de la canción, álbum, año de publicación del álbum, etc.
La Figura 3.2 representa de forma visual la estructura del fichero IM AF, con las distintas *Boxes (ftyp, mdat, moov, meta)* y *FullBoxes (grup, prst, rusc, rumx, etc).*

3.3.2. Definición del tipo de fichero

La primera *box* que debe aparecer al comienzo del fichero define las características de las pistas de audio contenidas en el fichero IM AF, y es donde se define la *brand* del fichero.

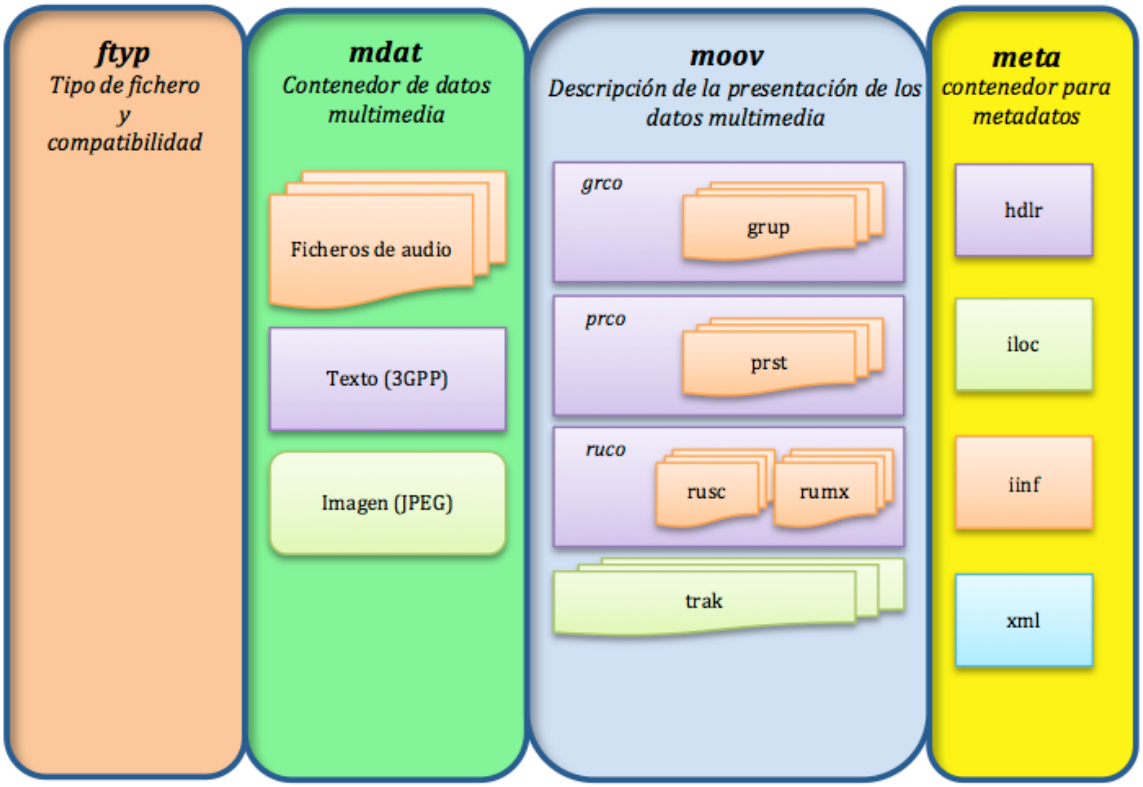

Figura 3.2: Estructura del fichero IM AF

Se trata de *File Type Box (ftyp)*, y su tamaño total es de 24 bits. Se ha implementado la *brand 'im02'*, la cual permite hasta 6 pistas de audio simultáneas. De este modo el fichero IM AF puede ser reproducido fácilmente en cualquier tipo de dispositivo, sin requerir grandes capacidades de procesamiento.

3.3.3. Pistas de audio

Una vez definido el tipo de fichero IM AF, el codificador extrae las muestras de audio de los archivos MP3 y las almacena en *Media Data Box (mdat)*. Esta *box* tendrá un campo *tamaño* que indica su tamaño total incluyendo todas las pistas de audio, la pista de texto y la imagen (si es que se incluyen en el archivo) y un campo *tipo* que almacenará *'mdat'*. A continuación todas las muestras se almacenarán byte a byte. Hay que señalar que las pistas de audio se almacenan unas detrás de otras, y el decodificador puede extraer las diferentes pistas gracias a la información almacenada en cada *Sample Table Box* (como se explicará posteriormente). A continuación, vamos a describir el proceso de creación de las *boxes* involucradas en el audio.

Media Data Box

Los pasos realizados por el codificador son:

- El codificador abre el primer fichero de audio MP3.
- Busca la cabecera de la primera trama del MP3, saltándose la etiquetas ID3 que puedan aparecer al comienzo del archivo. La cabecera de una trama MP3 está compuesta por 32 bits. El algoritmo contempla distintas cabeceras posibles. De este modo, habrá encontrado la cabecera cuando los valores de los campos de la cabecera sean los siguientes (consultar apartado 2.11 para mayor información sobre la estructura de un fichero MP3):
  - A : los once bits de alineación de trama deben estar a '1'.
  - B : los dos bits a '01' (*Layer 3*).
  - C : los dos bits a '01' (*MPEG 1*).

- D : se contempla los dos posibles casos (trama protegida/no protegida).
- E : los 4 bits a '1001' (128 kbps).
- F : dos bits a '00' (44100 Hz).
- G : se contemplan los dos posibles casos (bit *padding* o no).
- H : el bit a '0' (bit privado).
- I : dos bits a '01' (*Joint Stereo*).
- J : se contemplan las cuatro opciones posibles.
- K : el bit a '0' (sin *copyright*).
- L : se contemplan los dos casos (copia de media o media original).
- M : los dos bits a '00' (no énfasis).

- Una vez ha localizado la primera trama, almacena byte a byte todas las muestras del fichero MP3 en el archivo IM AF, concretamente en *Media Data Box*.
- A continuación hace el mismo proceso de búsqueda de cabecera válida y escritura de las muestras de audio para cada una de las pistas MP3, almacenándolas unas detrás de otras en *Media Data Box (mdat)*.
- Cálculo del tamaño de *Media Data Box (mdat)*. Dicho valor incluye el tamaño que ocupan las pistas de audio, la pista de texto (si la hubiere) y la imagen (si la hubiere).

Sample Table Box

A continuación se detallará el contenido almacenado en *Sample Table Box*, donde encontramos: *Sample Description Box*, *Sample Size Box*, *Time To Sample Box*, *Sample to Chunk Box* y *Chunk Offset Box*. Un archivo IM AF contendrá una *Sample Table Box* por cada pista de audio.

Sample Description Box (stsd)

Contiene información del tipo de codificación usada, *bitrate* e información especifica para la decodificación. La Tabla 3.2 muestra la estructura de *Sample Description Box*.

...				
stsd				
	mp4a			
		esds		
			ES_Descriptor	
				Decoder Config Descriptor

Tabla 3.2: *Boxes* en *Sample Description Box*

Audio Sample Entry Box (mp4a) almacena las especificaciones técnicas de una pista de audio (número de canales, frecuencia de muestreo, etc). En nuestro codificador, la frecuencia de muestreo será de 44100 Hz y el número de canales será igual a dos.
En *Elementary Stream Descriptor Box (esds)* se almacena la información relacionada con un flujo. De este modo, *Decoder Configuration Descriptor* proporciona información sobre los recursos necesarios para el flujo asociado. Por ejemplo, se almacena el *bitrate* medio y máximo del flujo asociado. El valor almacenado será de 128 kbps en ambos casos.

Sample Size Box (stsz)

Almacena el número de tramas de la pista y una tabla con el tamaño en bytes de cada trama. En el caso de ficheros MP3 de *bitrate* constante a 128 kbps, las tramas pueden ocupar 417 o 418 bytes. El valor será uno u otro dependiendo de si existe bit de *padding* o no en la trama.
El algoritmo busca todas las posibles cabeceras. Una vez encontrada una cabecera conocida, va contando el número de bytes hasta encontrar otra cabecera conocida. De este modo calcula el número de tramas y el tamaño de cada una de ellas.
La Tabla 3.3 muestra un ejemplo real del contenido que almacena *Sample Size Box*. Observamos en el ejemplo, que *Sample Size Box* está compuesta por 8953 tramas, ocupando cada una de ellas 417 o 418 bytes.

num_tramas	8953
tam_trama[0]	417
tam_trama[1]	417
tam_trama[2]	418
....	....
tam_trama[8952]	418

Tabla 3.3: Contenido de *Sample Size Box*

Time To Sample Box (stts)

Almacena una tabla, donde en cada entrada se almacenan dos valores: el número de tramas contiguas que tienen la misma duración de tiempo (*count*) y la duración de dichas tramas en milisegundos (*delta*). Asimismo, también incluye un entero que define el número de entradas de la tabla.

La duración de una trama MP3 es de 26,125 milisegundos. Esto provoca que haya que añadir una trama de 27 milisegundos cada 8 tramas. Como 8 tramas x 26,125 ms/trama = 209 ms, lo que almacenamos es 7 tramas x 26 ms/trama + 1 trama x 27 ms/trama = 209 ms. Es decir, escribe 7 tramas de 26 milisegundos, seguida de una de 27 milisegundos.

La Tabla 3.4 muestra un ejemplo del contenido de *Time To Sample Box*. Observamos que se almacenan 2194 entradas en la tabla, donde en la primera entrada se indica que las siete primeras tramas duran 26 milisegundos.

num_entradas	2194
(count,delta)[0]	(7,26)
(count,delta)[1]	(1,27)
....	....
(count,delta)[2193]	(2,26)

Tabla 3.4: Contenido de *Time To Sample Box*

Sample To Chunk Box (stsc)

La tramas se agrupan en estructuras denominadas *chunks*. *Sample to Chunk Box (stsc)* define una tabla que incluye el número de tramas que componen cada *chunk* y el número de *chunks* consecutivos con el mismo número de tramas. En el caso de archivos MP3 de *bitrate* constante, esta tabla contiene una entrada únicamente.

La Tabla 3.5 muestra un ejemplo de la información almacenada en *Sample To Chunk Box*. Esta tabla tiene una entrada con tres variables: la primera especifica el índice del primer *chunk* (dicho índice tiene el valor '1'); la segunda variable es el número de tramas de cada *chunk* ('8953' en este caso), y la tercera es el número de *chunks* consecutivos con el mismo número de tramas (como hay solo una entrada, se asigna el valor '1').

num_entradas	1
sample_to_chunk [0]	(1,8953,1)

Tabla 3.5: Contenido de *Sample To Chunk Box*

Chunk Offset Box (stco)

Indica la posición que ocupa la primera trama de la pista de audio almacenada en *Media Data Container Box (mdat)* del fichero IM AF. La Tabla 3.6 muestra un ejemplo del contenido de *Chunk Offset Box*. Se observa que contiene una tabla con una entrada, que indica que la primera trama de la pista de audio está situada en el byte '18709967' del archivo IM AF.

num_entradas	1
chunk_offset [0]	18709967

Tabla 3.6: Contenido de *Chunk Offset Box*

Para finalizar el presente apartado, mostraremos una figura que ilustra el proceso de creación de *Sample Table Box* con sus correspondientes *boxes*, tal y como se ha desarrollado a lo largo del apartado. Lo podemos ver en la Figura 3.3.

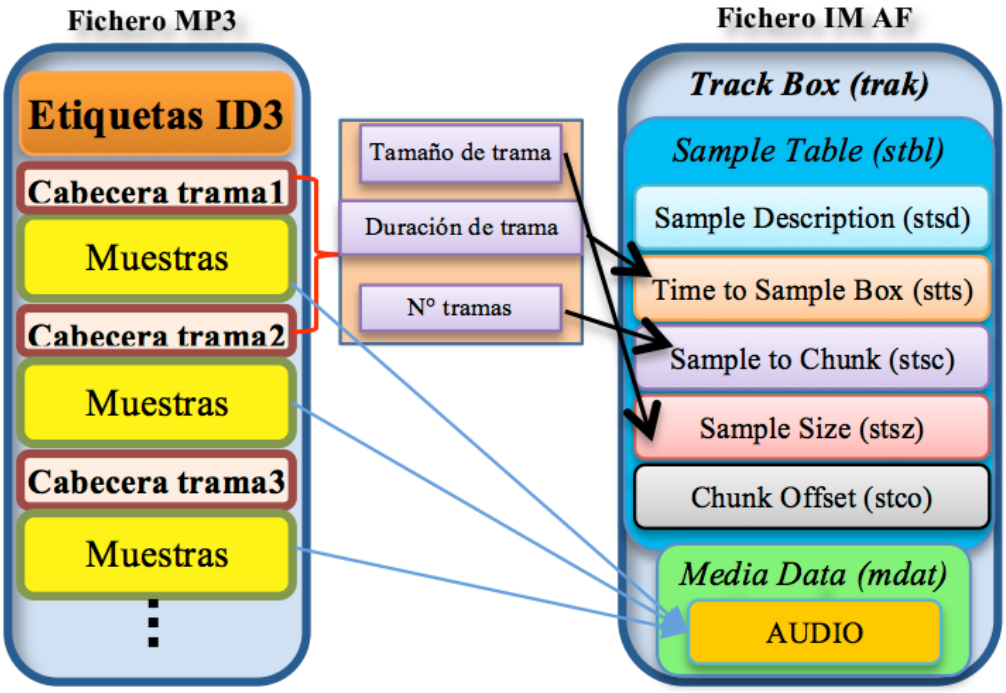

Figura 3.3: Creación de *Sample Table Box*

3.3.4. Pista de texto

Como se comentó anteriormente, IM AF permite incluir texto sincronizado con el audio, pudiendo representar dicho texto las letras de la canción del mismo modo que un karaoke. El formato empleado en la pista de texto viene definido en el estándar 3GPP TT.

Antes de continuar, hay que hacer una aclaración en cuanto a la nomenclatura empleada en este apartado. El texto a mostrar está compuesto por cadenas de texto, y cada cadena de texto está compuesta a su vez por palabras (separadas por espacios en blanco). Cuando hablamos de una cadena de texto, nos referimos al conjunto de caracteres que se muestran en pantalla en un instante determinado. La Figura 3.4 muestra un recuadro rojo con una cadena de texto, y en verde se hayan recuadradas las palabras. Dicha cadena de texto se muestra al minuto y dos segundos de comenzar la reproducción y permanecerá por pantalla varios segundos. En otros instantes de tiempo se mostrarán otras cadenas de texto.

El contenido que almacenará el fichero IM AF en relación con la pista de texto es el

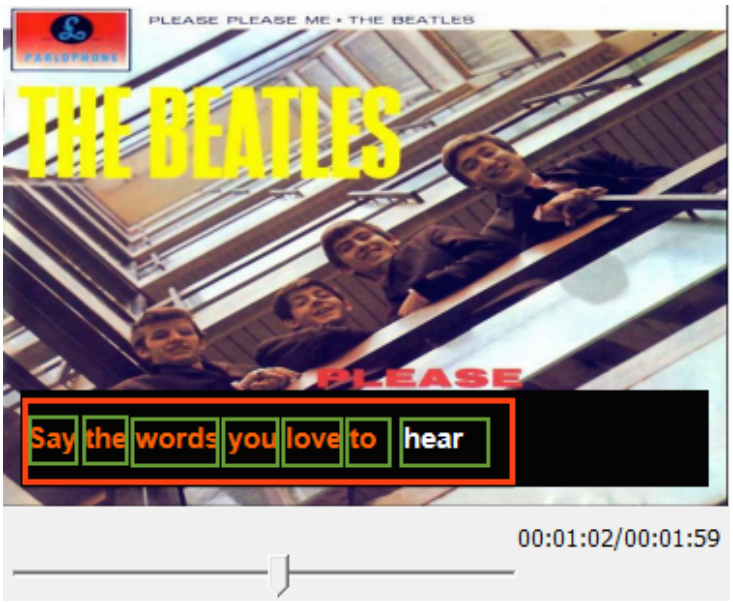

Figura 3.4: Cadena de texto en reproductor IM AF

siguiente:

- Las cadenas de texto que conforman el texto a mostrar. Se almacenan en *Media Data Box*.
- Una serie de modificadores de texto, que indican como se va a mostrar el texto. De este modo, se incluye la información sobre el color con el que se va a resaltar cada palabra de una cadena de texto, así como la temporización de cada palabra dentro de una cadena de texto. Esta información se almacena en *Media Data Box*.
- Los instantes de tiempos en los que mostrar cada cadena de texto al completo, para de este modo poder mostrar dicho texto en perfecta sincronía con la música. Se almacena en las *boxes* contenidas en *Sample Table Box*, de un modo similar a las pistas de audio.

A continuación, se describirá el proceso de creación del archivo que contiene las letras, y que nuestro codificador incorporará al archivo IM AF. Asimismo, se expondrá el proceso de creación de las *boxes* asociadas a la pista de texto.

Creación del archivo de partida que contiene las letras

Consiste en crear un archivo de texto (extensión .txt) que contenga las letras de la canción y la temporización. Dicho archivo es el que va a utilizar el codificador IM AF para incorporar las letras. Veamos un ejemplo en el que se muestra la estructura que debe tener dicho archivo:

[00:00:20.000] //instante en el que se muestra la siguiente frase en pantalla

Texto a mostrar

[00:00:28.500] //instante en el que deja de aparecer la frase en la pantalla

Este ejemplo mostraría la cadena de texto *"Texto a mostrar"* al transcurrir 20 segundos desde el inicio y dejaría de aparecer al transcurrir 28,5 segundos desde el inicio. Esta estructura se puede repetir las veces que sean necesarias para mostrar todas las letras de la canción.

Los instantes de tiempo en los cuales cada palabra de una cadena de texto se resaltan no son definidos por el usuario, de cara a no hacer muy engorrosa la creación del archivo que contiene las letras. El codificador será el encargado de asignar dichos instantes de tiempo.

Media Data Box

El formato del texto almacenado en *Media Data Box* es el siguiente: dos bytes indicando el tamaño de una cadena de texto, seguido de dicha cadena (que ocupa lo indicado en los dos bytes anteriores). Justo a continuación, se incluyen los modificadores, cuyo tamaño dependerá del número de modificadores empleados. Hay que señalar que en cada cadena de texto no hay necesidad de incluir el carácter de finalización de cadena, ya que el tamaño de dicha cadena está especificado. Esta estructura aparecerá tantas veces como sea necesaria para almacenar todo el texto de la canción.

En nuestra implementación se hace uso de dos modificadores de texto: *Text Hilight Color Box (hclr)* y *Text Karaoke Box (krok).*

El primer modificador de texto que aparece tras la cadena de caracteres es *Text Hilight Color Box.* Son 4 bytes que definen el color con el que se va a resaltar el texto al ritmo

de la música, de un modo similar a como se muestran las letras en un karaoke. Estos 4 bytes se corresponden con el color RGBA *(red green blue alpha)* en hexadecimal. Siendo el primer byte la intensidad de rojo, el segundo byte la intensidad de verde, el tercer byte la intensidad de azul y el cuarto el valor de transparencia. El color elegido para resaltar las letras es el naranja (valor hexadecimal utilizado es #FF6204FF).
El segundo modificador es *Text Karaoke Box (krok)*. Con este modificador conseguimos especificar los instantes de tiempo en los que queremos que se resalten las letras de una cadena de caracteres que está siendo mostrada por pantalla en un instante determinado. En nuestra implementación se resaltará palabra a palabra. El tamaño de *Text Karaoke Box* es variable, y depende del número de palabras que tenga la frase. Podemos encontrar los siguientes campos en *Text Karaoke Box*:

- instante inicial: especifica el instante de tiempo en el que se resalta la primera palabra de una cadena de texto. El valor del instante inicial es un valor relativo al valor especificado en *Sample Size Box*. En nuestro caso, asignamos el valor cero al instante inicial, consiguiendo que la primera palabra de cada cadena de caracteres mostrada por pantalla se resalte en el mismo instante de tiempo en el que se muestra la cadena texto;

- número de entradas: especifica el número de palabras que contiene la cadena de texto que se desea mostrar por pantalla en un instante determinado. Para calcular el número de palabras, se procesa la cadena de texto de entrada y se detectan los espacios en blanco entre palabras;

- instante final: se especifica el instante de tiempo en el que se deja de resaltar cada una de las palabras de la cadena de texto. En esta implementación se asigna un valor que haga que todas las palabras se resalten el mismo periodo de tiempo. De este modo, si por ejemplo tenemos una cadena de caracteres con cinco palabras y que permanece por pantalla cinco segundos, cada una de las cinco palabras permanecerá iluminada un segundo. Hemos optado por esta opción porque si el usuario tuviera que introducir aparte de la temporización de cada cadena de texto, la tem-

porización de cada palabra de dicha cadena, sería un proceso bastante engorroso;

- carácter de inicio: especifica la posición que ocupa el primer carácter de la palabra que queremos resaltar, dentro de la frase completa;
- carácter final: especifica la posición que ocupa el última carácter de la palabra que queremos resaltar, dentro de la frase completa;

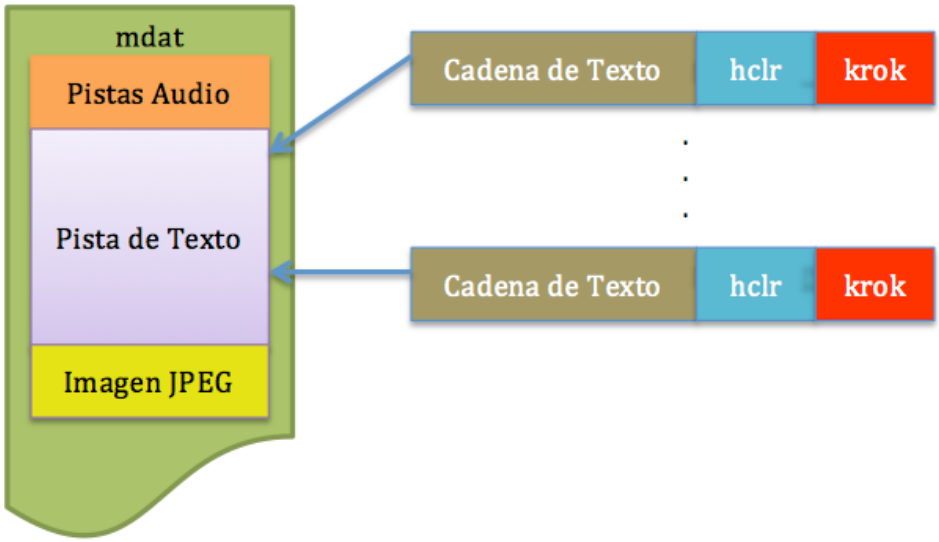

Figura 3.5: Proceso de codificación de las letras en *Media Data Box*

La Figura 3.5 ilustra el proceso de codificación de las letras en *Media Data Box*.
Las *boxes* que trataremos a continuación, contienen la información relativa a la temporización de cada cadena de texto, es decir, en qué instante se debe mostrar por pantalla dicha cadena. Como aclaración, anteriormente hemos explicado como se almacena la información relativa a la iluminación de las palabras dentro de la cadena, mientras que aqui establecemos los instantes de tiempo en los que la cadena completa se mostrara por pantalla.

Sample Size Box (stsz)

En *Sample Size Box* se guarda una tabla, donde en cada entrada se almacena el número de bytes que ocupa cada cadena de texto que aparece por pantalla en un instante

determinado. Este tamaño incluye el de los modificadores de texto.

La Tabla 3.7 muestra un ejemplo real del contenido que almacena *Sample Size Box*. En la Tabla se encuentran almacenadas 70 cadenas de texto, ocupando la primera cadena 45 bytes, la segunda 70 bytes, etc.

num_cadenas	70
tam_cadena[0]	45
tam_cadena[1]	70
tam_cadena[2]	45
....	....
tam_cadena[69]	78

Tabla 3.7: Contenido de *Sample Size Box*

Time To Sample Box (stts)

En *Time To Sample Box* se almacena el tiempo que permanece la cadena de texto por pantalla. De este modo, contendrá una tabla con tantas entradas como cadenas de caracteres existan, más una serie de entradas que representan los instantes en los que no se muestra nada por pantalla (cadenas de texto vacías).

La Tabla 3.8 muestra un ejemplo del contenido de *Time To Sample Box*. Observamos que se almacenan 70 cadenas de caracteres, y el valor *delta* que almacenamos se calcula mediante la siguiente expresión:

$$delta = duracion_intervalo \times escala_tiempo \tag{3.1}$$

La duración del intervalo son los segundos que se desea que la cadena permanezca representada en pantalla. La escala de tiempo es un valor entero que especifica el número de unidades de tiempo que pasan en un segundo. El valor utilizado por defecto como escala de tiempo en IM AF es 1000. Por tanto, el primer valor *delta* de la Tabla 3.8 nos indica que la primera cadena de caracteres aparece por pantalla durante los primeros 22 segundos de reproducción. La siguiente cadena se mostraría los 5 segundos posteriores, y

así sucesivamente. El valor *count* nos indica el número de cadenas de texto consecutivas que duran el mismo tiempo.

num_cadenas	70
(count,delta)[0]	(1,22000)
(count,delta)[1]	(1,5000)
(count,delta)[2]	(1,3831)
....	....
(count,delta)[69]	(1,3000)

Tabla 3.8: Contenido de *Time To Sample Box*

Sample To Chunk Box (stsc)

Las cadenas de texto se agrupan en un único *chunk*, por ello, *Sample To Chunk Box* define una tabla que solo contendrá una entrada.

La Tabla 3.9 muestra un ejemplo de la información almacenada en *Sample To Chunk Box*. Como se vió en el anterior apartado, esta tabla tiene tres variables: la primera especifica el índice del primer *chunk* (dicho índice tiene el valor '1');la segunda variable es el número de cadenas de texto de cada *chunk* (en nuestro ejemplo hay 70 cadenas de caracteres y todas se agrupan en un *chunk*) y la tercera es el número de *chunks* consecutivos con el mismo número de cadenas de texto (como hay solo una entrada, se asigna el valor '1').

num_entradas	1
sample_to_chunk [0]	(1,70,1)

Tabla 3.9: Contenido de *Sample To Chunk Box*

Chunk Offset Box (stco)

Chunk Offset Box contiene una tabla con una única entrada, donde se almacena la posición de memoria en la que aparece el primer byte de la secuencia de texto dentro de *Media Data Box*.

num_entradas	1
pos_mem [0]	22451954

Tabla 3.10: Contenido de *Chunk Offset Box*

La Tabla 3.10 muestra un ejemplo real del contenido de dicha *box*. Como observamos, contiene una única entrada que nos indica que en el byte '22451954' del archivo IM AF se encuentra el primer byte de la primera cadena de texto.

3.3.5. Imagen JPEG

Para la inclusión de una imagen JPEG en el archivo IM AF se utilizan principalmente dos *boxes*: *Media Data Box (mdat)* y *Meta Box (meta)*.

En primer lugar, el codificador calcula el tamaño de la imagen JPEG. A continuación, dicha imagen es copiada en *Media Data Box*, justamente a continuación del último byte de la pista de texto (si la hubiere, en otro caso se almacena a continuación de la última pista de audio).

Diversa información relacionada con la imagen, y que será necesaria en el proceso de decodificación, se almacena en dos *boxes* contenidas en *Meta Box*: *Item Location Box (iloc)* y *Item Information Box (iinf)*.

El tamaño de la imagen y su *offset* en *Media Data Box* se almacena en *Item Location Box*, en particular en los campos *extent_length* y*extent_offset*, respectivamente. El offset es la posición que ocupará el primer byte del archivo JPEG en *Media Data Box* del archivo IM AF.

El nombre de la imagen y el tipo de codificación se almacenan en *Item Information Box*. El tipo de codificación se establece en el campo *content_encoding* y lleva por valor la cadena de caracteres *"jpg"*.

La Figura 3.6 ilustra el proceso de inclusión del archivo JPEG en IM AF.

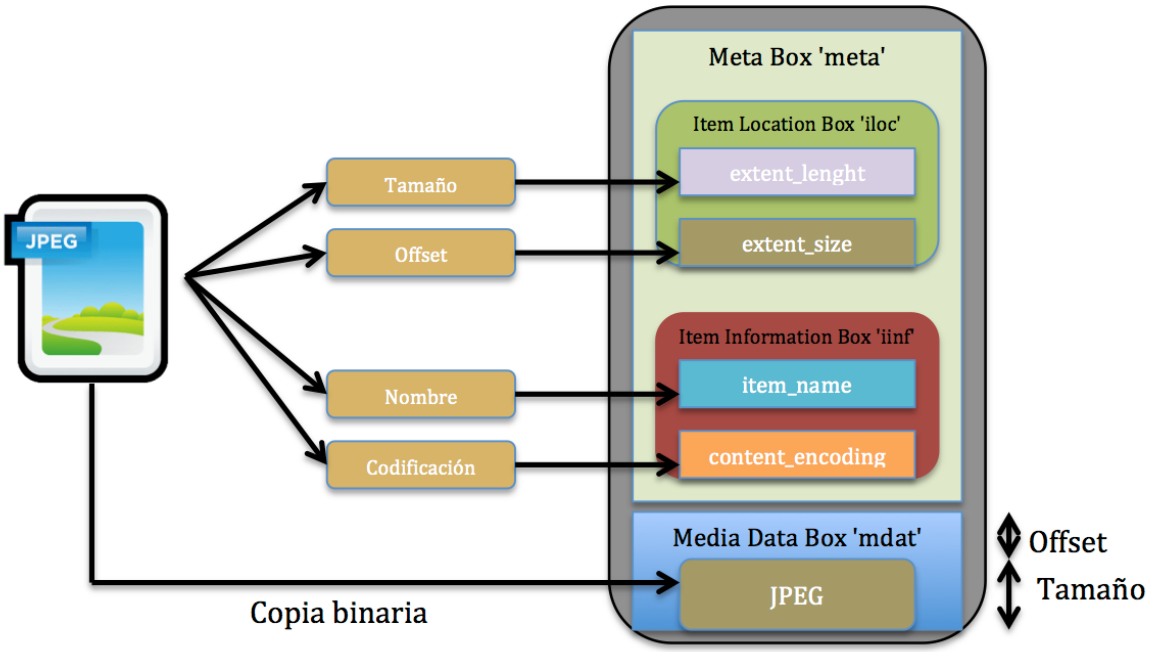

Figura 3.6: JPEG en fichero IM AF

3.3.6. Metadatos

La estructura de *boxes* usada para almacenar los metadatos se muestra en la Tabla 3.11.

meta		Meta Box
	hdlr	Handler Reference Box
	iloc	Item Location Box
	iinf	Item Information Box
	xml	XML Box

Tabla 3.11: *Boxes* usadas para los metadatos

Meta Box tiene dos funciones diferenciadas:

- Alojar información descriptiva relacionada con el artista, álbum, canción, etc;

- Almacenar información acerca de la imagen JPEG incluida en el fichero IM AF. En concreto, tal y como se desarrolló en el anterior apartado, contiene el tamaño de la imagen y su offset en el fichero.

Handler Reference Box

Define la estructura o el formato de los contenidos de *Meta Box*. El campo *handler_type* indica el formato de los contenidos, almacenando en este caso la cadena *"meta"*. Por otro lado, *name* es una cadena de caracteres que da un nombre legible a la *box*, para propósitos de depuración del código. Toma el valor de *"mp7t"*.

Item Location Box

Como ya se comentó en el anterior apartado, proporciona el tamaño en bytes que ocupa la imagen JPEG y el *offset* absoluto de dicha imagen respecto del inicio del fichero IM AF. Esta información se encuentra en los campos, *extent_offset* y *extent_length*, respectivamente.

Item Information Box

Proporciona información extra sobre el fichero JPEG :

- Nombre: el campo *item_name* contiene *"image"*.
- Tipo de codificación: el campo *content_encoding* contiene *"jpg"*.

Esta información no es obligatoria y de hecho el reproductor IM AF proporcionado por MPEG no muestra esta información. Se usa únicamente para fines de depuración.

XML Box

La información sobre el artista, álbum, canción, etc se encuentra almacenada en formato Extensible Markup Language (XML). El codificador incluye esta información en el fichero IM AF resultante escribiendo una cadena de caracteres con formato XML, donde se incluye toda la información deseada.

3.3.7. Grupos

Para la creación de grupos en un archivo IM AF se utilizan dos *boxes*: *Group Container Box* y *Group Box*. La Tabla 3.12 muestra la estructura de boxes usadas.

grco		Group Container Box
	grup	Group Box

Tabla 3.12: *Boxes* usadas para los grupos

Group Container Box

Almacena el número de grupos que albergará el archivo IM AF. Dicho valor vendrá especificado por el usuario del codificador IM AF.

Group Box

En el archivo IM AF aparecerá *Group Box* tantas veces como el número de grupos que se hayan definido en *Group Container Box*. Cada *Group Box* contiene información específica de un grupo, siendo dicha información almacenada en los siguientes campos :

- identificador de grupo: número entero de 32 bits que identifica el grupo. Este valor debe ser representado usando valores desde #80000000 a #FFFFFFFF. El codificador los asigna consecutivamente desde #80000001;
- nombre de grupo: cadena de caracteres que asigna un nombre legible al grupo. Dicha cadena es proporcionada por el usuario del codificador;
- descripción del grupo: cadena de caracteres que describe de forma resumida el grupo. El usuario es el encargado de introducir dicha cadena;
- número de elementos: número entero que indica el número de elementos que conforman el grupo. Se corresponde con el número de pistas de audio que estarán presentes en el grupo. También viene dado por el usuario;

- identificador de elemento: es un array que contiene el identificador de cada una de las pistas que conforman el grupo. El identificador de cada pista viene dado por el usuario, y será un número entero comprendido entre el uno y el número de pistas que contenga el archivo IM AF;
- modo de activación de grupo: es un *flag* que define la forma en la que los elementos de un grupo deben ser activados cuando un grupo es activado. La implementación elegida hace que se activen todos los elementos del grupo. En el caso de que se haya definido una regla min/max, el máximo número de elementos activados será el mismo que el máximo número de elementos que define dicha regla;
- volumen de referencia del grupo: especifica el volumen que debe ser aplicado al grupo cuando se activa. El volumen de referencia es introducido por el usuario del codificador IM AF, y deberá estar comprendido en el rango [0,100].
- *flags*: sirve para indicar al reproductor si se desea mostrar la información referente al grupo creado o no. Nuestro codificador habilitará por defecto que dicha información sea mostrada en el reproductor. El valor que toma el campo *flags* para ello es #02;

3.3.8. *Presets*

A continuación se detallará el proceso de creación de las *boxes* empleadas en los *presets*. De los tipos de *presets* de la Tabla 2.2, se han implementado los dos siguientes:

- Pista con volumen estático
- Pista con volumen dinámico aproximado

La Tabla 3.13 muestra la estructura de *boxes* utilizada. Hay tantos *Preset Box (prst)* como *presets* definidos en el archivo IM AF, y a su vez todas estas *boxes* se encuentran en un único *Preset Container Box (prco)*.

Preset Container Box

Encontramos los siguientes campos:

prco		Preset Container Box
	prst	Preset Box

Tabla 3.13: *Boxes* usadas para los *presets*

- número de *presets*: es el número de *presets* que se desean definir. El usuario es el encargado de indicarlo. En nuestra implementación, el número máximo de *presets* que pueden ser incluidos en el archivo IM AF es de diez, y viene definido por la variable global *maxpresets*.

- identificador de *preset* por defecto: se trata de un entero que indica el identificador del preset activo en el estado inicial, sin la interacción del usuario. Por defecto, el codificador le asigna el identificador más bajo que se haya definido.

Preset Box

Como hemos comentado antes, se crea una *Preset Box* por cada *preset*. Contiene los siguientes campos:

- *flags*: sirve para indicar al reproductor si se desea mostrar la información referente al *preset* o no. El valor asignado es "0x02", habilitando que dicha información sea mostrada.

- identificador de *preset*: es un número entero que identifica el *preset*. El codificador asigna automáticamente a cada *preset*, valores desde el uno en adelante.

- número de elementos del *preset*: en nuestro implementación, todas las pistas formarán parte del *preset*. Por tanto, el codificador asigna el número de pistas de audio existentes.

- identificador de cada elemento del *preset*: se trata de un identificador a cada pista de audio que conforma el *preset*. Los identificadores asignados automáticamente por el codificador serán números enteros consecutivos, comenzando por el número uno.

- tipo de *preset*: es un entero que indica el tipo de *preset* que se desea incluir, de entre los que se definen en la Tabla 2.2. El usuario introducirá el número entero que identifica el tipo de *preset* deseado. Una estructura *switch/case* en el codificador será la encargada de realizar la implementación apropiada.

- volumen global del *preset*: es un entero que indica el volumen de salida global del *preset*. En este caso, el codificador asigna automáticamente un valor de "100".

- volumen de los elementos del *preset*: es un entero que indica el volumen de cada pista de audio del *preset*. Dichos valores son proporcionados por el usuario, asignando un valor entero entre "0" y "100".

- número de canales de audio: es un entero que indica el número de canales de las pistas de audio. Nuestro codificador IM AF admite pistas estéreo únicamente, por tanto, asignará automáticamente el valor "2".

- nombre del *preset*: es una cadena de caracteres que identifica de forma legible el *preset*. El codificador asigna automáticamente dicha cadena, dependiendo del tipo de *preset* elegido. Por ejemplo, en el caso de seleccionar un *preset* con pistas en las que el volumen permanece estático, asignará la cadena *"Static track volume preset"*.

En el caso del *preset* "Pista con volumen dinámico aproximado", se necesitan además un conjunto adicionales de parámetros. El estándar IM AF usa una representación eficiente que define el cambio de volumen en un intervalo de tiempo, en vez de definir las variaciones en cada muestra. De este modo, el espacio necesario de almacenamiento para los *presets* dinámicos se reduce. El cambio de volumen viene representado por :

- muestra inicial;

- duración del intervalo de tiempo (número de muestras) en la que el cambio de volumen tiene lugar;

- nivel de volumen al final del intervalo;

Este tipo de *preset* se ha utilizado para crear un efecto *fade-in* y *fade-out*. El efecto *fade-in* consiste en aumentar gradualmente el nivel de volumen de una pista, partiendo desde el silencio. Por su parte, en el efecto *fade-out* se parte de cierto nivel de volumen y se va disminuyendo progresivamente dicho nivel hasta alcanzar el silencio. El usuario del codificador IM AF solo tendrá que elegir entre *fade-in* o *fade-out* y el codificador se encarga de establecer una configuración predeterminada de los parámetros.

3.3.9. Reglas

A continuación, se va a proceder a explicar la implementación de las reglas en el codificador IM AF. La Tabla 3.14 muestra la estructura de *boxes* empleada.

ruco		Rule Container Box
	rusc	Selection Rule Box
	rumx	Mixing Rule Box

Tabla 3.14: *Boxes* usadas para definir las reglas

Rule Container Box

Contiene información general sobre las reglas aplicadas en los elementos de un fichero IM AF:

- número de reglas de selección: es un entero que especifica el número de reglas de selección contenidas en *Rule Container Box*. El usuario podrá elegir entre la creación de una o ninguna regla de selección.
- número de reglas de mezcla: es un entero que especifica el número de reglas de mezcla contenidas en *Rule Container Box*. Al igual que con las reglas de selección, el usuario podrá crear o no una regla de mezcla.

Selection Rule Box

Contiene información específica sobre la regla de selección definida:

- identificador de regla de selección: entero que identifica la regla de selección. El codificador asignará automáticamente el valor "1".

- tipo de regla de selección: entero que identifica el tipo de regla de selección. El usuario será el encargado de introducir dicho valor. La Tabla 3.15 muestra los distintos valores que puede tomar este campo, así como su significado.

Tipo de regla de selección	**Descripción**
0	Regla Min/Max
1	Regla de Exclusión
2	Regla de no silencio
3	Regla de Implicación

Tabla 3.15: Tipos de reglas de selección

  Una estructura *switch/case* será la encargada de seleccionar la implementación, en función del tipo de regla de selección elegida por el usuario.

- identificador del elemento: entero que representa el identificador del elemento involucrado en esta regla de selección. En el caso de la regla Min/Max, el codificador asignará automáticamente el identificador del primer grupo creado. En los otros casos, dicho valor lo introdurá el usuario.

- mínimo número de elementos: entero que se usa únicamente en la regla min/max . Especifica el mínimo número de elementos que deberían ser reproducidos simultáneamente. El valor debe ser introducido por el usuario. En caso contrario, el valor por defecto es cero.

- máximo número de elementos: entero usado únicamente en la regla min/max. Especifica el máximo número de elementos que deberían ser reproducidos simultáneamente. Dicho entero debe ser especificado por el usuario. En otro caso, el valor por defecto es igual al número de elementos que contiene el grupo implicado.

- identificador del elemento clave: entero que representa el identificador del elemento sobre el cual se aplica la regla. Es usado únicamente en reglas de exclusion e implicación. Dicho identificador debe ser introducido por el usuario.

- descripción de la regla de selección: cadena de caracteres que proporciona una descripción legible de la regla de selección. Por ejemplo, *"Min/Max Rule"*, *"Exclusion Rule"*,etc. Esta cadena de caracteres es asignada automáticamente por el codificador, en función de la regla seleccionada.

La Tabla 3.16 resume los parámetros necesarios en cada una de las reglas de selección.

Tipo	**ID elemento**	**ID elemento clave**	**Min. num. elementos**	**Max. num. elementos**	**Descripción regla**
0	ID grupo	No	Si	Si	Si
1	Si	Si	No	No	Si
2	Si	No	No	No	Si
3	Si	Si	No	No	Si

Tabla 3.16: Parámetros en reglas de selección

Mixing Rule Box

Contiene información específica sobre la regla de mezcla definida:

- identificador de regla de mezcla: entero que identifica unívocamente la regla de mezcla. El codificador asigna automáticamente el valor "1".

- tipo de regla de mezcla: entero que identifica el tipo de regla de mezcla. El usuario es el encargado de establecer dicho valor. La Tabla 3.17 muestra los distintos valores que puede tomar este campo, así como su significado.

  Una estructura *switch/case* será la encargada de seleccionar la implementación, en función del tipo de regla de mezcla elegida por el usuario.

Tipo de regla de mezcla	Descripción
0	Regla de equivalencia
1	Regla superior
2	Regla inferior
3	Regla de límites

Tabla 3.17: Tipos de reglas de mezcla

- identificador de elemento: entero que representa el identificador del elemento involucrado en la regla de mezcla. Este campo es usado en los cuatro tipos de reglas de mezcla. Dicho valor es introducido por el usuario.
- volumen mínimo: valor fijo que especifica el volumen mínimo del elemento. Es usado únicamente en el caso de definir una regla de límites . El valor vendrá definido por el usuario.
- volumen máximo: valor fijo que especifica el máximo volumen del elemento. Es usado únicamente en el caso de definir una regla de límites. El valor vendrá definido por el usuario.
- identificador del elemento clave: entero que describe el identificador del elemento sobre el que se aplicará la regla. Es usado para las siguientes reglas: regla de equivalencia, regla superior, regla inferior.
- descripción de la regla de mezcla: cadena de caracteres que describe de forma legible la regla de mezcla. Por ejemplo, las cadenas “Regla de Equivalencia”, “Regla Superior”,etc. Las asigna automáticamente del decodificador.

La Tabla 3.18 resume los parámetros necesarios en cada una de las reglas de mezcla.

Tipo	**ID elemento**	**ID elemento clave**	**Min. volumen**	**Max. volumen**	**Descripción regla**
0	Si	Si	No	No	Si
1	Si	Si	No	No	Si
2	Si	Si	No	No	Si
3	Si	No	Si	Si	Si

Tabla 3.18: Parámetros en reglas de mezcla

Capítulo 4. Integración del codificador IMAF en Sonic Visualiser

Este capítulo se centrará en la integración de nuestro codificador IM AF en el software Sonic Visualiser. De este modo, es posible la creación de archivos IM AF de una forma flexible, sencilla y agradable al usuario, mediante el uso de una interfaz gráfica. Aún cobra más sentido dicha integración, mediante el desarrollo de un decodificador IM AF en Sonic Visualiser, de forma que se pueda realizar la creación y reproducción de dichos archivos interactivos en dicho software, tal y como se realiza en este proyecto. Una característica destacable de Sonic Visualiser es que permite el uso de determinados *plugins*. El formato de *plugins* soportado por Sonic Visualiser se denomina VAMP, y se trata de una API *(Application Programming Interface)* que permite desarrollar algoritmos de procesado digital de señal. De este modo, las posibilidades del formato IM AF se amplían sustancialmente al integrarse en Sonic Visualiser.

El capítulo comenzará con una breve introducción a Sonic Visualiser, donde se comentará a grandes rasgos el funcionamiento y estructura de dicho software. La siguiente sección introducirá la librería Qt, usada en el desarrollo de Sonic Visualiser, y que será ampliamente utilizada en el desarrollo de la interfaz. Por último se detallará la integración del codificador en Sonic Visualiser.

4.1. Introducción a Sonic Visualiser

Sonic Visualiser es un software de código abierto y multiplataforma (Linux, OS/X y Windows), desarrollado en el *Centre for Digital Music, Queen Mary, University of London.* Está diseñado especialmente para los investigadores del área del procesado de la señal que pretenden estudiar en un entorno amigable y flexible el contenido de un fichero de audio. De este modo, la aplicación permite el análisis, visualización y anotación de ficheros de audio. Ha sido elegido para realizar la integración del códec IM AF, en lugar de otras herramientas de análisis musical debido a su uso extensivo, ya que cuenta con más de 10000 usuarios activos (en Abril de 2014). Su principal fortaleza reside en que soporta *plugins* desarrollados para análisis automático. La Figura 4.1 muestra el aspecto de la interfaz de Sonic Visualiser, con dos pistas de audio activas.

Sonic Visualiser cuenta con una serie de herramientas de visualización de propósito general, que permiten la visualización de espectrogramas y formas de onda. También incluye métodos sencillos para la anotación de audio, usando el teclado, ratón o dispositivos Musical Instrument Digital Interface (MIDI). Es extensible con un número de métodos de análisis automático de audio, usando el formato de *plugin* VAMP. Por último, proporciona la posibilidad de importar y exportar las anotaciones en distintos formatos, incluyendo *Music Ontology RDF.*

La aplicación está desarrollada en C++ y hace uso de la librería Qt para el desarrollo de la interfaz.

4.1.1. Herramientas de visualización

Por defecto, el programa muestra la forma de onda cuando se carga un fichero de audio. Es posible aumentar la zona deseada de dicha visualización o desplazarse por ella, utilizando una serie de herramientas de navegación.

También se incluye la visualización del espectrograma, teniendo la posibilidad de visualizarlo directamente con tres juegos de parámetros, usando las letras *G*, *M* y *K*. Con la tecla *G* se visualiza un espectrograma genérico que cubre todo el rango de frecuencias

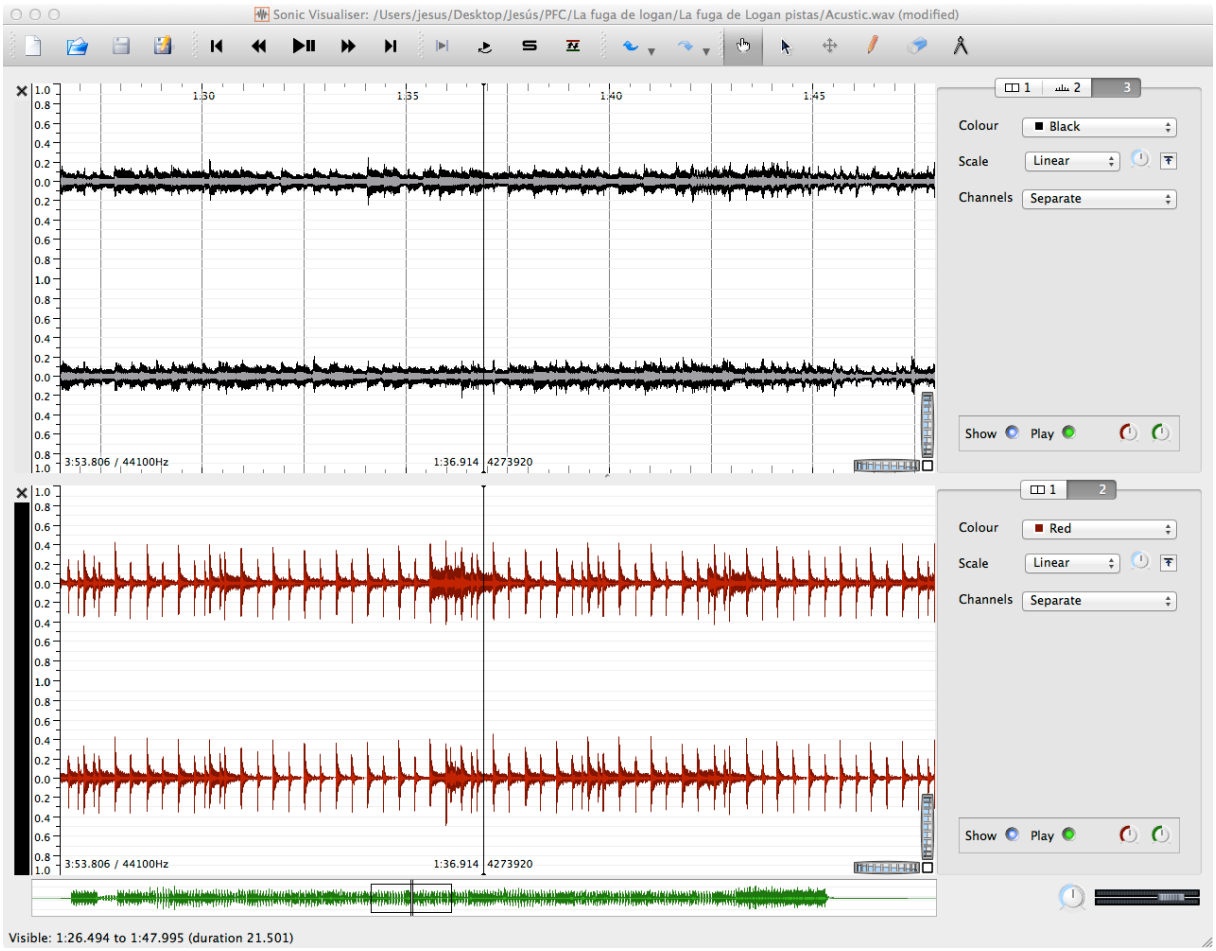

Figura 4.1: Aspecto de la interfaz de Sonic Visualiser

con una escala lineal de frecuencia. La tecla *M* muestra un espectrograma que enfatiza el rango de frecuencias en el que se suelen encontrar las melodías y la tecla *K* muestra un espectrograma similar al obtenido al pulsar la tecla *M*, pero dotando de mayor libertad al usuario a la hora de elegir los parámetros.

Sonic Visualiser usa paneles y capas para permitir al usuario apilar distintas vistas unas encimas de otras, estando todas alineadas en el eje de tiempo. Muestra cualquier número de paneles (distintas areas de la ventana, verticalmente apiladas) y cada panel puede mostrar cualquier número de capas (representaciones de datos superpuestos). La Figura 4.2 muestra una captura de pantalla de un panel con cuatro capas: forma de onda (color negro), instantes de tiempo(color azul), divisiones de tiempo (color gris), valores del eje Y en el tiempo (color púrpura).

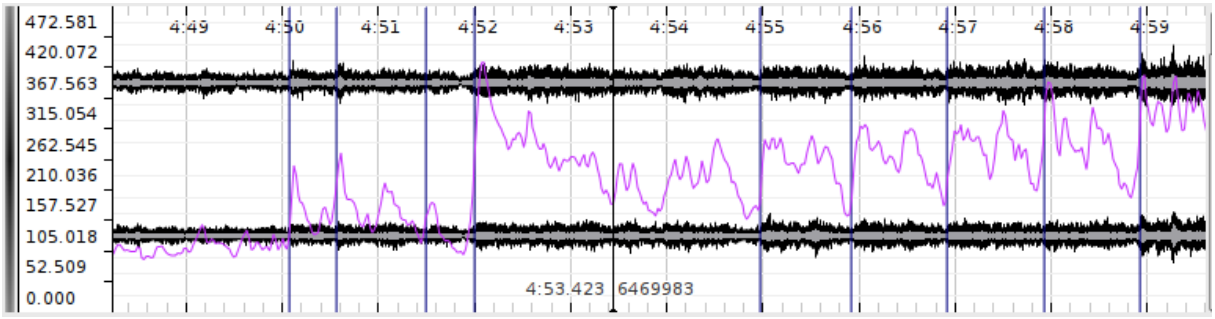

Figura 4.2: Panel con cuatro capas

4.1.2. Capas de anotación

Es posible mostrar en Sonic Visualiser capas de anotación de varios tipos, como por ejemplo instantes de tiempo, etiquetas o imágenes. De este modo, el usuario puede anotar puntos de una pista de audio, añadiendo una capa de instantes de tiempo y haciendo clic con el ratón o usando el teclado para almacenar los instantes de tiempo deseados mientras el audio está sonando. Las capas de anotación serán de gran interés para nosotros, pues servirán para mostrar las letras de la canción IM AF en perfecta sincronía con el audio.

4.1.3. *Plugins*

Sonic Visualiser soporta el formato de *plugins* VAMP, con los que podemos realizar funciones tales como: detección de acordes, detección de las notas de una melodía o alineación de pistas. También admite *plugins* de efectos LADSPA, que permiten generar nuevos ficheros de audio que pueden ser analizados del mismo modo.
Los resultados de aplicar un *plugin* serán mostrados en una nueva capa del tipo apropiado y sobre el panel activo. Si se desea que dichos resultados sean mostrados en un panel distinto, hay que crear un nuevo panel antes de ejecutar el *plugin*. El *plugin*, a partir del fichero de audio de entrada y los parámetros que seleccione el usuario, generará valores que serán mostrados por pantalla. La Figura 4.3 muestra algunos ejemplos de salidas de un *plugin* VAMP en Sonic Visualiser.

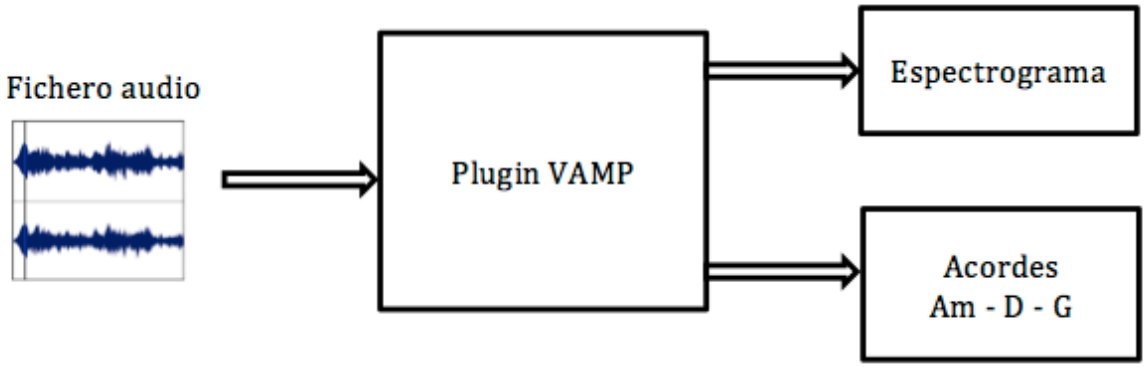

Figura 4.3: Posibles salidas de un *plugin* VAMP en Sonic Visualiser

4.1.4. Sumario de clases de Sonic Visualiser

El software Sonic Visualiser está estructurado en clases con funcionalidades bien diferenciadas. A continuación se describirán brevemente las clases más importantes según su funcionalidad:

- Modelos de datos: Un modelo es algo que contiene, o sabe como obtener datos. Por ejemplo, la clase *WaveFileModel*, es un modelo que conoce como obtener datos de un fichero de audio.

- Capas gráficas: Una capa es algo que sabe como dibujar las partes que conforman un modelo en una línea de tiempo. Por ejemplo, la clase *WaveformLayer* es una capa que permite dibujar formas de onda.

- Clase de la ventana principal: Se trata de la clase *MainWindow*. Será empleada en la integración del códec IM AF en Sonic Visualiser, donde se incluirán nuevos métodos que permitirán la creación de un archivo IM AF mediante una interfaz gráfica, así como la reproducción de ficheros IM AF.

- Conversión de un modelo (por ejemplo, audio) en otro (por ejemplo, más audio o una curva extraída de éste).

- Lectura de ficheros y escritura: Por ejemplo la clase *AudioFileReader*.

- Cálculo de la FFT: la clase *FFTModel*.

- Reproducción de audio: por ejemplo las clases *AudioCallbackPlaySource* y *AudioGenerator*.

- *Widgets*: muestran propiedades editables. Por ejemplo, la clase *PropertyBox* usada para las propiedades de una capa.

4.2. La libreria Qt

Qt es una biblioteca multiplataforma utilizada principalmente para el desarrollo de aplicaciones con interfaz gráfica de usuario. Qt es desarrollada como software libre y es de código abierto. Para su implementación han participando desarrolladores de Nokia, Digia, entre otros. Utiliza el lenguaje de programación C++ de forma nativa, aunque adicionalmente puede ser utilizada en otros lenguajes de programación. Entre las aplicación populares que hacen uso de Qt se encuentran: Google Earth, Skype o VLC Media Player.
La aplicación Sonic Visualiser también hace uso de la librería Qt, por tanto para la integración del códec IM AF en Sonic Visualiser, es de gran importancia familiarizarse con el funcionamiento de dicha biblioteca.

4.2.1. Señales y *slots* en Qt

Cuando se desarrolla una interfaz gráfica de usuario, la forma de diseñar el programa cambia con respecto a cuando se realiza secuencialmente. El usuario puede pulsar los botones en cualquier momento, y el sistema debe responder a esos eventos. Además, es posible que la acción sobre un componente suponga un cambio en otro, por lo que es deseable que cualquier objeto se pueda comunicar con cualquier otro.
En Qt se genera una señal cuando hay un evento. Los componentes de Qt tienen muchas señales predefinidas, aunque es posible crear nuestras propias señales.
Un *slot* es una función que se invoca como respuesta a una señal. Los componentes de Qt tienen muchos *slots* predefinidos, aunque también es posible añadir nuestros propios *slots*.

4.2.2. Clases de Qt

En este apartado se describen someramente algunas de las clases predefinidas en Qt que se usaran en la integración.

- QObject: clase de la cual se derivan muchas de las clases importantes de la librería Qt. Todas las clases que heredan de QObject o algunas de sus subclases pueden contener señales y *slots*.

- QWidget: clase base de todos los objetos de la interfaz de usuario. Esta clase hereda de QObject. El *widget* es el elemento atómico de la interfaz de usuario, ya que recibe los eventos del teclado, ratón y otros eventos del sistema de ventanas.

- QAction: proporciona una interfaz de acciones de usuario, y puede ser insertada en *widgets*.

- QCheckBox: proporciona una casilla de verificación con una etiqueta de texto. Consta de una casilla que permite dos estados distintos, marcado y desmarcado.

- QString: esta clase proporciona una cadena de caracteres *Unicode*, el cual es un

estándar de codificación de caracteres diseñado para facilitar el tratamiento informático.

- QPushButton: proporciona un botón, el cual al ser pulsado ejecuta una determinada acción. Ejemplos de este tipo de botón los encontramos en los botones *Aceptar* o *Cancelar*.

- QFileDialog: proporciona al usuario un cuadro de diálogo que le permite seleccionar ficheros o directorios.

4.3. Integración del codificador IM AF

4.3.1. Introducción

La integración del codificador IM AF se ha llevado a cabo haciendo uso de Qt Creator. Qt Creator es un entorno de desarrollo integrado creado para el desarrollo de aplicaciones, haciendo uso de las bibliotecas Qt. El compilador elegido en Qt Creator será MinGW (*Minimalist GNU for Windows*), que es una implementación de los compiladores GCC para la plataforma Win32.
Los pasos realizados para la integración del codificador IM AF en Sonic Visualiser son los siguientes:

- Compilación del código fuente de Sonic Visualiser

  En Qt Creator se crea un nuevo proyecto con todos los ficheros que conforman el código fuente de Sonic Visualiser y posteriormente se configuran todos los parámetros necesarios para que dicha compilación sea satisfactoria. Entre estos parámetros se encuentran la selección del compilador (MinGW) y sus ajustes, así como la instalación de las librerías necesarias en las rutas correspondientes.

  Llegados a este punto, me gustaría señalar que este paso consumió una cantidad significativa de horas y fue de gran ayuda el asesoramiento por parte de Chris Cannam, desarrollador de Sonic Visualiser.

- Análisis de la estructura del código de Sonic Visualiser

  Una vez compilado Sonic Visualiser, se procede al análisis de la estructura del código fuente, con el fin de conocer los ficheros donde incluir las funciones necesarias para dotar al programa con la funcionalidad del codificador IM AF.

- Desarrollo de la interfaz gráfica e integración del codificador IM AF en Sonic Visualiser

  En este paso se procede a la creación de la interfaz gráfica que permitirá la generación del archivo IM AF de una forma amigable y flexible. De este modo, se dotará a Sonic Visualiser de la capacidad de generar ficheros IMAF, con las características del codificador expuesto en el capítulo anterior.

4.3.2. Desarrollo de la interfaz gráfica e integración

La integración del codificador IM AF en Sonic Visualiser supone la adición de las clases que conforman el codificador que habiamos desarrollado previamente, con ciertas modificaciones ya que todo lo referente a la interacción con el usuario deja de ser a través de línea de comandos y pasa a realizarse a través de una interfaz gráfica. Asimismo, se procede a la modificación de la interfaz gráfica de Sonic Visualiser, de modo que ahora se incluya una nueva opción en el menú de Sonic Visualiser, que permita desplegar la interfaz que permite controlar nuestro codificador.

Modificación del menú *"File"* de Sonic Visualiser

El objetivo es la adición de una nueva opción en el menú *"File"* de Sonic Visualiser, de modo que al hacer clic sobre dicha opción se abra la interfaz que permite la creación de un archivo IM AF.

La Figura 4.4 muestra la opción añadida al menú de Sonic Visualiser.

Para lograr tal fin, se crea un objeto de la clase *QAction*. Un objeto de la clase QAction puede contener un icono, menú de texto o un atajo de teclado. En este caso, contendrá un menú de texto con el siguiente texto: *"Export IM AF File"*. De este modo, se añade al

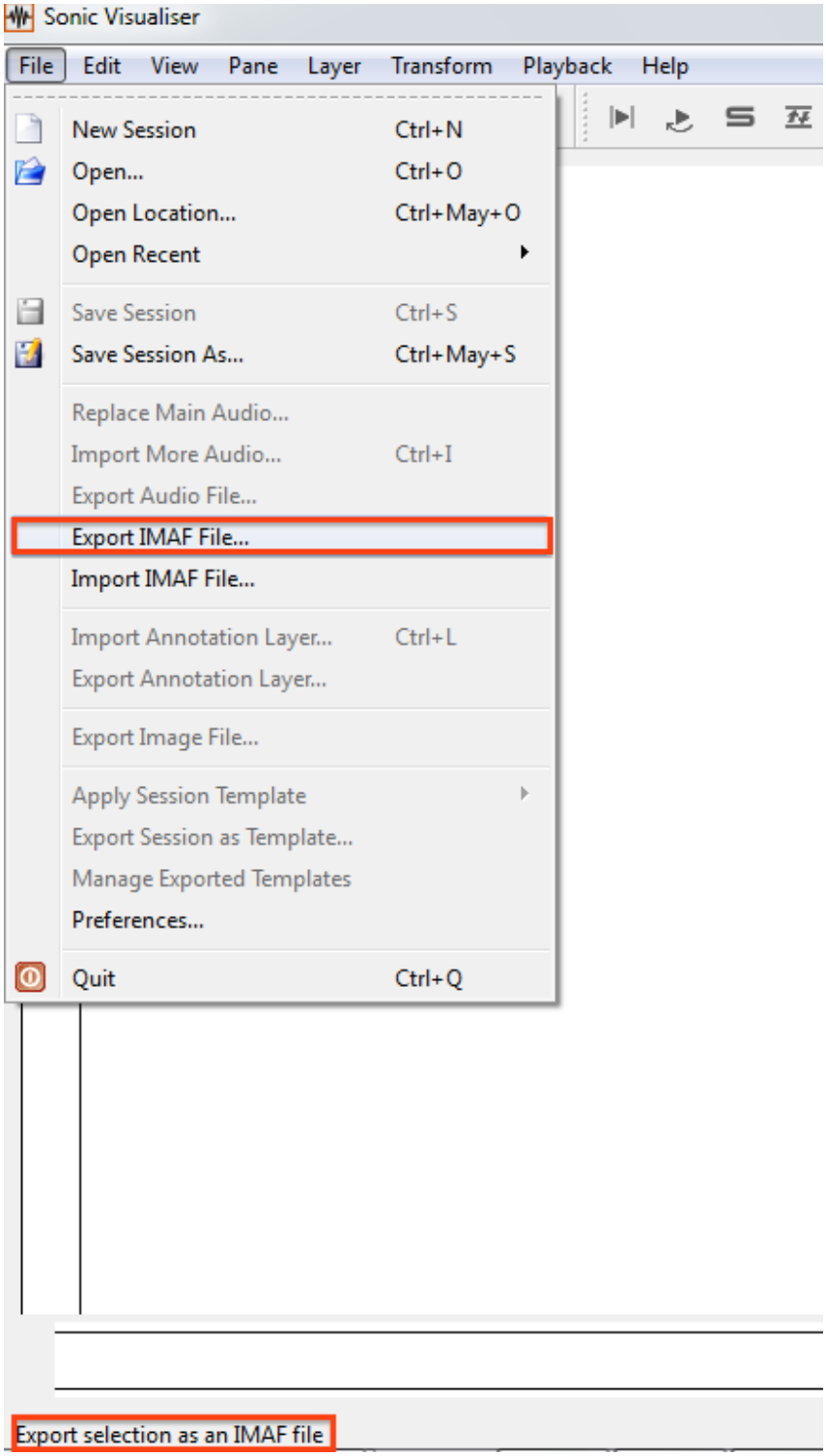

Figura 4.4: Menú que permite invocar el codificador IM AF en Sonic Visualiser

menú *"File"* de Sonic Visualiser la opción *"Export IM AF File"*. Asimismo, cuando se sitúe el puntero del ratón sobre dicha opción del menú, se muestra en la barra de estado que aparece en la zona inferior izquierda lo siguiente: *"Export selection as an IM AF File"*. Esto se hace invocando al método definido en la clase QAction llamado *setStatusTip*. Para añadir al menú la opción se usa el método *addAction*. Por último se conecta señal y *slot* mediante el método *connect*, de forma que al hacer clic sobre la función del menú, se invoque la interfaz del codificador. Hay que señalar que las pistas de audio que incluirá el archivo IM AF serán las que estuvieran abiertas antes de seleccionar en el menú la opción *"Export IM AF File"*.

Creación de la interfaz gráfica del codificador

La Figura 4.5 muestra el aspecto de la interfaz gráfica de nuestro codificador IM AF en Sonic Visualiser.

Encontramos distintas regiones en la ventana:

- Reglas de selección: Podemos elegir entre los cuatro reglas de selección definidas en el estándar, Regla Min/Max (*Min/Max Rule*), Regla de no silencio (*Not Mute Rule*), Regla de exclusión (*Exclusion Rule*) y Regla de Implicación (*Implication Rule*). Para poder elegir entre las cuatro reglas, se definen cuatro botones de opción, los cuales se corresponden con cuatro objetos del tipo QRadioButton. El emplazamiento de dichos botones se realiza con el método *move*, indicando las coordenadas x e y.

  Para la selección de los parámetros de configuración de la regla seleccionada (*Parameter 1* y *Parameter 2*) se hacen uso de dos objetos de tipo QSpinBox.

- Reglas de mezcla: nos encontramos con las cuatro opciones definidas en el estándar, Regla de equivalencia (*Equivalence Rule*), Regla inferior (*Lower Rule*), Regla superior (*Upper Rule*), Regla de límites (*Limit Rule*).

  En el caso de las reglas de mezcla, se definen cuatro parámetros, donde los dos primeros parámetros (*Parameter 1 y Parameter 2*) indican las pistas sobre las que aplicar la regla y los otros dos parámetros nos permiten indicar los valores de volu-

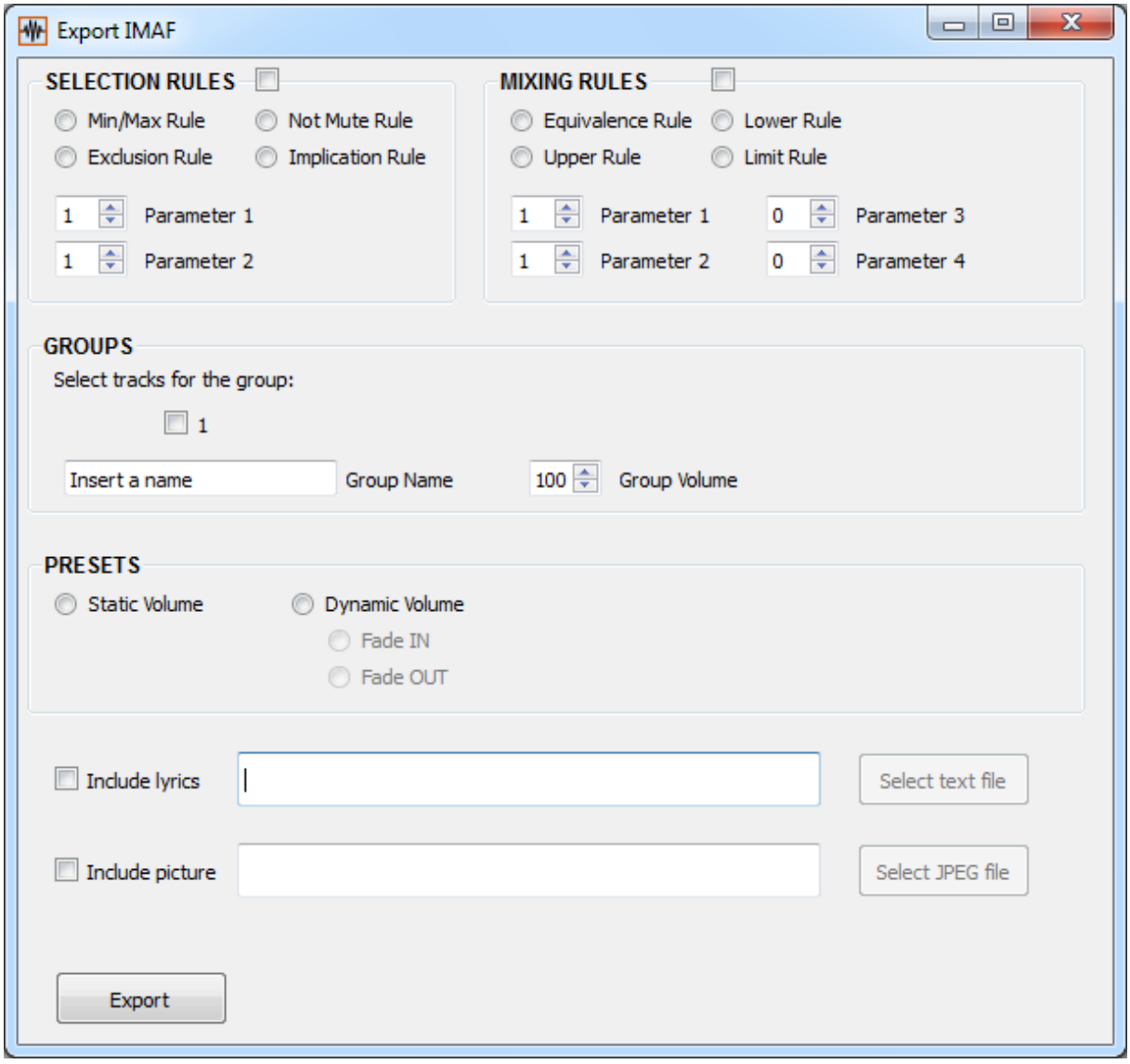

Figura 4.5: Interfaz del codificador IM AF

men mínimo y máximo (solamente utilizado en *Limit Rule*).

- Grupos: se muestra una casilla de verificación por cada una de las pistas que incluye el archivo IM AF, permitiendo seleccionar las pistas que conformarán el grupo. Estas casillas de verificación se crean mediante objetos del tipo QCheckBox. El nombre asignado al grupo se almacena en una variable de tipo QString.

- *Presets*: se permite elegir al usuario entre *preset* con volumen estático o dinámico. En el caso de volumen dinámico, se permite la selección de un efecto *fade-in* o *fade-out*.

- Letras: para la elección del archivo que contiene las letras de la canción con la temporización, se define un objeto de tipo QCheckBox mediante el cual el usuario puede elegir si desea incluir letras o no.En caso afirmativo, al hacer clic en el botón *"Select text file"* (definido mediante un objeto de tipo QPushButton), se abre un cuadro de dialogo que permite seleccionar dicho fichero.

- Imagen JPEG: del mismo modo que para las letras, un objeto de tipo QCheckBox le da la opción al usuario de incluir una imagen o no, y un objeto de tipo QPush-Button permite abrir el cuadro de diálogo para seleccionar la imagen deseada una vez pulsado el botón.

- Botón *Export*: Una vez se ha terminado la configuración de todas las características del archivo IM AF a generar, se pulsa el botón *Export*, el cual permite abrir un cuadro de diálogo del tipo QFileDialog y así elegir la ruta y el nombre del archivo IM AF a crear. Una vez realizado ésto, se invoca al codificador IM AF que habiamos desarrollado previamente, con las modificaciones pertinentes en lo referente a la interacción con el usuario.

Capítulo 5. Desarrollo del decodificador IM AF en Sonic Visualiser

Una vez realizada la integración del codificador IM AF en Sonic Visualiser, se procede al desarrollo del decodificador. De este modo, es posible la creación y reproducción de archivos IM AF en Sonic Visualiser.
Las características del decodificador que desarrollamos son las siguientes:

- Decodificación de las pistas de audio del archivo IM AF, presentando las pistas de audio por separado y pudiendo controlar individualmente cada una de ellas.
- Decodificación de las letras de la canción, mostrándolas en perfecta sincronía con el audio.
- Creación de un panel con los controles de volumen de cada una de las pistas de audio, de forma que el usuario pueda manipular cómodamente dichos volúmenes en el momento de la reproducción.

A continuación se va a proceder a explicar el proceso realizado por el decodificador para extraer las pistas de audio y la pista de texto. Para finalizar, se presentará la interfaz gráfica del decodificador. Para una mayor compresión de los siguientes apartados, puede ser de gran utilidad la Tabla 3.1, donde se presenta la estructura que presentan las *boxes* de un fichero IM AF.

5.1. Pistas de audio

En primer lugar, el decodificador realiza la apertura del fichero IM AF en modo binario. Las pistas de audio se encuentran almacenadas en *Media Data Box (mdat)*, de este modo el algoritmo realiza la búsqueda de la cadena "mdat". Los cuatro bytes anteriores a dicha cadena se corresponden con el tamaño de *Media Data Box*. Este tamaño se almacena, ya que es de utilidad en cálculos posteriores.

Con el fin de obtener el número total de pistas contenidas en el fichero (pistas de audio y texto), se procede a la búsqueda de *Movie Header Box (mvhd)*, donde el campo *next_track_ID* indica el número de pistas de audio contenidas en el archivo IM AF más uno. Dicho campo se encuentra 96 bytes después del último byte del campo que indica el tipo de *Movie Header Box (mvhd)*.

El algoritmo continua leyendo el fichero IM AF hasta encontrar *Handler Reference Box (hdlr)* de la primera pista de audio. Ahí encontramos el campo *handlerType*, el cual indica si la pista en cuestión es de audio o de texto. Este campo se haya localizado ocho bytes después del último byte del campo que indica el tipo de *Handler Reference Box (hdlr)*.

En el caso de que la pista sea de audio se realiza la búsqueda de *Chunk Offset Box (stco)*, donde se encuentra el campo *chunk_offset[0]*, que almacena la posición que ocupa el primer byte de la pista de audio en *Media Data Box*. El proceso realizado desde la localización de *Handler Reference Box* hasta este punto, se repite para el número total de pistas de audio contenidas en el archivo IM AF, ya que encontramos en cada fichero IM AF tantas *Track Header Box (trak)* como pistas de audio y texto se encuentren.

Llegados a este punto, se conocen el número de pistas de audio y la posición que ocupan en el archivo IM AF. También se conoce si incluye una pista de texto y su posición en el archivo IM AF. Con esta información, el siguiente paso es la extracción de las pistas de audio y la apertura en Sonic Visualiser, aprovechando la capacidad de Sonic Visualiser de abrir múltiples pistas de audio codificadas en formato MP3.

La Figura 5.1 muestra el proceso de decodificación de las pistas de audio del archivo IM AF.

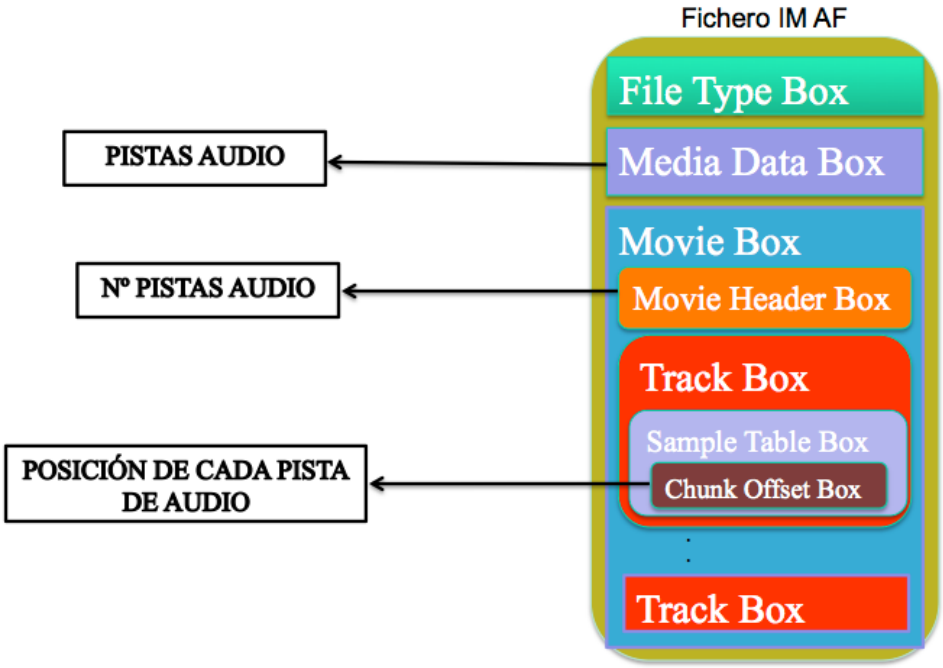

Figura 5.1: Proceso de decodificación de las pistas de audio

5.2. Pista de texto

En el caso de que se incluya una pista de texto en el archivo IM AF, se procede a la búsqueda de *Time To Sample Box (stts)*, de dicha pista de texto. En el campo *num_cadenas* se almacena el número de cadenas de texto que contiene el archivo y en el campo *sample_delta* encontramos el tiempo que permanece cada cadena en pantalla. Un bucle recorrerá la tabla que contiene los instantes de tiempo de cada una de las cadenas, y los almacenará en un array.

En *Sample Size Box (stsz)* se encuentra una tabla que almacena el número de bytes que ocupa cada cadena de texto, incluyendo los modificadores. El campo *num_cadenas* contiene el número de entradas de la tabla, y dicho valor es usado por el bucle que se encarga de leer cada una de las entradas *tam_cadena* de la tabla.

La posición que ocupa el primer byte de la primera cadena de texto en el archivo IM AF se encuentra almacenada en el campo *pos_mem[0]* de *Chunk Offset Box (stco)*.

A continuación, el algoritmo comienza la lectura de las cadenas de caracteres a partir de la posición indicada por *Chunk Offset Box*. Los dos primeros bytes de cada cadena de caracteres indican el número de bytes que ocupa dicha cadena sin incluir los modificadores. Debido a que Sonic Visualiser nos ofrece la opción de personalizar el texto a mostrar en sus paneles, no haremos uso de los modificadores incluidos en el archivo IM AF, dejando al usuario la posibilidad de realizar dicha personalización mediante la interfaz de Sonic Visualiser. Por otro lado, en Sonic Visualiser no es necesario resaltar cada una de las palabras al ritmo de la música, ya que una barra vertical nos va indicando los instantes de tiempo de cada palabra. Por tanto, el algortimo almacena las cadenas de caracteres, pero no los modificadores.

Una vez extraída la cadena de caracteres y el instante de tiempo en el que debe mostrarse por pantalla se hace uso de las opción que nos ofrece Sonic Visualiser para mostrar texto en instantes de tiempo elegidos por el usuario. Para ello se crea un panel con una capa de representación de instantes de tiempo, en la que se van agregando las cadenas de texto en los instantes adecuados. Para que se muestre el texto en los instantes de tiempo correctos hay que realizar algunos cálculos. El instante de tiempo extraído de la tabla se debe dividir por la escala de tiempo, que como vimos anteriormente era "1000". Y dicho resultado se debe multiplicar por "44100", que se corresponde a la frecuencia de muestreo. Ese valor resultante y la cadena se añade al panel creado, y este proceso se repite para cada una de las cadenas de texto almacenadas.

La Figura 5.2 muestra el proceso de decodificación de la pista de texto del archivo IM AF.

5.3. Interfaz Gráfica

Para la apertura del archivo IM AF en Sonic Visualiser, se añade la opción *"Import IMAF File"* en el menú *File* (Figura 5.3). Para ello se crea un objeto del tipo *QAction*, y mediante el método *addAction* se agrega al menú. Cuando se pulsa sobre dicha opción del menú se abre un cuadro de dialogo que permite seleccionar el archivo IM AF a reproducir. El método *getOpenFileName* de la clase *QFileDialog* nos ofrece dicha posibilidad.

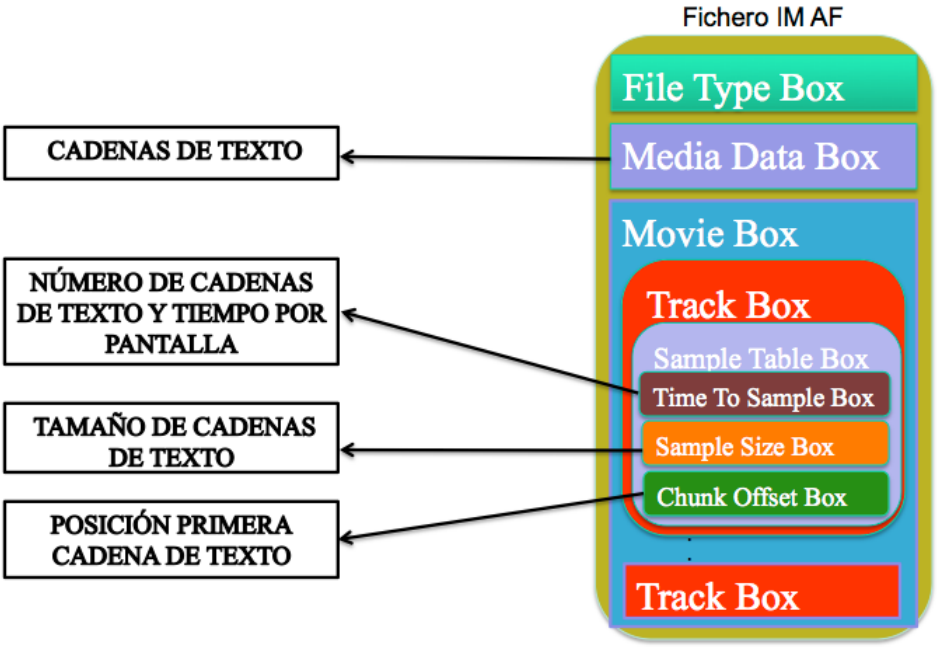

Figura 5.2: Proceso de decodificación de la pista de texto

Una vez invocados los métodos encargados de la decodificación de las pistas de audio y texto se muestran las formas de onda de cada una de las pistas de audio y se crea un panel con una capa del tipo *TimeInstants* que permite mostrar las cadenas de caracteres decodificadas en los instantes de tiempo adecuados.

Con el fin de facilitar el control del nivel de volumen de cada una de las pistas de audio, se ha desarrollado una ventana que incluye un control giratorio de volumen para cada una de las pistas. Dicha ventana aparece automáticamente al abrir un archivo IM AF.

La Figura 5.4 muestra el aspecto de la interfaz gráfica al abrir un archivo IM AF con pistas de audio y texto. Se observan las formas de onda de varias pistas de audio, la letra de la canción, y una ventana con los controles de volumen de las seis pistas que conforman el archivo IM AF.

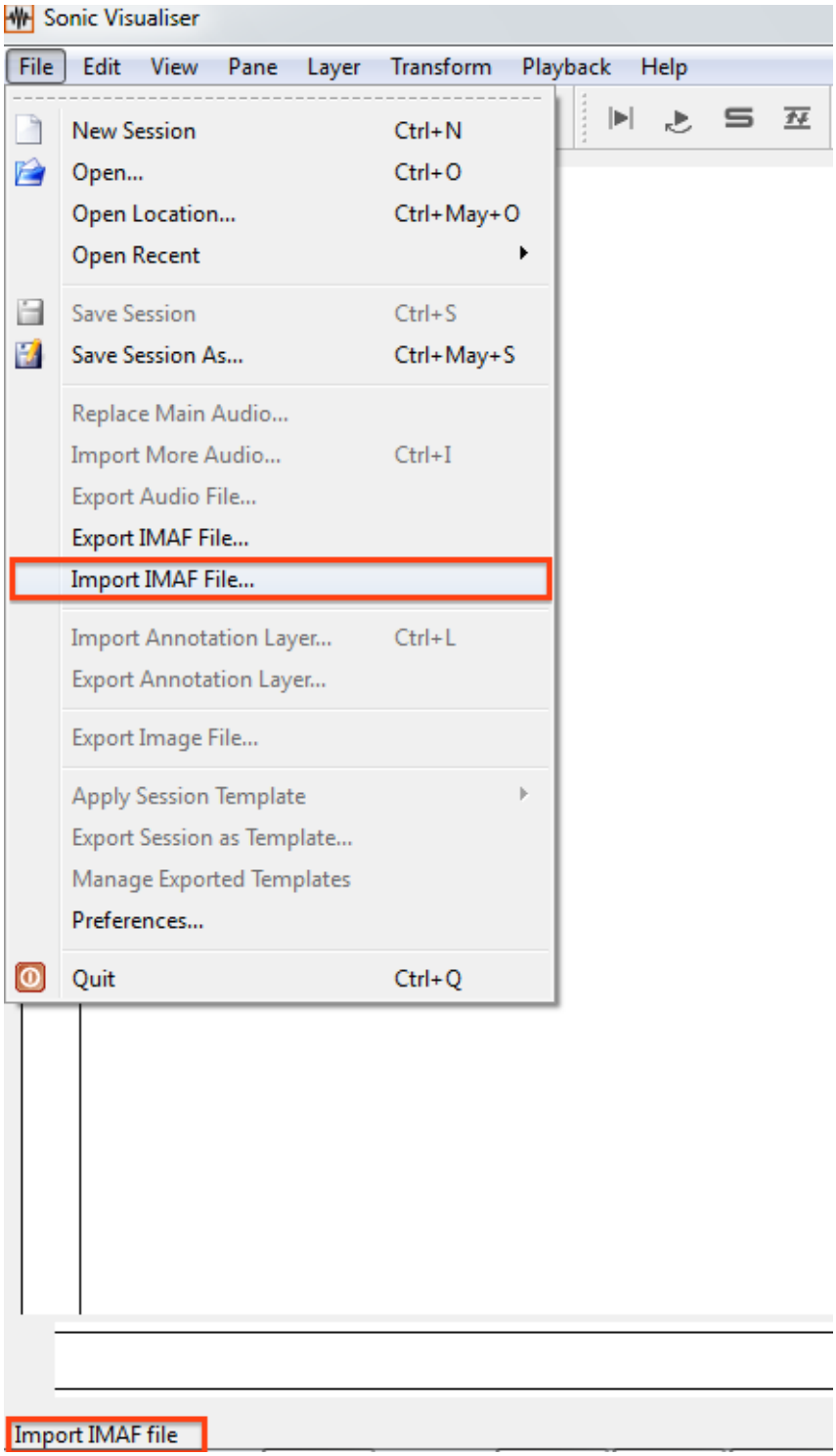

Figura 5.3: Menú que permite la ejecución del decodificador IM AF

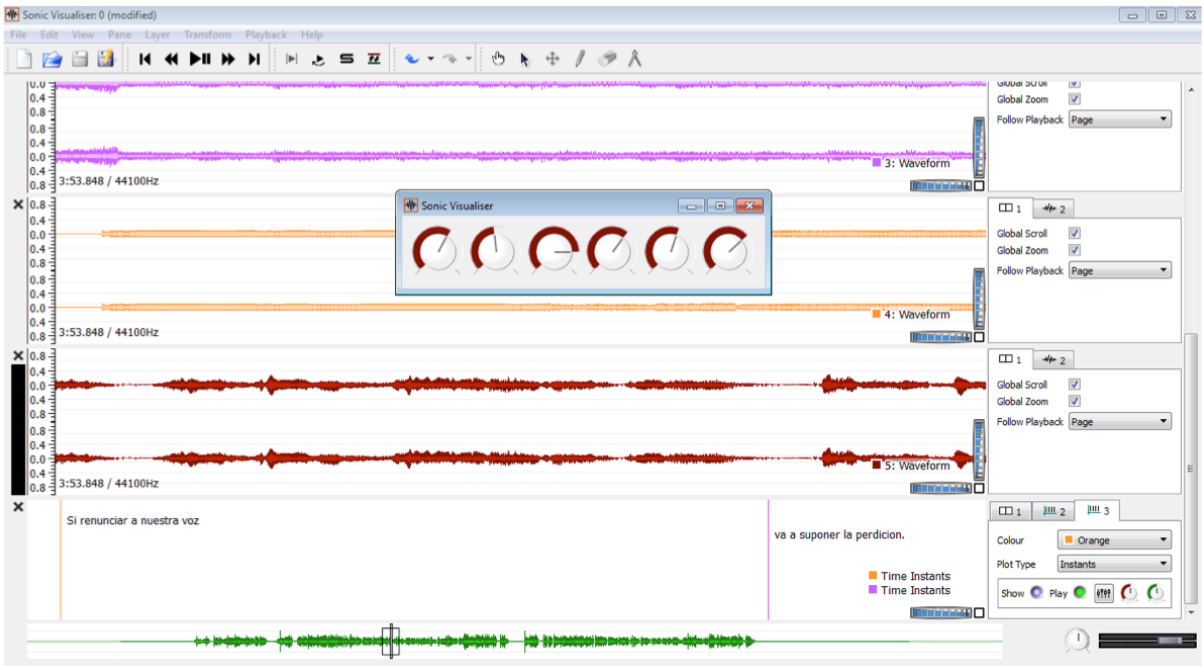

Figura 5.4: Interfaz del decodificador IM AF

Capítulo 6. Pruebas

Con el fin de realizar la verificación del códec IM AF desarrollado, se procede a la creación de cuatro ficheros IM AF con nuestro codificador. El reproductor IM AF proporcionado junto al estándar nos servirá para comprobar que los archivos que genera nuestro codificador se reproducen de la forma esperada, y que por tanto nuestro codificador genera archivos IM AF acordes al estándar.

En este capítulo se expondrá el concepto de puntos de conformidad y su importancia a la hora de comprobar el correcto funcionamiento del códec. Asimismo, se detallarán las características de cada uno de los cuatro ficheros creados y su relación con cada uno de los puntos de conformidad.

Una vez creados los ficheros de prueba, se comprueba su correcto funcionamiento con el software de referencia que se nos aporta junto al estándar. Se mostrarán distintas capturas de pantalla que muestran que el funcionamiento es acorde a los parámetros elegidos en la creación de dichos ficheros IM AF. El decodificador IM AF se verifica con estos ficheros, una vez comprobados que se reproducen correctamente en el reproductor IM AF aportado junto al estándar.

6.1. Puntos de conformidad *(Conformance points)*

Un fichero IM AF que cumple con el estándar debe ajustarse a una serie de puntos de conformidad, tal y como se describe en el estándar. Estos puntos de conformidad son subcategorias que organizan los ficheros por los componentes que soportan. Hay que señalar que el concepto de puntos de conformidad no es el mismo que el concepto de *brand*

(definido anteriormente), ya que en todas las *brands* pueden existir distintos componentes, como metadatos, pista de texto o una imagen. Los puntos de conformidad del formato IM AF son los siguientes :

1. Punto de conformidad 1 (C1) proporciona la capacidad básica de reproducir pistas de audio de acuerdo a un *preset*. Un fichero IM AF que se ajuste a este punto de conformidad debería definir una *brand* en *File Type Box (ftyp)* y contener al menos un *preset*. Además, el fichero debe cumplir las siguientes características :

   - El fichero IM AF debe ajustarse al formato ISO-BMFF
   - Contener uno de los siguientes formatos de audio:
     - MPEG-1 Audio Layer III
     - MPEG-4 Audio AAC Profile
     - MPEG-D SAOC Baseline profile
     - PCM

2. Punto de conformidad 2 (C2) proporciona la capacidad básica de reproducir pistas de audio de acuerdo al modo de usuario (donde se permite la selección de pista/grupo y el control de volumen) que cumple con las reglas definidas. Además de las capacidades del punto de conformidad 1, un fichero IM AF que se ajusta a este punto de conformidad debería contener al menos un grupo y una regla (de selección o de mezcla). Los ficheros que se ajustan a este punto de conformidad debería contener los componentes especificados en el punto de conformidad 1 (C1).

3. Punto de conformidad 3 (C3) proporciona la capacidad para reproducir texto sincronizado con audio, una imagen y metadatos, además de las capacidades definidas en el punto de conformidad 1. Los ficheros que se ajusten a este punto de conformidad debería soportar los siguientes componentes:

   - Los componentes especificados por el punto de conformidad 1 (C1)
   - JPEG

- 3GPP Timed Text
- MPEG-7 Multimedia Description Scheme

6.2. Creación de ficheros de verificación

Para verificar que el codificador es conforme al estándar IM AF, crearemos diversos ficheros IM AF que cumplen los puntos de conformidad anteriormente expuestos.

Para ser exhaustivos, se compararán cuatro ficheros creados, uno para cada uno de los tres puntos de conformidad y otro con todas las posibles características que un fichero IM AF puede incluir.

Los ficheros creados deben ser leídos por el reproductor IM AF proporcionado junto al estándar, para de este modo comprobar el correcto funcionamiento del codificador, ajustándose al estándar.

La Figura 6.1 muestra para cada uno de los cuatro archivos creados los componentes que contienen de acuerdo a cada uno de los puntos de conformidad. Todos los ficheros creados pertenecen a la brand *'im02'*, tal y como se expuso con anterioridad.

	Estructura fichero						Audio	Metadatos	Imagen	Texto
Nº	ISO-BMFF	Brand	'grco'	'prst'	'rusc'	'rumx'	MP3	MDS	JPEG	3GPP TT
1	C1	C1	-	C1	-	-	C1	-	-	-
2	C1	C1	C2	C1	C2	C2	C1	-	-	-
3	C1	C1	-	C1	-	-	C1	C3	C3	C3
4	C1	C1	C2	C1	C2	C2	C1	C3	C3	C3

Figura 6.1: Puntos de conformidad de los ficheros creados

La Figura 6.2 muestra de forma detallada los componentes definidos en cada uno de los cuatro ficheros de conformidad creados.

	Nombre fichero	Brand	Nº pistas audio	Componentes	Presets	Nº grupos	Reglas	
							Selección	Mezcla
1	test1.ima	im02 (im03, im11)	4	MP3	- Preset con pista de volumen estático	-	-	-
2	test2.ima	im02 (im03, im11)	4	MP3	- Preset con pista de volumen estático	1	Regla Min/Max	Regla Inferior
3	test3.ima	im02 (im03, im11)	5	MP3 3GPP JPEG MDS	- Preset con pista de volumen dinámico aproximado	-	-	-
4	test4.ima	im02 (im03, im11)	6	MP3 3GPP JPEG MDS	- Preset con pista de volumen estático - Preset con pista de volumen dinámico aproximado	2	Regla de Implicación	Regla de Límites

Figura 6.2: Detalles de los ficheros creados

El fichero *test1.ima* se ajusta a las características del Punto de conformidad 1. Contiene cuatro pistas de audio en formato MP3 y un *preset* en el que el volumen de las pistas de audio es estático.
El fichero *test2.ima* se ajusta a las características detalladas en el Punto de conformidad 2. Está formado por cuatro pistas de audio, un *preset* en el que el volumen de las pistas permanece estático, un grupo, una regla Min/Max y una regla inferior.
Por otro lado, el fichero *test3.ima* se ajusta a las características del Punto de conformidad 3. Está formado por cinco pistas de audio, una pista de texto con las letras en sincronía con el audio, una imagen con la carátula del disco y metadatos. También contiene un *preset* donde se realiza un efecto *fade-in*.
El último fichero *test4.ima* reúne los tres puntos de conformidad y contiene 6 pistas de audio, una pista de texto con las letras en sincronía con el audio, la carátula del disco, metadatos, dos *presets*, dos grupos y las reglas de implicación y de límites.

6.3. Verificación de los ficheros

Los ficheros IM AF creados se reproducen correctamente en el reproductor IM AF de referencia aportado junto al estándar. Todos los componentes, tales como audio, imagen, texto y metadatos se presentan tal y como se definió en el proceso de codificación.
A continuación se incluirán capturas de pantalla correspondientes a la reproducción de cada uno de los ficheros IM AF creados, poniendo de manifiesto cada una de las características de dichos archivos.

6.3.1. Fichero *test1.ima*

Se compone de cuatro pistas de audio, tal y como se muestra en la Figura 6.3, donde observamos los controles de volumen correspondientes a las cuatro pistas. La información sobre el tipo de codificación del audio y el tipo de *brand* implementada aparece en Figura 6.4. Estos dos parámetros quedarán inalterados en cada uno de nuestros ejemplos, ya que nuestra implementación únicamente permite archivos MP3 para el audio y la *brand*

implementada es la *'im02'*. El único *preset* definido mantiene el volumen de las pistas a un mismo nivel durante la reproducción (ver Figura 6.5).

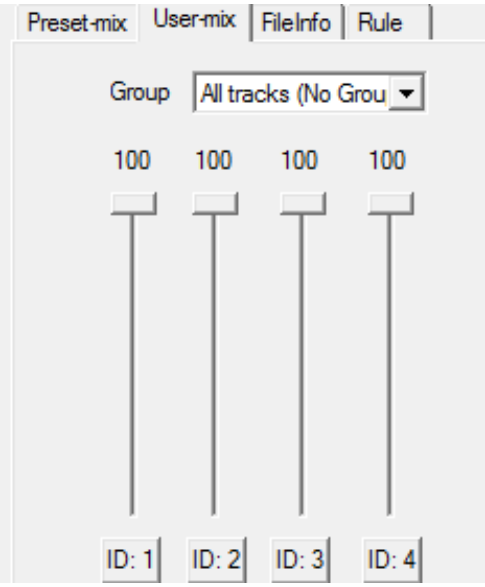

Figura 6.3: Controles de volumen de las cuatro pistas de audio en el modo de usuario (test1.ima)

Figura 6.4: Codificación del audio y *brand* implementada (test1.ima)

6.3.2. *Fichero test2.ima*

Contiene cuatro pistas de audio en formato MP3. Se ha definido un grupo llamado *"Ritmo"* compuesto por la base rítmica de la canción, conteniendo dos pistas de audio (Figura 6.6).

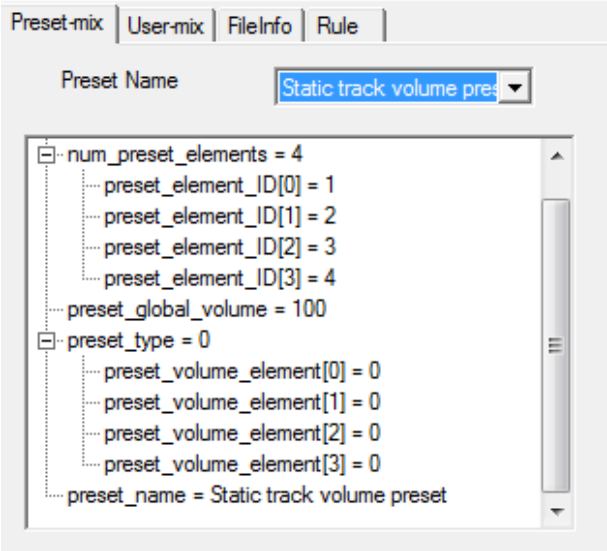

Figura 6.5: *Preset* donde el volumen permanece estático (test1.ima)

Las dos reglas definidas se muestran en la Figura 6.7, donde se puede observar una Regla Min/Max y una Regla inferior (*Lower Rule)*. Por último se define un *preset* donde el volumen no varía (Figura 6.8).

6.3.3. *Fichero test3.ima*

En este caso el archivo cuenta con cinco pistas de audio MP3 (Figura 6.9). Las letras de la canción se iluminan al ritmo de la música y se incluye la carátula del disco (Figura 6.10). La información sobre el intérprete, canción, álbum, etc se incluye dentro de los metadatos (Figura 6.11). El *preset* incluido es de tipo dinámico y realiza un efecto *fade-in* (Figura 6.12).

6.3.4. *Fichero test4.ima*

La última pista incluye todas la características que puede incluir el formato IM AF. Está compuesto por 6 pistas de audio (Figura 6.13), las letras de la canción en sincronía con la música y la carátula del disco (Figura 6.14). Contiene dos *presets*, en el primero de ellos el volumen permanece estático (Figura 6.15) y en el segundo se incluye un efecto *fade-in* (Figura 6.16) . Se definen dos grupos (Figura 6.17) y dos reglas (Figura 6.18), una

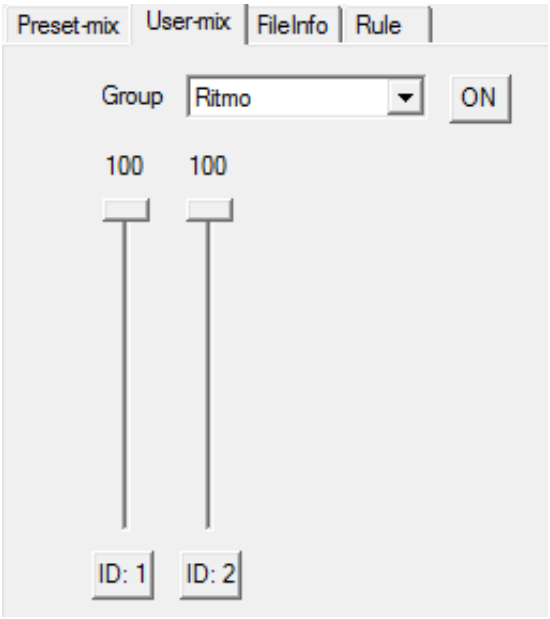

Figura 6.6: Grupo *"Ritmo"* compuesto por la base rítmica de la canción (test2.ima)

Figura 6.7: Reglas Min/Max e Inferior (test2.ima)

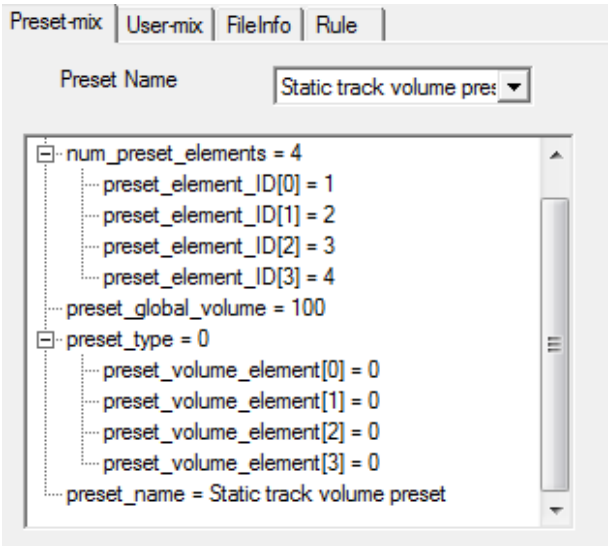

Figura 6.8: *Preset* donde el volumen permanece estático (test2.ima)

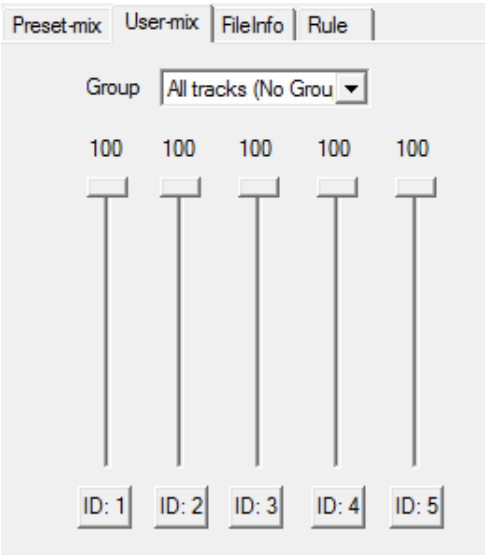

Figura 6.9: Controles de volumen de las cinco pistas de audio en el modo de usuario (test3.ima)

Figura 6.10: Letras de la canción y carátula del disco (test3.ima)

Figura 6.11: Metadatos (test3.ima)

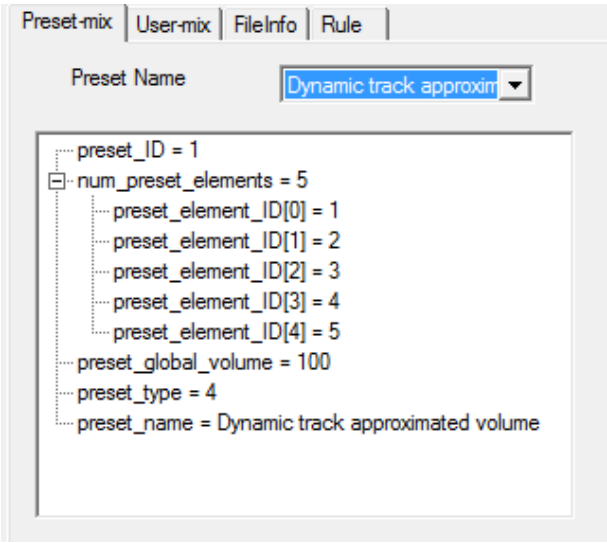

Figura 6.12: *Preset* con volumen dinámico (test3.ima)

regla de implicación y una regla de límites.

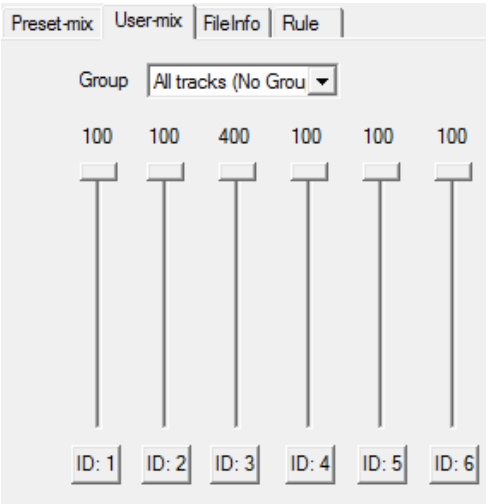

Figura 6.13: Controles de volumen de las seis pistas de audio en el modo de usuario (test4.ima)

Figura 6.14: Letras de la canción y carátula del disco (test4.ima)

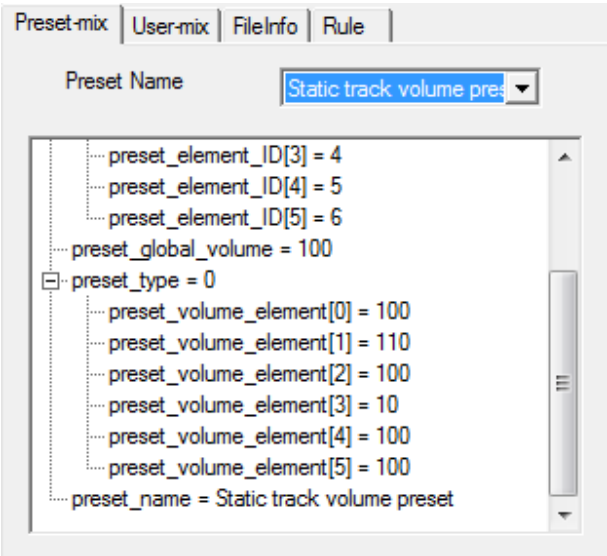

Figura 6.15: *Preset* donde el volumen permanece estático (test4.ima)

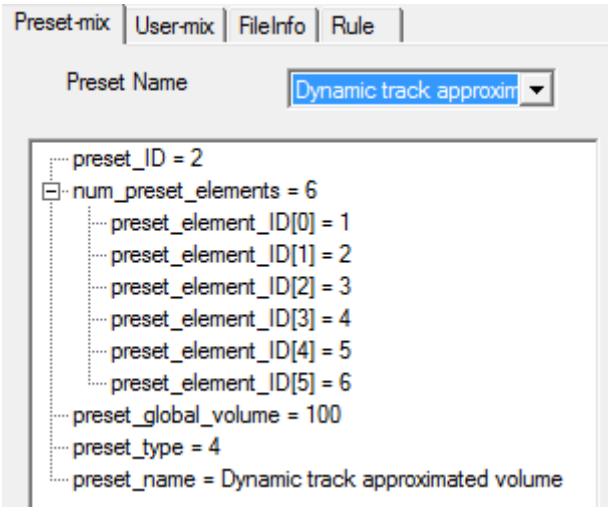

Figura 6.16: *Preset* con volumen dinámico (efecto *fade*) (test4.ima)

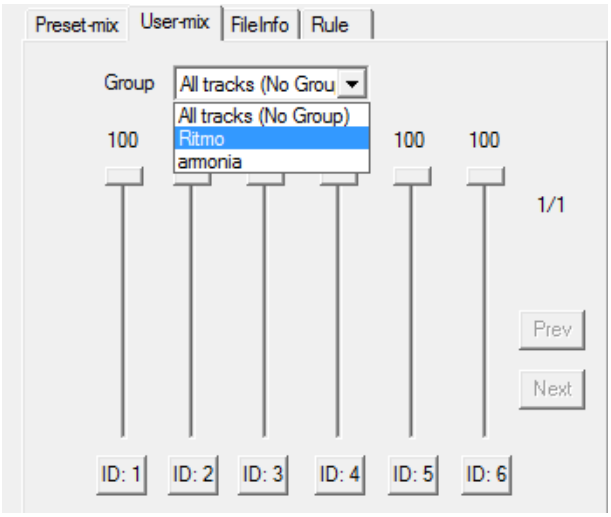

Figura 6.17: Dos grupos definidos : Ritmo y armonía (test4.ima)

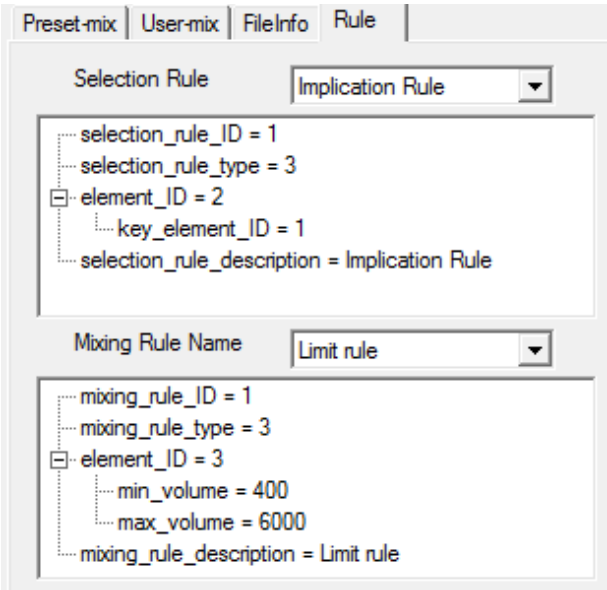

Figura 6.18: Regla de Implicación y Regla de Límites (test4.ima)

6.4. Verificación del decodificador IM AF en Sonic Visualiser

Una vez comprobado el codificador IM AF mediante la creación de cuatro ficheros IM AF y su correcta reproducción en el *software* de referencia proporcionado junto al estándar, procederemos a la verificación del funcionamiento del decodificador implementado en Sonic Visualiser. Para ello, utilizaremos estos cuatro ficheros que ya sabemos que se ajustan al estándar, pues se reproducen sin error alguno en el software de referencia.

Los cuatro ficheros se decodifican correctamente, extrayéndose las pistas de audio, mostrándose las letras en correcta sincronía con la música y ofreciendo al usuario un control más sencillo sobre los volúmenes de cada una de las pistas mediante un panel con controles de volumen. A continuación se incluirán capturas de pantalla de la reproducción de cada uno de los ficheros.

6.4.1. *Fichero test1.ima*

Podemos observar las formas de onda de las cuatro pistas de audio y el panel con los controles de volumen de cada una de las pistas (Figura 6.19).

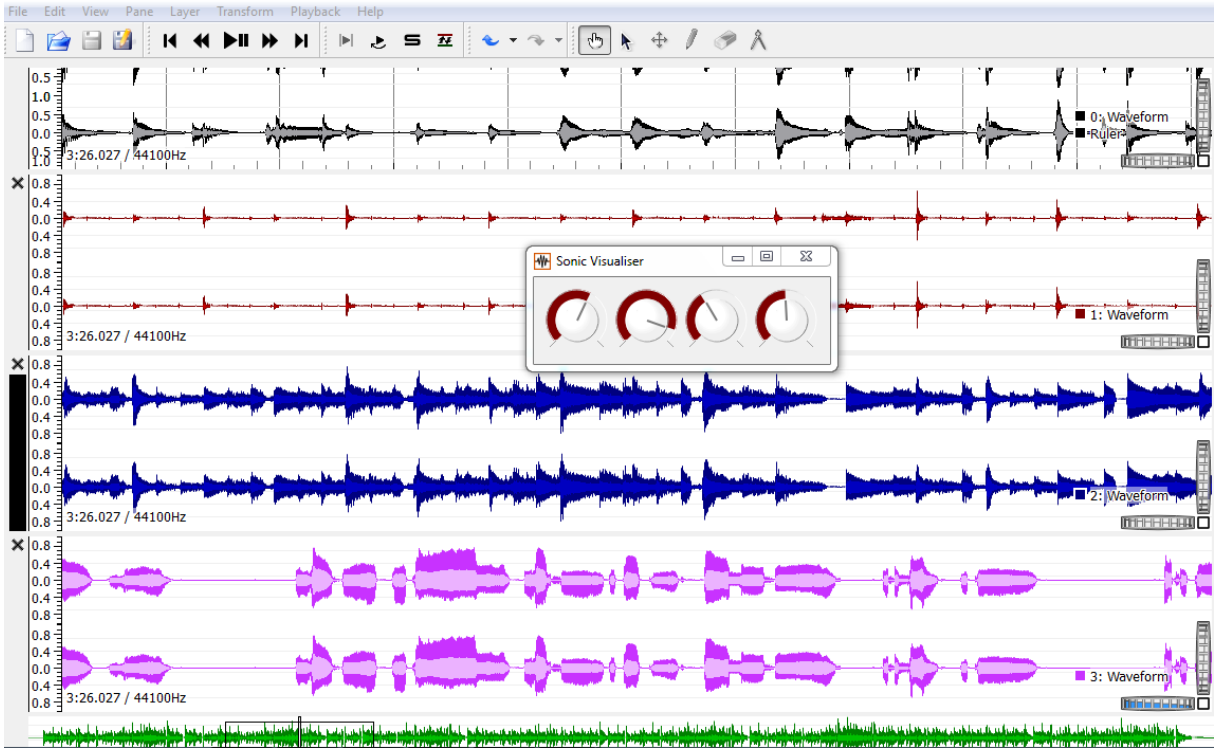

Figura 6.19: Fichero test1.ima

6.4.2. *Fichero test2.ima*

Del mismo modo que en el ejemplo anterior, se observan las formas de onda de las cuatro pistas y el panel de control de volumen de las pistas (Figura 6.20).

6.4.3. *Fichero test3.ima*

Se muestran las formas de onda de las pistas de audio, las letras de la canción y el panel de control de volumen de las cinco pistas de audio (Figura 6.21).

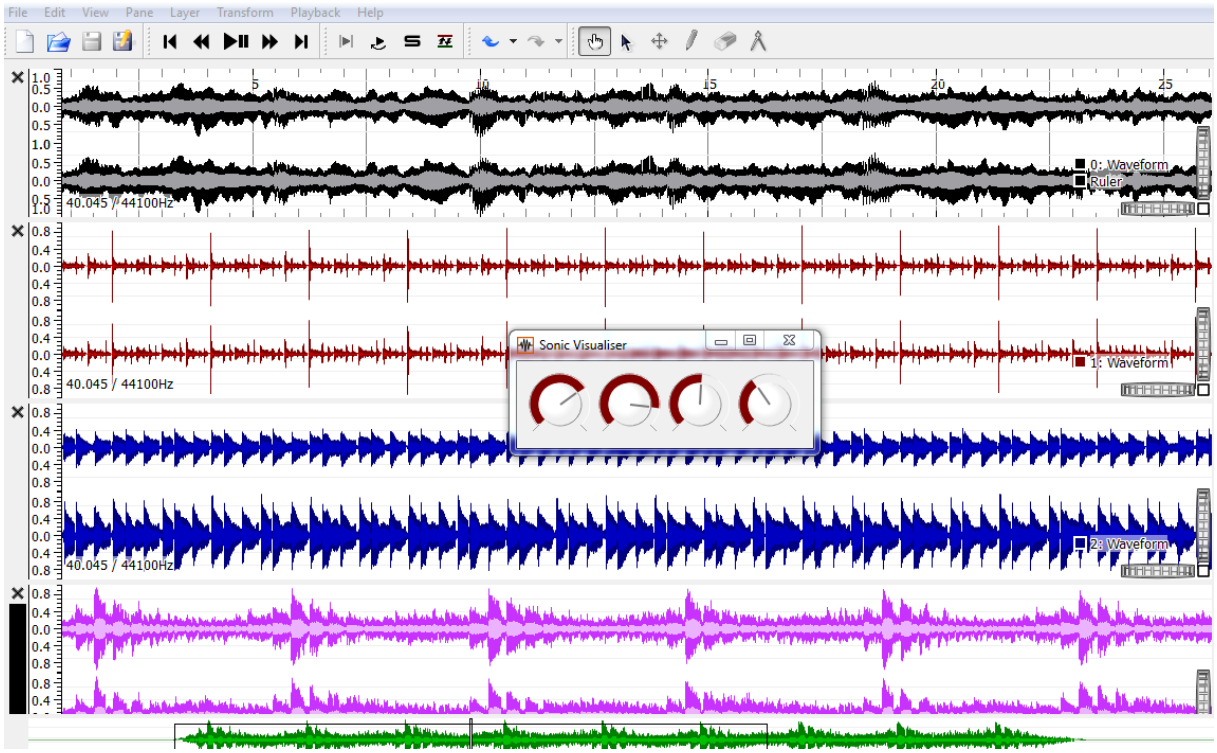

Figura 6.20: Fichero test2.ima

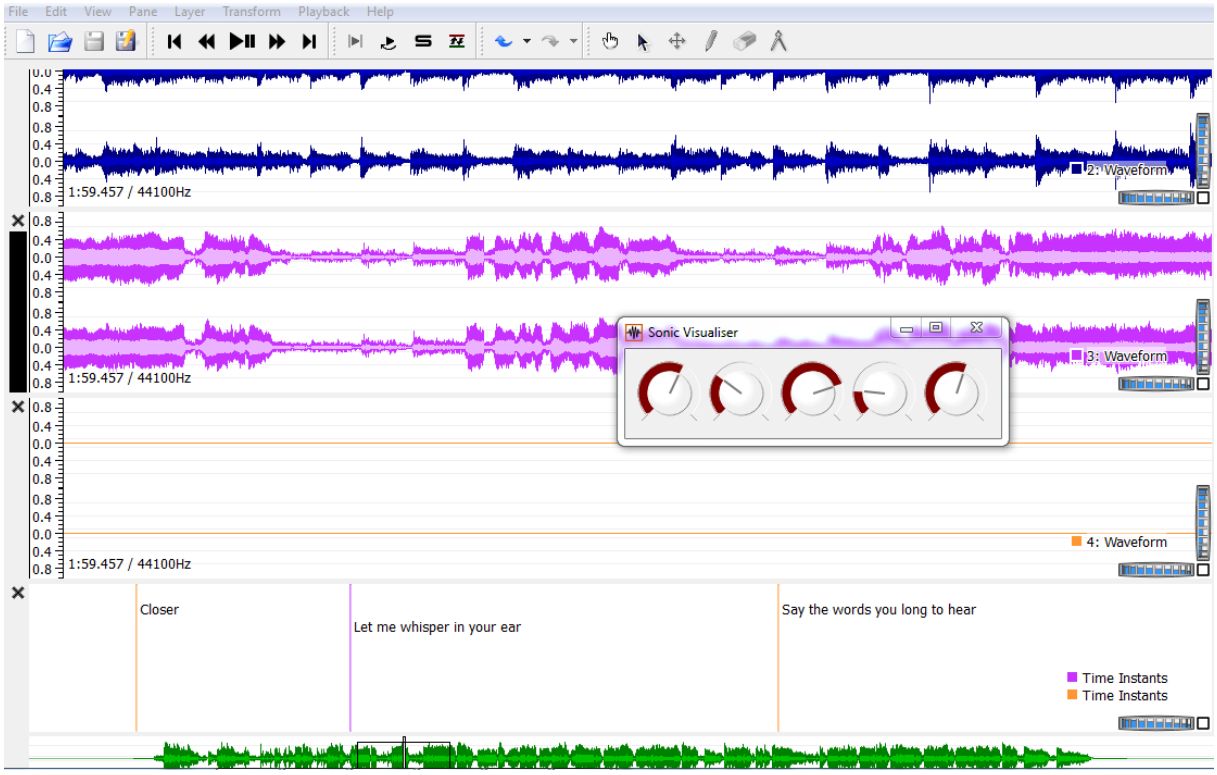

Figura 6.21: Fichero test3.ima

6.4.4. *Fichero test4.ima*

Por último, podemos ver las formas de onda de las pistas de audio, las letras de la canción y el panel de control de volumen de las seis pistas de audio (Figura 6.22).

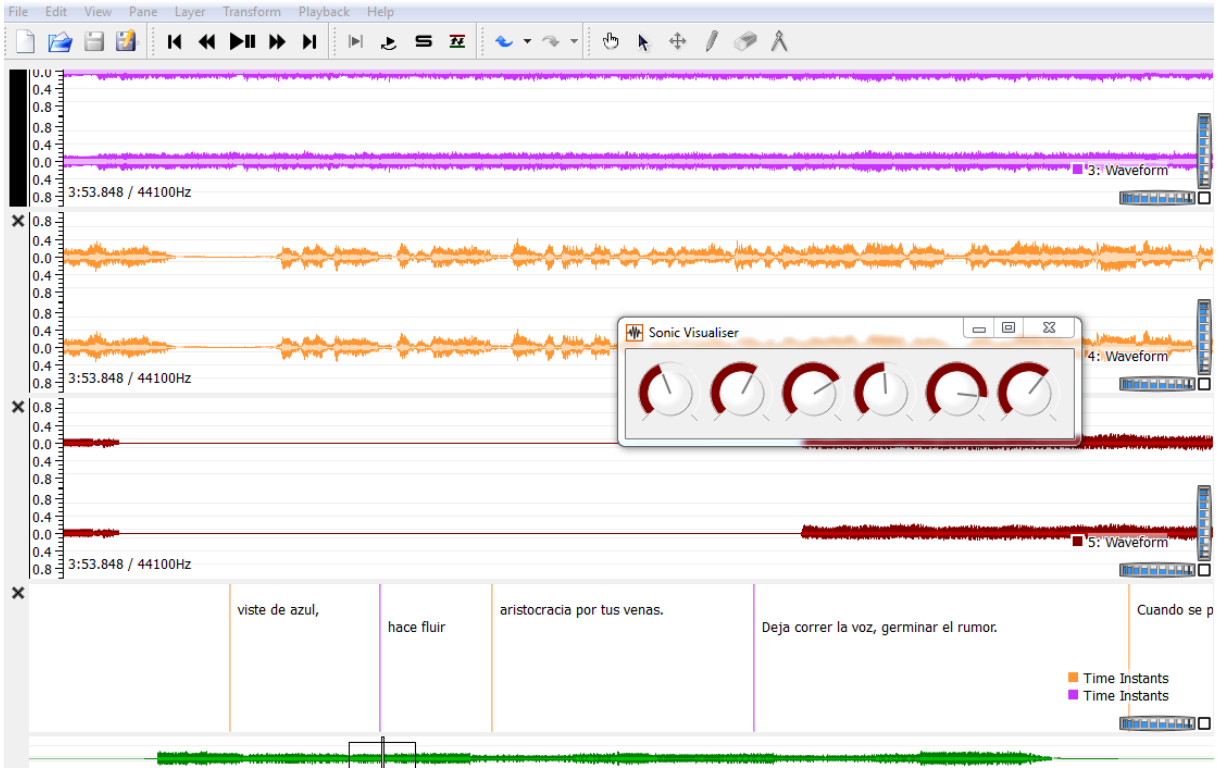

Figura 6.22: Fichero test4.ima

Capítulo 7. Conclusiones y líneas futuras

En este último capítulo se presentan las conclusiones de nuestro proyecto, así como posibles líneas futuras que continúan el trabajo.

7.1. Conclusiones

En este Proyecto Fin de Carrera se ha realizado el diseño e implementación de un códec IM AF y su integración en Sonic Visualiser.
El códec IM AF desarrollado es la primera implementación realizada a día de hoy, y permite la creación de archivos interactivos IM AF con las siguientes características:

- Pistas de audio en formato MP3;
- Texto en sincronía con el audio;
- Imagen, como puede ser la carátula del álbum;
- Reglas, *presets* y grupos;
- Metadatos;

Asimismo, el códec permite la correcta reproducción de las pistas de audio y la pista de texto de los ficheros IM AF generados en nuestro codificador.
La integración de nuestro códec IM AF en Sonic Visualiser presenta múltiples ventajas, entre las cuales podemos citar:

- Permite al usuario la creación de ficheros IM AF de una forma sencilla y fiable, haciendo uso de una interfaz flexible e intuitiva.

- Al importar un fichero IM AF en Sonic Visualiser se muestran de una vez las pistas de audio que conforman una canción. Esto es de gran ayuda para los usuarios de Sonic Visualiser, que de otro modo tendrían que buscar cada pista y abrirla una a una. Es decir, Sonic Visualiser con soporte IM AF puede ser usado por productores musicales como una herramienta de archivado, para almacenar e importar sus proyectos musicales de una vez, consiguiendo una mayor productividad y eficiencia.

- Posibilidad de presentar los acordes de una determinada canción alineados en el tiempo con la letra de la canción. De este modo, un intérprete de guitarra podría ensayar sobre la canción de su banda favorita, silenciando la pista/pistas de guitarra y leyendo al ritmo de la música los acordes a ejecutar y la letra a cantar. Para ello se hace uso de Chordino [16]. Se trata de un *plugin* de Sonic Visualiser que permite la extracción de acordes de una pista de audio, mostrándose por pantalla el nombre del acorde que se está ejecutando en un determinado instante.

  La Figura 7.1 muestra una captura (con un zoom de la región que queremos poner de relieve) del funcionamiento del formato IM AF en conjunción con Chordino. Podemos observar que se presentan los acordes de la canción alineados con la letra de la canción.

- De un modo similar a Chordino, con el *plugin* Melodia [17] es posible la extracción de los tonos de una determinada melodía. Mediante el uso conjunto de nuestro códec IM AF en Sonic Visualiser y el *plugin* Melodia, podemos presentar la letra de la canción correctamente alineada con los tonos de la melodía de un determinado instrumento (o voz).

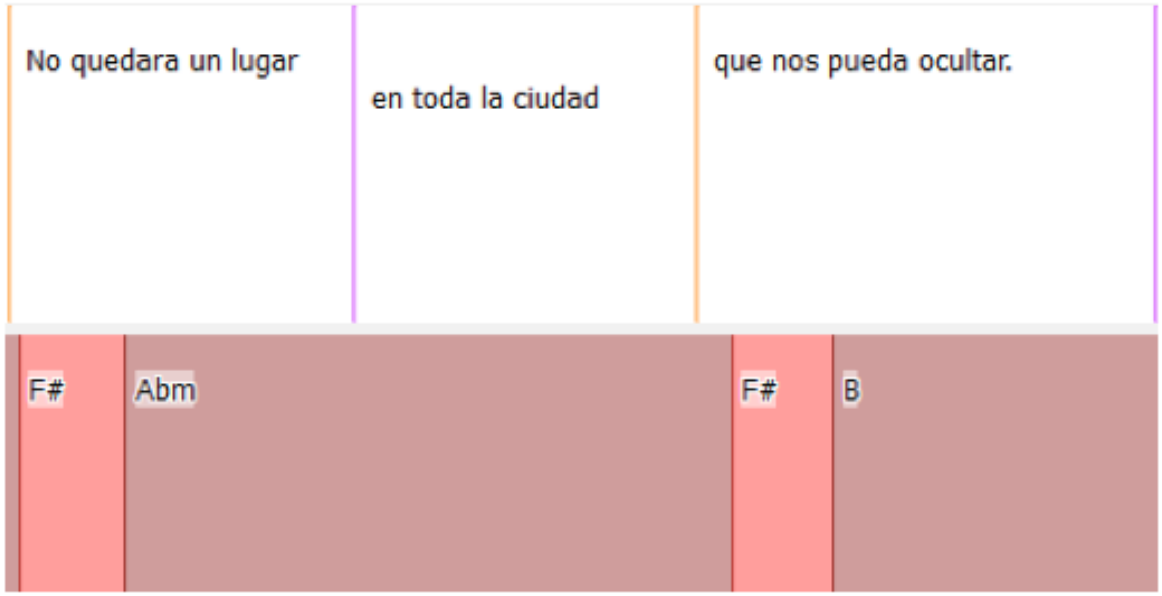

Figura 7.1: Acordes alineados con la letra

7.2. Desarrollos futuros

La integración del formato IM AF en Sonic Visualiser abre nuevas posibilidades para los investigadores del area del procesado de la señal, ofreciendo un banco de pruebas para el desarrollo de nuevos *plugins* para:

- extracción de las letras de la canción a partir de la pista vocal (conversión voz a texto) [18] y extracción de los parámetros temporales de las letras. De este modo, el usuario no tiene que incluir en el archivo IM AF las letras ni tampoco los instantes de tiempo en los que se deben mostrar cada una de las palabras. Especialmente, la anotación de los instantes de tiempo es un proceso bastante tedioso, por lo que un *plug-in* que realice este proceso de forma automática sería de gran interés;

- separación de fuente, para extraer los distintos instrumentos musicales a partir de la mezcla estéreo de la canción [19]. De este modo, sería posible la creación de un fichero interactivo IM AF, a partir de la versión estéreo;

Por otro lado, podría ser de gran interés darle la posibilidad al usuario de almacenar sus propias mezclas, haciendo uso del panel que se ha implementado en Sonic Visualiser para el control de los volúmenes. En ese sentido, el usuario podría almacenar sus propios *presets* y poder compartirlos en redes sociales (por ejemplo, Facebook) con sus amigos. Además, gracias a la flexibilidad del formato IM AF, dicho fichero IM AF no tiene por que contener las pistas de audio, que es el componente que requiere de más espacio. De este modo, las pistas de audio pueden estar almacenadas en una URL (que se indica dentro del fichero IM AF) y así los ficheros que se compartan a través de las redes sociales requerirán un espacio considerablemente menor, lográndose eficiencia en el intercambio de archivos.
Adicionalmente, se está desarrollando un reproductor IM AF en HTML5 [20], que está en fase experimental, y cuyo fin es permitir a los usuarios reproducir archivos IM AF, sin la necesidad de instalar ningún software.

Apéndice A. Artículo publicado

En este apéndice se muestra el artículo presentado en el *AES 53rd International Conference on Semantic Audio*, celebrado en Londres entre los días 27-29 de Enero de 2014. El artículo lleva por título: *Interactive Applications by MPEG-A Support in Sonic Visualiser*[21].

INTERACTIVE MUSIC APPLICATIONS BY MPEG-A SUPPORT IN SONIC VISUALISER

JESUS C. GARCIA [1], COSTANTINO TAGLIALATELA [2], PANOS KUDUMAKIS [3], ISABEL BARBANCHO [1] , LORENZO J. TARDON [1], MARK SANDLER [3]

[1] *Universidad de Málaga , Spain*
jcorral@uma.es, {ibp, lorenzo}@ic.uma.es

[2] *Seconda Universita' Degli Studi Di Napoli, Italy*
c.taglialatela@tiscali.it

[3] *Queen Mary University of London*
{panos.kudumakis,mark.sandler}@eecs.qmul.ac.uk

New interactive music services have emerged, despite currently using proprietary file formats. Having a standardized file format could benefit the interoperability between these services. In this regard, the ISO/IEC Moving Picture Experts Group (MPEG) issued a new standard, the so called, MPEG-A: Interactive Music Application Format (IM AF).
The purpose of this paper is to describe the design and implementation of an IM AF codec and its integration into Sonic Visualiser. In this way, the visualization of the chords or the pitch of the main melody aligned in time with the song's lyrics is achieved. Furthermore, this integration provides the semantic audio research community with a test-bed for further development and comparison of new Sonic Visualiser VAMP plug-ins, e.g., for the conversion of singing voice to text and/or automatic highlighting of lyrics for karaoke applications.

1 INTRODUCTION

The fact that the MP3 file format reigns supreme in the music market is not news. Despite having its market position constantly "attacked" by new formats such as AAC and Ogg Vorbis, nothing has changed so far with respect to user experience [1]. In fact, users continue to play the role of "passive listener" and they are still unable to modify the songs and adapt them to their own taste.
In this context, new interactive music services have emerged with the aim of enhancing the listeners' experience. However, each service uses its own proprietary file format and this has made it difficult to boost a global interactive music market. Thus, a standardized file format is inevitably required to provide the so much needed interoperability between various interactive music players and interactive music songs and albums [2].
This issue is addressed in a recently published standard by ISO/IEC Moving Picture Experts Group (MPEG), known as, MPEG-A: Interactive Music Application Format (IM AF) [3][4]. IM AF allows users to modify the mixing style of a song by changing the volume of each music instrument according to their taste. Other multi-track formats have already been available, e.g., MOGG [5], IEEE 1599 [6] and iXMF [7].

However, IM AF is distinguished by features such as: *timed-text* synchronized with audio tracks, which can represent the lyrics or the chords of a song; *images* related to the song, album and/or artist; *presets* which are predefined versions of a song offered by the music producer (e.g., a karaoke or rhythmic version); and, *rules* constraining the user mixing result to preserve the artistic creation of the composer.
The IM AF specifications come together with a reference software (a non-optimized) player and conformance files. In this way MPEG enables companies to build IM AF compliant encoders and offer associated services and products to market. However, no one has released an open source implementation for an IM AF encoder yet, although commercial services exist [8].
The work described in this paper comes to fill in this gap, proposing an open source implementation of a codec compliant to the IM AF file format [9], with the aim to help the interactive music market blossoming. With respect to the latter, on-line music forums and social networks could be handy: personal mixes of songs could be exported and easily shared between users, having lighter files that contain only information about the mixing parameters while the audio tracks can be made available through various on-line music services. Each audio track could even be replaced by the

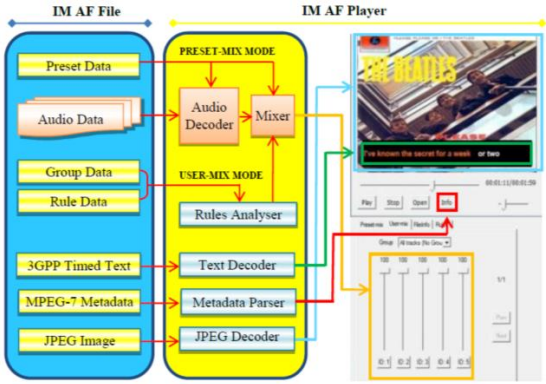

Figure 1. IM AF file format structure and IM AF player architecture with the corresponding media data decoders/parsers.

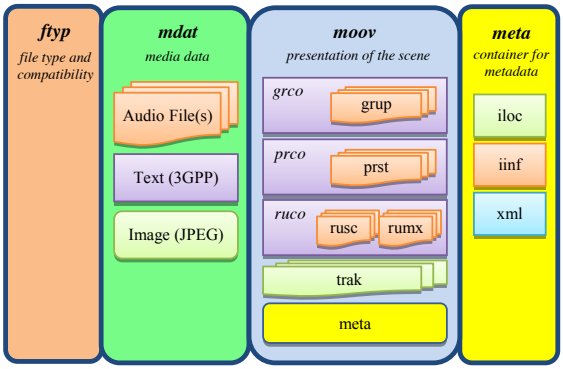

Figure 2. Detailed IM AF file format structure.

users' personal recordings. As a result, it encourages people to develop singing and music instruments playing skills through active learning.

Furthermore, the paper introduces the benefits of Sonic Visualiser with IM AF support. Sonic Visualiser [10] is an open source application designed to assist semantic audio and signal processing researchers to study in a friendly way and take a look at what lies inside an audio file. It has been selected among other music analysis and annotation tools because of its widespread use (>10.000 active users). An essential strength of Sonic Visualiser is its ability to support third-party plug-ins. The native plug-in format is called VAMP, an audio processing plug-in system for plug-ins that extract descriptive information from audio data. In this context, with respect to IM AF, we will make use of VAMP plug-ins specifically designed for audio alignment [11], chords [12] and pitch extraction [13].

In particular, the rest of this paper is structured as follows: the IM AF file format is described in Section 2; in Sections 3 and 4 the implementation aspects of the IM AF player and encoder are discussed, respectively; Section 5 introduces a discussion on how Sonic Visualiser with IM AF support can be used with respect to the aforementioned plug-ins; future developments and conclusions are given in Sections 6 and 7, respectively.

2 FILE STRUCTURE

The framework of the IM AF file is based on the MPEG-4 ISO Based Media File Format (ISO-BMFF) standard [14]; however, IM AF has introduced some improvements to enable interactivity control. The IM AF standard supports compression of the audio tracks in various formats including PCM, MP3, AAC, and SAOC [15]. It also supports the JPEG file format for still pictures [16], the 3GPP timed text for lyrics [17] and the MPEG-7 Multimedia Description Scheme for metadata [17]. Figure 1 shows the architecture of an IM AF interactive music player with the corresponding decoders/parsers for media data.

The IM AF file format consists of a series of boxes that include all relevant data. There are two different types of boxes: general boxes (simply called *Boxes*), which may contain other boxes inside them, and *FullBoxes* which just contain data (no other boxes allowed inside them).

Figure 2 shows how *Boxes* (*ftyp*, *mdat*, *moov*, *meta*) and *FullBoxes* (*grup*, *prst*, *trak*, *etc*.) are located inside an IM AF file [3][4]. All boxes start with a header, which defines their *size* and *type*. The *size* specifies the number of bits that the box will occupy inside the file. The *type* specifies the box identifier (i.e. *ftyp*, *mdat*, *moov and meta)*. Following the same order of boxes as in Figure 2 is not mandatory, however, it is recommended to start with the *File Type Box 'ftyp'*, which identifies the *type* of the file by declaring its *brand compatibility*. The *brand* is related to the maximum number of simultaneously decoded audio tracks and depends on the processing capabilities of the device that will play the file.

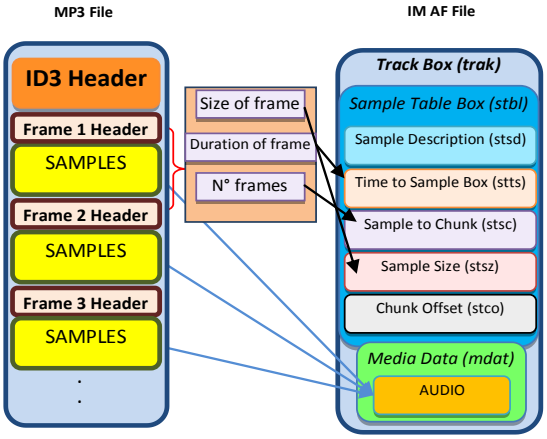

Figure 3. Audio samples and associated data transfer from MP3 to IM AF.

The *Movie Box 'moov'*, describes the scene presentation. This may include one or more *Track Boxes* '*trak*'. Each '*trak*' box contains the description for one type of media (audio or timed-text), while the real media content is stored in the *Media Data Container Box 'mdat'*. This is illustrated in Figure 2. Alternatively, the '*trak*' box may define a URL addressing where the media data are stored. Thus, IM AF files can be very light in terms of storage requirements mainly consisted of metadata.
The *Metadata Box 'meta'* includes metadata, such as, simple background information for a song (title, singer, album, etc.).

3 THE PLAYER

In this section the reference software IM AF player (decoder) and its features are briefly described.

On the player side, users have two different modes of listening a song: *preset-mix mode* and *user-mix mode*. In the *preset-mix mode*, the user could select one of the pre-defined producer mixes. In the *user-mix mode*, the user could select/de-select the tracks and control their volume. All the user's actions are refereed by the *rules analyser* (Figure 1), which decides if the chosen mix is compatible with the rules imposed by the artist (i.e., to avoid complete muting of the guitar track in a rock song).
Independently from the chosen mode, a timed-text track is always available for displaying lyrics, chords or other text information, and so is for pictures in the background.
For legacy players or devices that do not provide multi-track support, IM AF files can still be played. An audio track containing the classic producer's mix of all the instruments in a song can be included in the IM AM file and set as the only "track enabled" that will be played by the legacy player; however, an IM AF player is able to switch it off and play the multi-track version.

4 IMPLEMENTATION OF THE ENCODER

The encoder that creates the IM AF file has been implemented in C programming language. It does not use any external libraries, ensuring that it is cross-platform and will be easily compiled in any computer.
The implementation of a compliant IM AF encoder is mainly based around "filling in" the ISO-BMFF boxes in the right way. This is further explained in the next sections.

4.1 Multi-track Audio

The first step in the encoding process is to extract the audio samples of every single input MP3 audio files and store them into an ISO-BMFF *Media Data Container Box 'mdat'*. This may involve skipping, temporary in this stage, possible MP3 ID3 TAGs containing metadata information related to song, album and artist.
The second step in the encoding process is to extract the following values from each MP3 frame: size (in bytes), duration (in milliseconds) and its number of frames. The IM AF encoder writes this information in the corresponding containers inside a *Track Box 'trak'* as it is illustrated in Figure 3.

The information needed by the encoder in order to reproduce the media data will be contained in tables. Those tables are held in boxes inside the *Sample Table Box* '*stbl*'. The *Sample Description Box* '*stsd*' gives information of the coding type used, the sampling rate and decoder specific information. The *Time to Sample Box* '*stts*' contains a table that stores for each entry two values: the number of consecutive MP3 frames with the same duration and their duration (in milliseconds). Frames are grouped into chunks. The *Sample to Chunk Box* '*stsc*' defines a table that includes the number of frames of each chunk and the number of chunks that have the same number of frames. For constant bitrate MP3s, this table includes only one entry, because all the frames are equally sized. In the *Sample Size Box* '*stsz*' are stored the number of MP3 frames once more and a table giving the size (in bytes) of each of these audio frames. Finally, the *Chunk Offset Box* '*stco*' indicates the position of the first frame of the audio track in the *Media Data Container Box 'mdat'* of the IM AF file.

4.2 3GPP Timed-Text

IM AF supports text visualization in a very customizable way, with possibility to change style and colour for the font and to add visual effects to it. Timed-text data are composed of text samples and sample description. Every sample is a text string (list of characters), which is stored in an external *".txt"* file and can be synchronized with other timed media. Text samples can be followed by sample modifier boxes containing information about how the text string should be rendered such as highlighted text for karaoke. Sample description specifies the way text is rendered, its horizontal and vertical justification, its background and foreground colour, font type and size, etc. The text file contains the strings and their display start/end time. This information is used for dynamic highlighting,(i.e., for lyrics in karaoke style). The syntax is very simple:

[start time] *Text String* **[end time]**

The encoder seeks every text string inside the text file and it copies them in the '*mdat*' box of IM AF file. After a text string is stored, the encoder writes the sample modifier boxes. Values like the highlight colour of the text or the highlight start/end time of words in a phrase are saved at this point. Storage of timing information is similar to audio tracks, as the same boxes are used (*stts*, *stsc*, *stsz* and *stco*, as showed in Figure 3). Duration for every string is saved in '*stts'* box (*sample delta = duration of interval*timescale)*. In '*stsz'* is stored the size of every string, including the modifiers.
Another important box is '*tx3g'*, contained inside *Sample Description Box* ('*stsd'*), which defines sample descriptions for the text track (i.e., font type and size, horizontal and vertical justification, background colour) [17].

4.3 MPEG-7 Metadata

Different 'meta' boxes can be contained at the same time in different levels, as shown in Table 1. Every level hosts textual XML metadata information.

Metadata level	Location
track level	*trak*/*meta* box
song level	*moov*/*meta* box
album level	*meta* box of file

Table 1. Hierarchical levels of metadata in an IM AF.

The encoder receives these data either from the producer or extracts them from the ID3 tags of the input MP3 files, then stores them all in the *XML Container Box* '*xml'*. MPEG-7 is the multimedia content description standard supported by IM AF [17].

4.4 JPEG Still Pictures

Another feature of the IM AF interactive player is that it can display images while a song is played. This is taking place by supporting the JPEG file format for still pictures [16]. Images can be associated, e.g., to music album's cover and/or artist's photos. A picture is included as metadata in the standalone *Metadata Box* '*meta*' in the IM AF file format. Similar to MPEG-7 creation metadata can also be a hierarchy of still pictures at album, song and track level, as shown in Table 1.
From the encoder point of view, the size of the desired JPEG picture is calculated, and then the whole *'.jpeg'* image file is copied in the *Media Data Container Box 'mdat'* box. The image's size and its offset in the *'mdat'* are saved in the *Item Location Box* '*iloc*'; in particular into the *'extent_lenght'* and *'extent_offset'* values. Other information, such as picture's name and encoding type, are stored in *Item Information Box* '*iinf*''.

4.5 Additional boxes

IM AF allows music tracks to be grouped (e.g., all the guitar tracks of a song can consist the *guitar group*). It also provides support for *presets* (e.g., 'karaoke' or 'acoustic' versions of the same song) and *rules* (e.g., not muting completely the guitar solo track in a song). Presets are involved in the player's *preset-mix mode*, while rules are defined to referee user's interaction in *user-mix mode*. Some additional boxes had been introduced to ISO-BMFF structure to define these IM AF's features: *Group Container Box* '*grco*', *Presets Container Box* '*prco*' and *Rules Container Box* '*ruco*'. Each container can host, respectively, one or more *Group Boxes* '*grup*', *Preset Boxes* '*prst*' and *Selection Rule Boxes* '*rusc*' or *Mixing Rule Boxes* '*rumx'*.

Groups

Several audio tracks can be gathered in a group (e.g., all guitars of a song). The groups' container is *Group Container Box 'grco'* box, it hosts one or more *Group Boxes* '*grup*', that is, as many as the groups desired by the producer. This feature allows hierarchical structure of audio tracks in the IM AF file. The elements in a group can be a set of tracks or other groups, allowing several hierarchical levels. Number of tracks per group,

name and number of groups are defined by the producer.

Rules

Rules are restrictions imposed by the producer/artist on the remixing level allowed of a song by a user (e.g., may vocals cannot be muted completely). The main goal of these rules is to preserve the artistic creation. The rule's container may host two types of rules: *selection rules* and *mixing rules*.
The four kinds of selection rules are: *exclusion* rule, *not mute* rule, *implication* rule and *min*/*max* rule. Each one of these essentially defines if a track is in *active* state or not. When an audio track is played its state is *active*, otherwise is *inactive*. The *exclusion rule* specifies which tracks will never be in the active state at the same time. The *not mute rule* defines a track always in the active state. This means that the track will always be played. The *implication rule* specifies that the activation of a track implies the activation of another track. The *min*/*max* rule is used to set the min/max number of active tracks in groups.
The four kinds of mixing rules are: *limits* rule, *equivalence* rule, *upper* rule and *lower* rule applied on elements' volumes. *Mixing rules* shall be considered only for the elements which are in *active* state at rendering time. The *limits rule* sets the minimum and maximum limits of the volume of each track. *Equivalence* and *upper*/*lower* rules are binary rules applied for coupling of audio tracks. The *equivalence rule* applied to two different tracks means that these tracks are always played at the same volume. Therefore, if the volume of one of these tracks is changed, the volume of the other track will also be changed by the same level. Volumes affected by *upper*/*lower rules* shall respect a non-strict superiority/inferiority relationship.

Presets

The IM AF standard defines some default presets. Every preset contains information related to the volume of each track in a song (e.g., a *default preset* could be the original mix by the music producer or a *karaoke preset* with the vocal track muted). The *Preset Container Box* '*prco*' stores one or more *Preset Boxes* '*prst*', that is, as many as the available default preset versions of the mixed-down song.
Presets form two main categories: *static* and *dynamic*. Using *static presets*, the encoder sets a fixed volume to each element involved for the whole duration of the song. In *dynamic presets* the volume of a track (or various tracks, simultaneously) can vary over time [4]. This functionality is useful to create effects like fade-in or fade out.
Every available preset can be equalized using standard filters implemented into the IM AF player, such as: Low and High Pass Filter, Low and High Shelf Filter and/or Peaking Filter. More than one filter can be applied on each track at the same time.

5 RESULTS AND DISCUSSION

Compliance of the developed encoder to the IM AF standard is achieved by creating IM AF files -- similar to conformance files provided by the standard -- which in turn are parsed and played successfully using the reference software player.

As an example, an IM AF created file could have the generic features as those shown in Table 2.

Example1.ima	
Format of tracks	MP3
Number of tracks	6
Groups	1
Presets	2
Rules	2
Still pictures	Yes (album image)
Timed-text track	Yes (lyrics)

Table 2. Generic features of an IM AF created file for testing the encoder compliance to the standard.

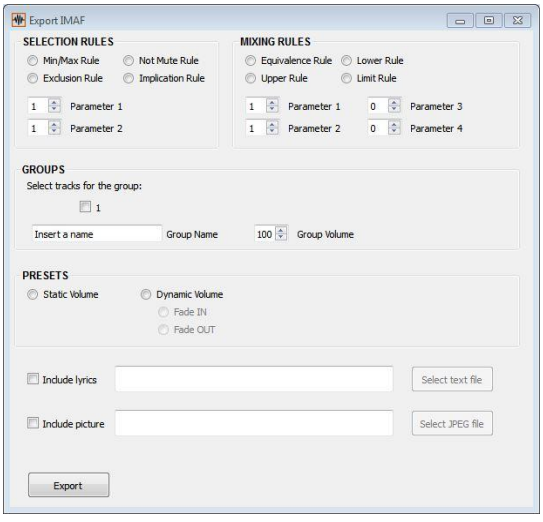

Figure 4. The user interface of the IM AF encoder in Sonic Visualiser.

Combinations of features can be made in creating IM AF files. This example is just a representation of the possibilities of the developed IM AF encoder.

The developed IM AF codec has been fully integrated into Sonic Visualiser. The source code of Sonic Visualiser with IM AF support is available through the SoundSoftware.ac.uk repository [9]. In the latter URL [9], an IM AF (.ima) song is also available for testing and experimenting with the IM AF codec. This song includes several audio tracks, timed text lyrics and the cover of the album. The lyrics have been annotated manually using the Sonic Visualiser itself.

Figure 4 shows the user interface of the IM AF encoder in Sonic Visualiser. This interface has been implemented using the Qt library [19]. Beyond enhancing the user's experience, using IM AF multi-track file format integrated into a tool for the analysis and annotation of audio data, such as Sonic Visualiser, opens up new opportunities for semantic audio researchers.

A possible use case scenario is presented in the following using a Sonic Visualiser plug-in for chords extraction, known as Chordino [12]. Figure 5 shows the difference on using the same Chordino algorithm, with the same set-up, between the single instrument tracks and the classic mix-down of a song. Processing of the mixed track cannot be precise enough for having a reliable chords extraction, due to the overlap of several instruments. A more accurate extraction is achieved by applying the Chordino plug-in to each single instrument track available in an IM AF file (e.g., on the guitar track only, for extracting the guitar chords). Furthermore, the chords extracted are now presented aligned in time with the song's lyrics, as shown in Figure 6. The latter consists the novelty of this paper and becomes feasible thanks to IM AF support in Sonic Visualiser.

The same discussion as the one about chords extraction could be done for the extraction of the melody pitch of a song. Using the VAMP plug-in called Melodia, an algorithm for melody pitch estimation [13], it is possible to have a more efficient pitch extraction from only the vocal track, rather than the mixed song. Furthermore, the melody pitch is now presented aligned in time with the song's lyrics. Extracting the lyrics from the vocal track (singing to text) could be another possible scenario but there is not yet such a VAMP plug-in implemented; however, IM AF support in Sonic Visualiser could encourage such developments since it provides a test-bed for such developments and/or comparisons among different algorithms (e.g., singing voice to text [20], source separation for music instruments extraction from an MP3 [21]).

Another use case scenario of IM AF in Sonic Visualiser involves a VAMP plug-in called MATCH [11], an algorithm for audio alignment between two renditions of the same piece of music. IM AF allows users to add their own musical instrument/vocal track in a song by replacing the existing ones; alignment between the tracks, in such a scenario, is needed for having a proper resulting mix, ensuring that all the instruments start at the same point and follow the same beat. A tool like MATCH can carry out this task easily. However, Sonic Visualiser can run the MATCH plug-in appropriately for calculating automatic alignments between two tracks, it does not do so for multiple audio tracks. The latter is a further demonstration on the opportunities opened up for the semantic audio research community using Sonic Visualiser with IM AF support, as a test-bed not only for the development of new but also for further improvement of existing VAMP plug-ins.

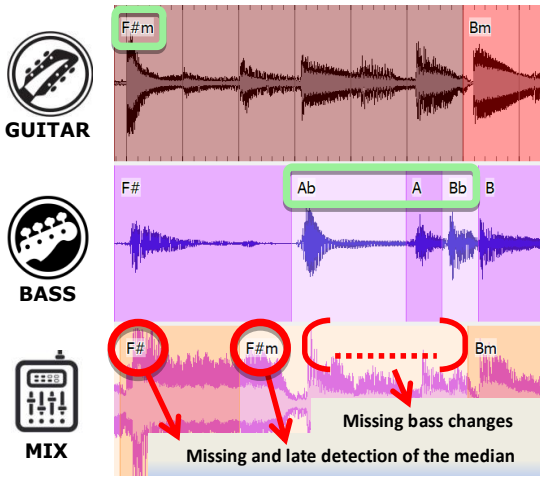

Figure 5. Usage of Chordino in Sonic Visualiser for chords extraction: a) from individual music instruments (guitar and bass) and, b) from the mixed-down track (producer mix/preset).

6 FUTURE DEVELOPMENTS

This innovative file format is expected to enable interoperable commercial interactive music services and collaborative music creation communities to blossom on Internet, due to the flexible IM AF's features inherited from ISO-BMFF. That is, different contents can be stored in different places (e.g., groups, presets and rules are stored into the file; audio tracks, text and pictures are stored on a server linked by URLs), enabling consequently efficient exchange and sharing of IM AF

files in social networks. Actually, we are working in this direction by enabling users to record each tracks' volume while disc-jockeying (DJ'ing). In this way users could save their own mixes and share them with their friends using social networks (e.g., Facebook). Furthermore, an alternative cross-platform and cross-browser HTML5 IM AF player [22] has also been developed for users that would only like to listen to their friends' mixes with no need to download Sonic Visualiser.

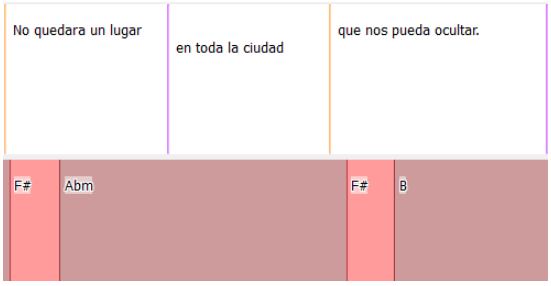

Figure 6. Chords aligned in time with the song's [9] lyrics in Sonic Visualiser.

7 CONCLUSIONS

This paper describes the design and implementation of an IM AF codec and its integration into Sonic Visualiser. This integration provides the semantic audio research community with a test-bed for further development, testing and performance comparisons of newly developed VAMP plug-ins. Some use cases have been described using existing VAMP plug-ins for chord and melody extraction, as well as, music instrument alignment. Based on its features, IM AF has all the potential to be the new music industry file format.

8 ACKNOWLEDGMENTS

Costantino Taglialatela acknowledges that this work has been done during his visit at Queen Mary University of London. Panos Kudumakis acknowledges that this work has been partially done during his visit at the University of Malaga in the context of the program Andalucía TECH: Campus of International Excellence and in conjunction to UK EPSRC project EP/H043101/1 SoundSoftware.ac.uk

9 REFERENCES

[1] P. Kudumakis, "MP3: something's gotta change!", Audio!, pp. 6, Vol. 1, Issue 3, April 2011. Editors: P. Curzon, M. Barthet and S. Dixon.

[2] I. Jang, P. Kudumakis, M. Sandler, K. Kang: "The MPEG Interactive Music Application Format Standard", *IEEE Signal Processing Magazine*, pp. 150-154, Vol. 28, Issue 1, Jan. 2011.

[3] ISO/IEC 23000-12:2010 - Information Technology - Multimedia Application Format (MPEG-A) -- Part 12: *Interactive Music Application Format*.

[4] ISO/IEC 23000-12:2010/Amd.2:2012 - Information technology - Multimedia application format (MPEG-A) - Part 12: *Interactive music application format*, AMENDMENT 2: *Compact representation of dynamic volume change and audio equalization*.

[5] MOGG files, Multitrack Digital Audio Format, (Online), Available: http://moggfiles.wordpress.com (last accessed 01.12.13).

[6] L. A. Ludovico: "Key concepts of the IEEE 1599 Standard", *Proceedings of the IEEE CS Conference The Use of Symbols To Represent Music And Multimedia Objects*, pp. 15-26, 2008.

[7] C. Grigg: "Preview: Interactive XMF - A Standardized Interchange File Format for Advanced Interactive Audio Content", *115th Audio Engineering Society Convention*, October 2003

[8] iKlax Media, (Online), Available: http://www.iklaxmusic.com (last accessed 01.12.13).

[9] Panos Kudumakis, "MPEG developments" (Online), Available: https://code.soundsoftware.ac.uk/projects/mpegdevelopments (last accessed 01.12.13).

[10] C. Cannam, C. Landone and M. Sandler: "Sonic Visualiser: An Open Source Application for Viewing, Analysing, and Annotating Music Audio Files", *Proceedings of the ACM Multimedia 2010 International Conference*.

[11] S. Dixon and G. Widmer: "MATCH: a music alignment tool chest", *Proceedings of the International Symposium on Music Information Retrieval*, pp. 492-497, 2005.

[12] M. Mauch and S. Dixon: "Approximate Note Transcription for the Improved Identification of Difficult Chords", *Proceedings of the International Symposium on Music Information Retrieval*, pp. 135-140, 2010.

[13] J. Salamon and E. Gómez: "Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics", *IEEE Transactions on Audio, Speech and Language Processing*, 20(6):1759-1770, Aug. 2012.

[14] ISO/IEC 14496-12:2008 - Information technology - Coding of Audio-Visual Objects – Part 12: *ISO base media file format*.

[15] ISO/IEC 23003-2:2010 - Information technology - MPEG audio technologies - Part 2: *Spatial Audio Object Coding (SAOC)*.

[16] ISO/IEC 10918-1:1994 - Information technology - *Digital compression and coding of continuous-tone still images (JPEG)*.

[17] ETS 3GPP TS 26.245-2004 - Transparent end-to-end Packet switched Streaming Service (PSS); *Timed text format*.

[18] ISO/IEC 15938-5:2003 - Information technology - Multimedia content description interface – Part 5: *Multimedia description schemes*.

[19] Qt Library, (Online), Available: http://qt.digia.com/ (last accessed 01.12.13).

[20] T. Hosoya, M. Suzuki, A. Ito and S. Makino: "Lyrics recognition from a singing voice based on finite state automation for music information retrieval", *Proceedings of the International Symposium on Music Information Retrieval*, pp. 532-535, 2005.

[21] J. Han, Z. Rafii and B. Pardo, "Audio Source Separation" and "REPEAT", *Research projects of Northwestern University*, Dep. of Elec. Eng. and Comp. Sc., (Online), Available: http://music.cs.northwestern.edu (last accessed 01.12.13).

[22] G. Herrero, P. Kudumakis, L. J. Tardon, I. Barbancho and M. Sandler, "An HTML5 Interactive (MPEG-A IM AF) Music Player", *10th International Symposium on Computer Music Multidisciplinary Research (CMMR)*, Marseille, France, 15-18 Oct. 2013.

Apéndice B. Capítulo publicado

En este apéndice se presenta el capítulo elaborado para el libro *Springer Handbook for Systematic Musicology*. Dicho capítulo lleva por título *Enabling Interactive and Interoperable Semantic Music Applications* [22].

Chapter 1

Enabling Interactive and Interoperable Semantic Music Applications

JESUS C. GARCIA[1], PANOS KUDUMAKIS[2], ISABEL BARBANCHO[1], LORENZO J. TARDON[1], MARK SANDLER[2]

[1] *Universidad de Malaga , Spain*
jcorral@uma.es,{ibp, lorenzo}@ic.uma.es

[2] *Queen Mary University of London*
{panos.kudumakis,mark.sandler}@eecs.qmul.ac.uk

1.1 Introduction

The music industry has been experiencing remarkable changes in the past two decades, especially since the popularization of the MP3 standard and the use of Internet by a large part of the population. However, such changes are mostly related to more efficient distribution of music content rather than the way users are interacting with the music itself. Despite the widespread use

of touchscreen smart phones and tablets which are enabling interactivity, users of formats like MP3, FLAC or AAC can still only interact with the music they listen to a limited extent. In this sense, interactive music services are emerging to enhance the listener's experience [1][2]. In recent years there have been various interactive services marketed as iKlax [3], MOGG [4] or MT9 [5].

However, such interactive services use their own proprietary formats. In order to ensure interoperability between different interactive music players and interactive songs, a standardized file format is required.

This shortcoming has been overcame by the ISO / IEC Moving Picture Experts Group (MPEG) which has recently released a standard, known as, MPEG-A: Interactive Music Application Format (IM AF) [6][7]. The latter is a multi-track format, so it contains individual audio tracks for different musical instruments. It allows the user to change the mix of the song, by changing the volume of each instrument separately.

The novelty of this format over other multitrack formats lies in adding multimedia content to enrich the user experience. So it is possible to include text synchronized with audio, which represents the lyrics or the chords of the song.

It may also contain images related to the album or the song. The producer has the ability to define rules to prevent the song becoming unrecognizable. In addition different presets of the song (e.g., a karaoke or rhythmic version) can also be defined.

Figure 1.1 shows a comparison of the characteristics of iKlax, MT9, MOGG and IM AF formats.

This chapter introduces the benefits of adding IM AF support to Sonic Visualiser [8]. Sonic Visualiser [9] is an open source application designed to assist semantic audio and signal processing researchers to take a look at what lies inside an audio file. It has been selected among other music analysis and annotation tools because of its widespread use (10.000 active users in December 2013). An essential strength of Sonic Visualiser is its ability to support third-party plug-ins. The

	iKlax	MT9	MOGG	IM AF
Multitrack	✔	✔	✔	✔
Pictures	✖	✔	✖	✔
Text	✖	✔	✖	✔
Presets	✔	✖	✖	✔
Rules	✔	✖	✖	✔
Groups	✔	✖	✖	✔

Figure 1.1: Comparison between different multitrack formats

native plug-in format is called VAMP, which enables developers to implement algorithms for extracting descriptive information from audio data. In this context, with respect to IM AF, we will make use of VAMP plug-ins specifically designed for chords [10] and melody pitch [11].

This chapter is structured as follows: the IM AF standard is described in Section 1.2; in Section 1.3 the implementation aspects of the IM AF encoder are discussed; Section 1.4 introduces a discussion on how Sonic Visualiser with IM AF support can be used with respect to the aforementioned plug-ins; future developments and conclusions are given in Section 1.5.

1.2 IM AF standard

An IM AF file consists of:

- Multiple audio tracks that represent the different parts of a song (instruments and/or voices);
- groups of audio tracks that define a hierarchical structure of audio tracks (e.g. all the guitars of a song can be gathered in the same group);
- presets which contain predefined information about the mixing of multiple audio tracks (for example, a karaoke version of the song or a version with only the rhythmic base);

Type	Component Name	Abbreviation	Specification
File Format	ISO Base Media File Format	ISO-BMFF	ISO/IEC 14496-12:2008
Audio	MPEG-1 Audio Layer III	MP3	ISO/IEC 11172-3:1993
	MPEG-4 Audio AAC profile	AAC	ISO/IEC 14496-3:2005
	MPEG-D SAOC Baseline profile	SAOC	ISO/IEC 2003-2:200x
	PCM	PCM	-
Image	JPEG Image	JPEG	ISO/IEC 10918-1
Text	3GPP Timed Text	3GPP TT	3GPP TS 26.245
Metadata	MPEG-7 Multimedia Description Scheme	MDS	ISO/IEC 15938-5:2003

Figure 1.2: Supported components in IM AF

- rules that introduce specific data related to the interaction with the user (definition of allowable actions concerning the audio tracks, group selection and volume control);
- additional multimedia information to enrich the user interaction (text synchronized with audio to represent the lyrics of a song, images related to the song, album or artist);
- metadata to describe the song, album or artist;

The IM AF standard supports compression of the audio tracks in various formats including PCM, MP3, AAC, and SAOC [12]. It also supports the JPEG file format for still pictures [13], the 3GPP timed text for lyrics [14] and the MPEG-7 Multimedia Description Scheme for metadata [15]. The framework of the IM AF file is based on the MPEG-4 ISO Based Media File Format (ISO-BMFF) standard [16].

Figure 1.2 shows all the supported components in IM AF format.

IM AF files consist of a set of objects, called boxes (Figure 1.3), which contain all data in the file. We can classify them into two types: general boxes (which may contain other boxes inside

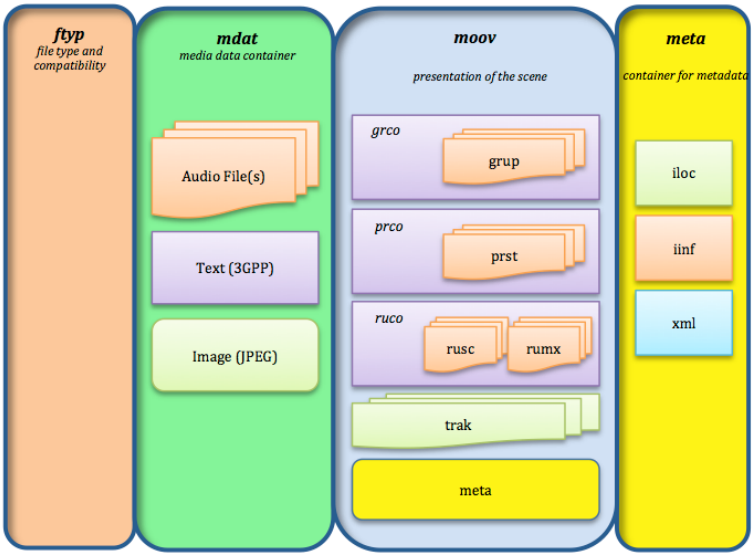

Figure 1.3: IM AF file format structure

them), and FullBoxes which just contain data (no other boxes allowed inside them).

The File Type Box 'ftyp' box should appear at the beginning of the file, as it defines the type of file, declaring their brand compatibility (see Figure 1.4.). The brand is related to the maximum number of audio tracks that can be decoded simultaneously depending on the processing capabilities of the device that will play the IM AF file.

Brands starting with 'im0' are intended for mobile phones and portable audio players, while brands starting with 'im1' are intended for general music services. 'im21' is the brand for the high-end application for professional user.

The multimedia content (audio, text, images) is stored within the Media Data Box 'mdat'.

The information needed for decoding the multimedia content is stored in the Movie Box 'moov'. This box can contain one or more Track Boxes 'trak'. Each Track Box contains the description of one type of media (audio or text) and it may include a URL indicating where the media con-

	Audio				Max. # of simultaneously decoded audio tracks	Max. sampling frequency/bits
Brands	AAC	MP3	SAOC	PCM		
'im01'	•	•			4	48kHz/ 16bits
'im02'	•	•			6	
'im03'	•	•			8	
'im04'	•	•	•		2	
'im11'	•	•		•	16	
'im12'	•	•	•		2	
'im21'	•			•	32	96kHz/ 24bits

Figure 1.4: IM AF brands

tent is stored. Thus, IM AF files can be very light in terms of storage requirements. The information on the groups, presets and rules are stored in Group Container Box 'grco', Preset Container Box 'prco' and Rule Container Box 'ruco', respectively.

The Metadata Box 'meta' includes metadata, such as, simple background information for a song (title, singer, album, etc.).

1.3 Implementation of the IM AF encoder

The encoder is responsible for creating the IM AF files. It has been implemented in C programming language and does not use external libraries. In this way, we can ensure that it can be compiled on any platform. The following sections explain the implementation of the encoder. For further details see [17]. Each section addresses a different functionality of the encoder.

1.3.1 Audio tracks

The File Type Box 'ftyp' appears at the beginning of the file and defines the characteristics of the audio tracks contained in the IM AF file. We have implemented the brand 'im02', which allows up to 6 simultaneous tracks of audio. Thus the IM AF file created can be easily reproduced in any kind of device without requiring high processing capabilities. The total size of this box is

24 bits.

The next step is to extract the audio samples of each of the MP3 files and store them in the box Media Data Box 'mdat'. The encoder searches the header of the first frame of the MP3 file, skipping any ID3 TAGs, and stores the samples byte by byte. Note that the audio tracks are stored one after another in the IM AF file. The player will need information in order to extract these audio tracks contained in the IM AF file. This information is stored in the Sample Table Box 'stbl'.

IM AF file stores a Track Box 'trak' for each audio track, containing several tables with the information necessary for the player to play the media data. Those tables are saved in boxes inside the Sample Table Box 'stbl'.

The Sample Description Box 'stsd' stores the technical specifications of an audio track. It contains information on the type of encoding used, bit rate, sample rate and specific information for decoding. The encoder we developed sets a sample rate of 44100 Hz and bit rate of 128 kbps.

The Sample Size Box 'stsz' stores the number of frames of the audio track and a table with the number of bytes in each frame. In the case of MP3 files with constant bit rate of 128 kbps, the frames may be 417 or 418 bytes. The IM AF encoder reads byte by byte the MP3 file, searching all possible headers. When there is a known header, the algorithm counts the number of bytes to find the next header known. Thus, the number of frames and their size are calculated.

The Sample Table Box 'stts' contains a table that stores for each entry two values: the number of contiguous frames that have the same length and duration of these frames (in miliseconds).

Frames are grouped into chunks. The Sample to Chunk Box 'stsc' defines a table that includes the number of frames of each chunk and the number of chunks that have the same number of frames. For constant bitrate MP3s, this table includes only one entry, because all the frames are equally sized.

In the Sample Size Box 'stsz' are stored the number of MP3 frames once more and a table

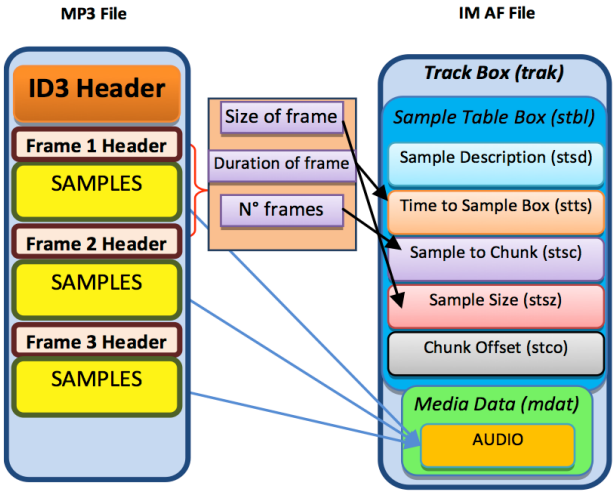

Figure 1.5: Audio samples and associated data transfer from MP3 to IM AF

giving the size (in bytes) of each of these audio frames.

Finally, the Chunk Offset Box 'stco' indicates the position of the first frame of the audio track in the Media Data Container Box 'mdat' of the IM AF file.

Figure 1.5 illustrates the process explained above.

1.3.2 3GPP Timed Text

Another feature of IM AF is that allows timed-text which may represent the lyrics of the song for karaoke applications. The format of the text track is defined in 3GPP TS 26.245 standard [14]. IM AF file stores character strings to be displayed, text modifiers that describe how the text should be shown, and temporal information in order to display the text in perfect sync with the music.

Strings and text modifiers are stored in the Media Data Box 'mdat', and information to display the text in sync with the audio is stored in the boxes as 'stts', 'stsc', 'stco'. The box 'stsd'

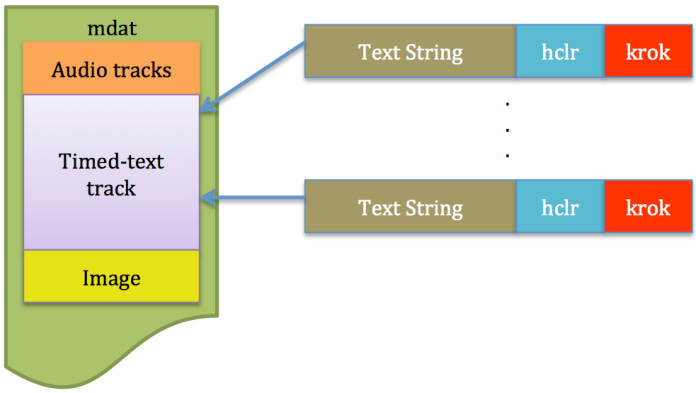

Figure 1.6: Lyrics encoding

stores the way in which to display the text.

Text strings are stored in an external text file with the following syntax:

[start time] Text String [end time]

The instant of time (in seconds) in which the text string is displayed appears before the text string. Secondly, the instant of time (in seconds) at which the string is no longer displayed on the screen is stored after the text string.

The encoder seeks every text string inside the text file and it copies them in the 'mdat' box of IM AF file. After a text string is stored, the encoder writes the text modifier boxes. The first modifier box is the Text Highlight Color Box 'hclr', it specifies the highlight color (expressed as hexadecimal rgb color) of the text. The second modifier box is the Text Karaoke Box 'krok'. This box specifies the highlighting color start/end time of words in a phrase (see Figure 1.6).

Storage of timing information is similar to audio tracks, as the same boxes are used ('stts', 'stsc', 'stsz' and 'stco', as showed in Figure 1.5). Duration for every string is saved in 'stts' box (sample delta = duration of interval*timescale). In 'stsz' the size of every string is stored including the modifiers. Another important box is 'tx3g', contained inside Sample Description Box ('stsd'),

Metadata	Location
track level	trak/meta box
song level	moov/meta box
album level	meta box for file

Figure 1.7: Levels of metadata in IM AF

which defines font type and size, horizontal and vertical justification or background colour.

1.3.3 Metadata

There are two types of metadata included in the IM AF file :

- metadata for the song and track;
- metadata for the album;

Figure 1.7 shows the boxes containing the various types of metadata.

The 'meta' box has two distinct functions : it stores the descriptive metadata (see Figure 1.8) and provides information about the JPEG picture included in the IM AF file by locating its length (in bytes) and its offset within the file.

Description	Level		
	Album	Song	Track
Title	•	•	•
Singer	•	•	-
Composer	-	•	-
Lyricist	-	•	-
performing musician	-	-	•
Genre	•	•	-
file date	•	•	•
CD track number of the song	-	•	-
Production	•	•	-
Publisher	•	•	-
copyright information	•	•	-
ISRC (International Standard Recording Code)	-	•	-
Image	•	•	-
URL site address related to the music and the artist (e.g. album homepage, music video)	•	•	-

Figure 1.8: Descriptive metadata in IM AF

The container for metadata information is the Meta Box 'meta'. This box includes a Handler

Box 'hdlr' indicating the structure of the contents. The handler type is set to 'mp7t'.

The descriptive metadata is located in the XML Box 'xml' in XML format. The IM AF encoder simply includes such information in the resulting file by writing the string value. For now, this value can only be changed in the source code.

1.3.4 JPEG Still Pictures

Another feature of the IM AF interactive player is that it can display images while a song is played. This is taking place by supporting the JPEG file format for still pictures [13]. Images can be associated, e.g., to music albums cover and/or artist's photos.

In the box 'mdat' the entire content of a JPEG file is stored. On the other hand, in the box 'meta' is saved the size of the picture and the picture offset within 'mdat'.

The first step performed by the encoder is to calculate the size of the image. This task is performed by the getImageSize () function. The image size is used to calculate the size of the box 'mdat', which is the sum of the size of the audio tracks, the track size of text and image.

The Item Location Box 'iloc' stores the images size and its offset in 'mdat', in particular into the 'extent-lenght' and 'extent-offset' values. The Item Information Box 'iinf' contains the picture's name ('item-name') and its encoding type ('content-encoding').

The insertImage() function inserts the image in the IM AF file within the box 'mdat'.

Figure 1.9 shows the JPEG picture insertion process in IM AF file during encoding.

1.3.5 Groups

For the creation of groups, two boxes are used: Group Container Box 'grco' and Group Box 'grup'. The Group Box is located inside Group Container Box.

The groupcontainer() function is responsible for creating these boxes and returns the size of

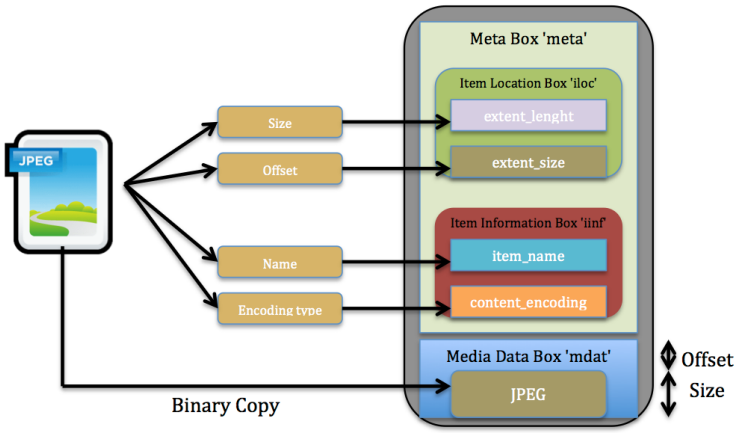

Figure 1.9: JPEG inclusion process in IM AF

the Group Container Box.

The Group Container Box stores the number of groups that have been defined inside the IM AF file. The encoder currently allows you to create one group, although the code can be modified easily to allow the creation of multiple groups.

The Group Box contains all other information necessary to define a group. The name assigned to the group is stored in the variable 'groupname'. An array stores the identifiers of the audio tracks that make up the group. The variable 'group-activation-mode' contains a flag that defines the way the elements of a group shall be activated when a group is switched on. The encoder assigns the value '1' to the variable. This means that the decoder plays all audio tracks contained in the group. The variable 'group-reference-volume' specifies the volume gain applied on this group when it is switched on. The variable 'flags' stores the value 0x02, so the group information contained in the Group Box is enabled to display on the player, but it is disabled to be edited by the user.

preset_type	availability	description
0	✔	static track volume preset
1	✔	static object volume preset
2	✔	dynamic track volume preset
3	✔	dynamic object volume preset
4	✔	dynamic track approximated volume preset
5	✖	dynamic object approximated volume preset
6	-	Value reserved
7	-	Value reserved
8	✔	static track volume preset with EQ
9	✖	static object volume preset with EQ
10	✖	dynamic track volume preset with EQ
11	✖	dynamic object volume preset with EQ
12	✖	dynamic track approximated volume preset with EQ
13	✖	dynamic object approximated volume preset with EQ

Figure 1.10: Presets defined in IM AF standard and their availability in reference software

1.3.6 Presets

The IM AF standard defines some default presets. Despite the availability of 11 presets, only 6 of them are implemented in the reference software. Figure 1.10 shows the available presets.

There are two categories of presets: static (fixed volume in each audio track) and dynamic (volume of the tracks can vary over time). There are two boxes involved in presets: Preset Container Box 'prco' and Preset Box 'prst'. The Preset Box is contained inside the Preset Container Box. The function presetcontainer() creates the presets and returns the size of 'prco' box. A switch-case structure selects the implementation, with 'preset-type' value as switching variable. The static presets require the definition of the playback volume gain for each audio track or object, by setting the 'preset-volume-element'. The presets that involve objects require the definition of the 'output-channel-type', that is the number of channels of each included audio track. The encoder sets this value to '1', for stereo tracks.

There is only one working dynamic preset "Dynamic track approximated volume preset". This preset is used to create a fade-in or fade-out effect. The variables 'start-sample-number'

SELECTION RULES						
	mixing rule type	element ID	key element ID	min num elements	max num elements	rule description
Min/Max	0	Group ID	-	✔	✔	✔
Exclusion	1	✔	✔	-	-	✔
Not mute	2	✔	-	-	-	✔
Implication	3	✔	✔	-	-	✔

Figure 1.11: Parameters in Selection Rules

and 'duration-update' indicate the sample where the volume change takes place and the number of samples that it occurs, respectively.

1.3.7 Rules

The function rulescontainer() is responsible for creating rules. It defines the structure of the Rule Container Box 'ruco' , which hosts a Selection Rule Box 'rusc' and a Mixing Rule Box 'rumx'.

Two switch-case structures are responsible for creating the desired Selection and Mixing rules, respectively.

Figures 1.11 and 1.12 show the parameters needed by the encoder to create each of the rules.

The 'element-ID' is an integer which represents the ID of the element involved in a selection rule. In the case of Min/Max rule , it must be the same ID of the group. In order to represent the ID of the element on which the rule is applied the encoder uses the 'key-element-ID'. The 'rule description' is a character string which allows a rule to be named.

1.4 IM AF in Sonic Visualiser

Sonic Visualiser is an application for viewing and analysing the content of audio files. This open source software was developed at the Centre for

MIXING RULES						
	selection rule type	element ID	key element ID	min volume	max volume	rule description
Equivalence	0	✔	✔	-	-	✔
Upper	1	✔	✔	-	-	✔
Lower	2	✔	✔	-	-	✔
Limit	3	✔	-	✔	✔	✔

Figure 1.12: Parameters in Mixing Rules

Digital Music at Queen Mary, University of London. IM AF codec has been fully integrated into Sonic Visualiser. Figure 1.13 shows the interface of the encoder in Sonic Visualiser. The source code as well as executables for both Windows and Mac OS/X of Sonic Visualiser with IM AF support are available through the SoundSoftware.ac.uk repository [18].

The integration of the IM AF codec in SV allows the user to create IM AF files in a quick, convenient and reliable way.

Furthermore, importing an IM AF file into SV opens up and display at once both individual music tracks and mixed versions (presets) of a song. This is a huge relief for SV users that otherwise would have to search their folders and import all of the music audio tracks one by one. That is, SV with IM AF support can be used by music producers as an archival tool for saving and importing their music projects at once, thus increasing their time-efficiency and productivity.

Furthermore, SV with IM AF support has been converted from an audio analysis and visualisation tool to a multimedia editing suite (e.g., multi-track audio, lyrics, pictures, etc).

Including text synchronized with audio in IM AF offers great potential to be used in conjunction with some SV plug-ins. The possibility of extracting the chords of a song by using Chordino [10] plug-in and present them in perfect sync

with the song's lyrics can be of great help to those musicians who wish to practice the songs of their favorite artists and improve their musical/vocals skills.

It is equally possible to extract the melody of a song by using Melodia [11] plug-in and display it perfectly aligned with the lyrics.

In order to facilitate interaction with the audio tracks, a panel with gain knobs to control the volume of each of the audio tracks is also included. Subsequent versions of our IM AF codec will allow users to record in real-time their own mixes of songs using these gain knobs and share them in social networks.

In the following section some use cases scenarios are presented together with instructions on how to use the IM AF codec in Sonic Visualiser. The first subsection is devoted to the steps necessary to create an IM AF file in Sonic Visualiser with certain characteristics. The following two subsections describe the steps required to extract the chords and the melody of a certain musical instrument of an IM AF song.

1.4.1 Creation of an IM AF file

We will show the steps to create an IM AF song with multiple audio tracks, two rules, a fade-in effect, a timed-text track and a picture. A similar process can be used for any combination of features that may be needed.

1. Download and run Sonic Visualiser.
2. File ->Open...
3. Select an audio track.
4. To include more audio tracks : File ->Import More Audio...
5. Select an audio track. Repeat steps 4) and 5) until you have opened all the desired audio tracks.
6. File ->Export IMAF File... At this point the interface of the IM AF encoder is popped up.

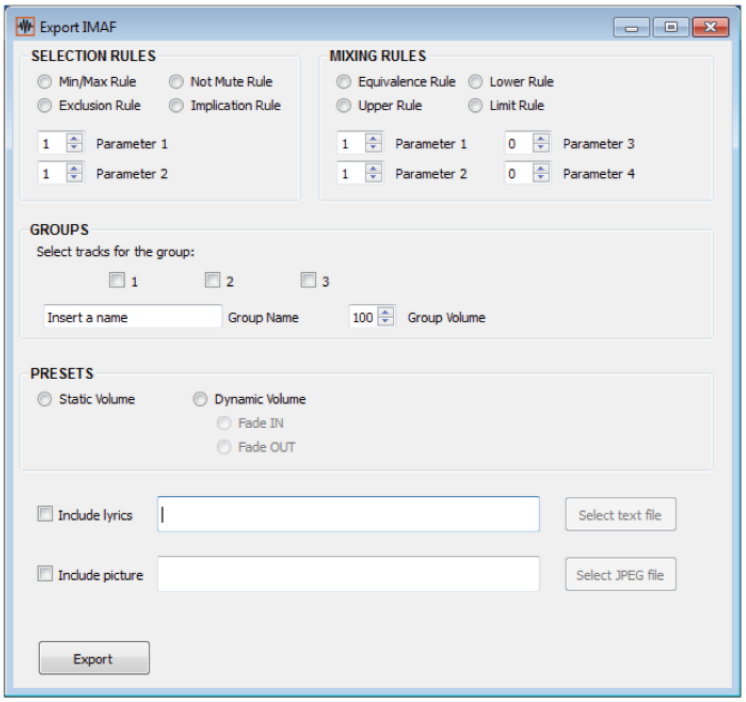

Figure 1.13: IM AF encoder interface in Sonic Visualiser

7. Select “Implication Rule” and choose Track A=1 and Track B=2. Thus if Track B is active then track A will also be active. If track A is inactive the track B will also be inactive.

8. Select “Limit Rule”, choose Track A=1, Min Volume=10, Max Volume =20. The minimum volume level of Track A will be 10, and the maximum volume level will be 20.

9. Select “Dynamic Volume” and “Fade IN”. A Fade-in effect is a gradual increase in the level of an audio track from silence at the beginning.

10. Select “Include lyrics” and press “Select text file”. Choose the text file. The syntax used to create the text file must be the same as shown in subsection 1.3.2.

11. Select “Include picture” and press “Select JPEG file”. Choose the image file.

12. Press “Export”.

13. Select the name of the IM AF file and the folder. Press “Save”.

14. If the file has been created properly you will see the next message : “IM AF file successfully created!”.

1.4.2 Chords extraction time aligned with lyrics

One of the possibilities offered by the integration of IM AF codec in Sonic Visualiser is the possibility of extracting the chords of a song and present them as text synchronized with the audio (see Figure 1.14). We will use a Sonic Visualiser plug-in for chords extraction, known as Chordino [10].

1. Run Sonic Visualiser.

2. File ->Import IMAF File...

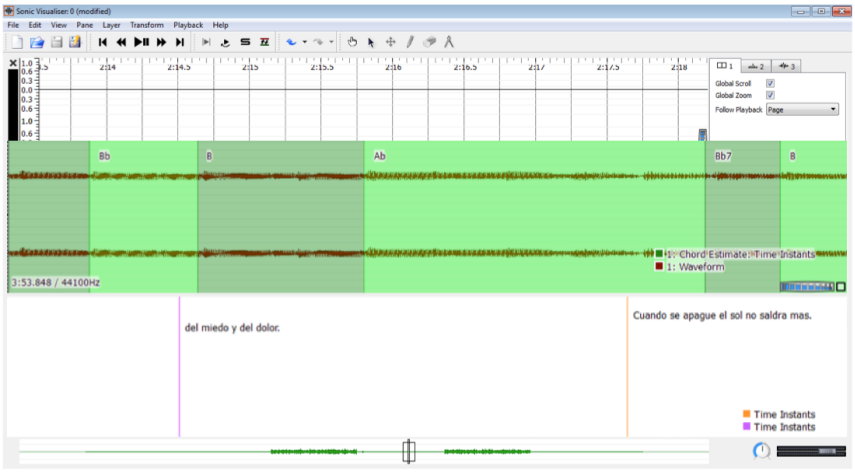

Figure 1.14: Lyrics aligned in time with chords in Sonic Visualiser with IM AF support.

3. Select an IM AF file.

4. You will see the waveforms of the audio tracks, a pane with the lyrics and a panel with the gain knobs of the different tracks.

5. Transform ->Analysis by Plugin Name-> Chordino->Chord Estimate... Select the audio track to apply the Plugin in Input Material. Press Accept.

1.4.3 Melody extraction time aligned with lyrics

Another interesting application of SV is the extraction of the melody pitch of a song. Using the VAMP plug-in called Melodia [11], an algorithm for melody pitch estimation, it is possible to have a more efficient pitch extraction from only the vocal track, rather than the mixed song. Furthermore, the melody pitch is now presented aligned in time with the song's lyrics (Figure 1.15).

1. Run Sonic Visualiser.

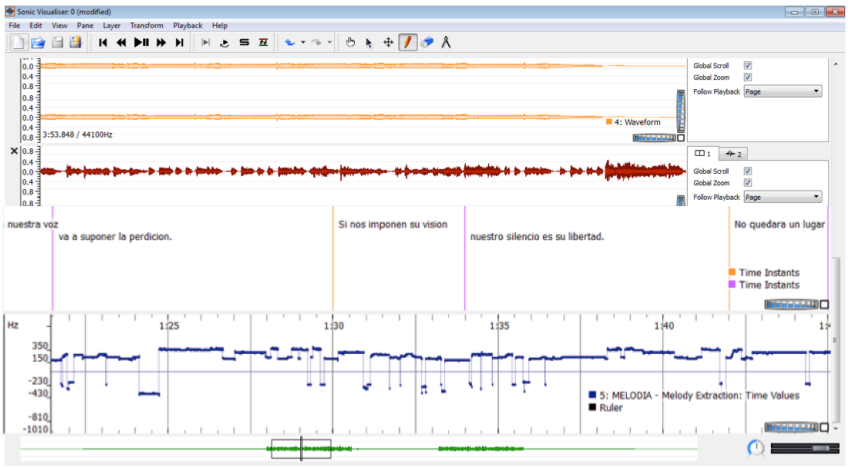

Figure 1.15: Lyrics aligned in time with melody pitch in Sonic Visualiser with IM AF support.

2. File ->Import IMAF File...
3. Select an IM AF file.
4. You will see the waveforms of the audio tracks, a pane with the lyrics and a panel with the gain knobs of the different tracks.
5. Pane ->Add New Pane.
6. Transform -> Analysis by Plugin Name -> MELODIA->Melody Extraction... Select the audio track to apply the Plugin in Input Material. Press Accept.

1.5 Future Developments and Conclusions

The integration of IM AF in Sonic Visualiser opens up new possibilities for researchers in the area of signal processing, offering a testbed for the development of new plug-ins such as:

- on extracting the lyrics from the vocal track (singing to text) [19] and automatic highlightning of lyrics. Thus relieving IM AF

creators from having to insert the lyrics and their time-stamps for highlighting. In particular, the latter task is very tedious, so a plug-in to make this process automatically would be extremely useful for Karaoke fans;

- on source separation for extracting music instruments from a mix-down song version [20];

As it has been already mentioned, it would be interesting to enable users to record each tracks' volume while disc-jockeying in Sonic Visualiser and share their own presets with their friends using social networks (e.g. Facebook). Due to the flexibility of the IM AF format, the file does not have to contain the audio tracks themselves but only their URL. In that way the tracks can be recalled from a server. As a result, it enables efficient exchange and sharing of IM AF files, which they may include only the mixing parameters.

Furthermore, an alternative cross-platform and cross-browser HTML5 IM AF player [21] has also been developed for users that would only like to listen to their friends' mixes with no need to download Sonic Visualiser.

This chapter described the design and implementation of an IM AF codec and its integration in SV as well as highlighting the benefits of their combined features with some use case scenarios.

1.6 Acknowledgments

Panos Kudumakis acknowledges that this work has been partially done during his visit at the University of Malaga in the context of the program Andaluca TECH: Campus of International Excellence and in conjunction to UK EPSRC project EP/H043101/1 SoundSoftware.ac.uk

Bibliography

[1] P. Kudumakis, "MP3: something's gotta change!", Audio!, pp. 6, Vol. 1, Issue 3, April 2011. Editors: P. Curzon, M. Barthet and S. Dixon.

[2] I. Jang, P. Kudumakis, M. Sandler, K. Kang: "The MPEG Interactive Music Application Format Standard", IEEE Signal Processing Magazine, pp. 150-154, Vol. 28, Issue 1, Jan. 2011.

[3] iKlax Media, (Online), Available: `http://www.iklaxmusic.com` (last accessed 12.01.14).

[4] MOGG files, Multitrack Digital Audio Format, (Online), Available: `http://moggfiles.wordpress.com` (last accessed 12.01.14).

[5] MT9, `http://en.wikipedia.org/wiki/MT9`(last accessed 12.01.14).

[6] ISO/IEC 23000-12:2010 - Information Technology - Multimedia Application Format (MPEG-A) – Part 12: Interactive Music Application Format.

[7] ISO/IEC 23000-12:2010/Amd.2:2012 - Information technology - Multimedia application format (MPEG-A) - Part 12: Interactive music application format, AMENDMENT 2: Compact representation of dynamic volume change and audio equalization.

[8] Jesus C. Garcia, Costantino Taglialatela, Panos Kudumakis, Lorenzo J. Tardon, Isabel Barbancho and Mark Sandler, 'Interactive music applications by MPEG-A support in Sonic Visualiser', AES 53rd International Conference on Semantic Audio, London, UK, 27-29 Jan. 2014.

[9] C. Cannam, C. Landone and M. Sandler: Sonic Visualiser: An Open Source Application for Viewing, Analysing, and Annotating Music Audio Files, Proceedings of the ACM Multimedia 2010 International Conference.

[10] M. Mauch and S. Dixon: Approximate Note Transcription for the Improved Identification of Difficult Chords, Proceedings of the International Symposium on Music Information Retrieval, pp. 135- 140, 2010.

[11] J. Salamon and E. Gomez: Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics, IEEE Transactions on Audio, Speech and Language Processing, 20(6):1759-1770, Aug. 2012.

[12] ISO/IEC 23003-2:2010 - Information technology - MPEG audio technologies - Part 2: Spatial Audio Object Coding (SAOC).

[13] ISO/IEC 10918-1:1994 - Information technology - Digital compression and coding of continuous-tone still images (JPEG).

[14] ETS 3GPP TS 26.245-2004 - Transparent end-to-end Packet switched Streaming Service (PSS); Timed text format.

[15] ISO/IEC 15938-5:2003 - Information technology - Multimedia content description interface Part 5: Multimedia description schemes.

[16] ISO/IEC 14496-12:2008 - Information technology - Coding of Audio-Visual Objects Part 12: ISO base media file format.

[17] Costantino Taglialatela, "MPEG IM AF encoder: Features development", BSc thesis, Seconda Universit degli Studi di Napoli, Italy , July 2013.

[18] Panos Kudumakis, "MPEG developments" (Online), Available: `https://code.soundsoftware.ac.uk/projects/mpegdevelopments` (last accessed 12.01.14).

[19] T. Hosoya, M. Suzuki, A. Ito and S. Makino: Lyrics recognition from a singing voice based on finite state automation for music information retrieval, Proceedings of the International Symposium on Music Information Retrieval, pp. 532-535, 2005.

[20] J. Han, Z. Rafii and B. Pardo, Audio Source Separation and REPEAT, Research projects of Northwestern University, Dep. of Elec. Eng. and Comp. Sc., (Online), Available: `http://music.cs.northwestern.edu` (last accessed 12.01.14).

[21] G. Herrero, P. Kudumakis, L. J. Tardon, I. Barbancho and M. Sandler, "An HTML5 Interactive (MPEG-A IM AF) Music Player", 10th International Symposium on Computer Music Multidisciplinary Research (CMMR), Marseille, France, 15-18 Oct. 2013.

Referencias

[1] P. Kudumakis, "Mp3: something's gotta change!" *Audio!*, vol. 1, no. 3, p. 6, Apr. 2011.

[2] iKlax Media, http://www.iklaxmusic.com, [Online; último acceso 1-Mayo-2014].

[3] MT9, http://en.wikipedia.org/wiki/MT9, [Online; último acceso 1-Mayo-2014].

[4] MOGG, http://moggfiles.wordpress.com, [Online; último acceso 1-Mayo-2014].

[5] I. Jang, P. Kudumakis, M. Sandler, and K. Kang, "The mpeg interactive music application format standard," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 150–154, Jan. 2011.

[6] "Information technology - multimedia application format (mpeg-a) – part 12: Interactive music application format," Standard ISO/IEC 23000-12:2010.

[7] C. Cannam, C. Landone, and M. Sandler, "Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files," in *Proceedings of the ACM Multimedia 2010 International Conference*.

[8] "Information technology - coding of audio-visual objects - part 12: Iso base media file format," Standard ISO/IEC 14496-12:2008.

[9] "Transparent end-to-end packet switched streaming service (pss); timed text format," Standard ETS 3GPP TS 26.245-2004.

[10] "Information technology - digital compression and coding of continuous-tone still images (jpeg)," Standard ISO/IEC 10918-1:1994.

[11] MP4-Browser, http://www.softpedia.com/progDownload/MP4-Browser-Download-73668.html, [Online; último acceso 1-Mayo-2014].

[12] Qt-Project, http://qt-project.org, [Online; último acceso 1-Mayo-2014].

[13] "Information technology - mpeg audio technologies - part 2: Spatial audio object coding (saoc)," Standard ISO/IEC 23003-2:2010.

[14] "Information technology - multimedia content description interface - part 5: Multimedia description schemes." Standard ISO/IEC 15938-5:2003.

[15] "Information technology - multimedia application format (mpeg-a) - part 12: Interactive music application format, amendment 2: Compact representation of dynamic volume change and audio equalization." Standard ISO/IEC 23000-12:2010/Amd.2:2012.

[16] M. Mauch and S. Dixon, "Approximate note transcription for the improved identification of difficult chords," in *Proceedings of the International Symposium on Music Information Retrieval*, pp. 135–140.

[17] J. Salamon and E. Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," in *IEEE Transactions on Audio, Speech and Language Processing*, pp. 1759–1770.

[18] T. Hosoya, M. Suzuki, A. Ito, and S. Makino, "Lyrics recognition from a singing voice based on finite state automation for music information retrieval," in *Proceedings of the International Symposium on Music Information Retrieval*, pp. 532–535.

[19] Z. Rafii and B. Pardo, "Repeating pattern extraction technique (repet): A simple method for music/voice separation," in *IEEE Transactions on Audio, Speech, and Language Processing*, Jan. , 2013.

[20] G. Herrero, P. Kudumakis, L. J. Tardon, I. Barbancho, and M. Sandler, "An html5 interactive (mpeg-a im af) music player," in *10th International Symposium on Computer Music Multidisciplinary Research (CMMR)*, Oct. , 2013.

[21] J. C. Garcia, C. Taglialatela, P. Kudumakis, L. J. Tardon, I. Barbancho, and M. Sandler, "Interactive music applications by mpeg-a support in sonic visualiser," in *AES 53rd International Conference on Semantic Audio*, London, Jan. 27–29, 2014.

[22] J. C. Garcia, P. Kudumakis, I. Barbancho, L. J. Tardon, and M. Sandler, "Enabling interactive and interoperable semantic music applications," in *Springer Handbook for Systematic Musicology.*