UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE MATEMATICHE,
FISICHE E NATURALI

Corso di Laurea in Scienze e Tecnologie della Comunicazione Musicale

PROTOTIPO DI MEDIA PLAYER IN FLASH CON GESTIONE DI PLAYLIST IN XML

Elaborato finale di:
Invernizzi Norberto
Matricola 742701

Relatore: Prof. Luca Andrea Ludovico

Correlatore: Dott. Adriano Baratè

Anno Accademico 2010-2011

“A mio padre”

INDICE

INTRODUZIONE

ActionScript 3.0 è il linguaggio di programmazione dell'ambiente runtime di Adobe Flash Player. Rende possibili l'interattività, la gestione dei dati e molte altre operazioni relative al contenuto e alle applicazioni Flash. ActionScript 3.0 viene eseguito dall'AVM (ActionScript Virtual Machine), che fa parte di Flash Player. Il codice ActionScript viene solitamente compilato in formato codice byte (una sorta di linguaggio di programmazione generato e interpretato dai computer) mediante un compilatore che, a sua volta, lo incorpora in file SWF, che vengono quindi eseguiti in Flash Player (l'ambiente runtime).
ActionScript 3.0 implementa ECMAScript for XML (E4X), recentemente standardizzato come ECMA-357. E4X offre una serie funzionale di costrutti di linguaggio per la gestione del codice XML. Esso, infatti, sveltisce lo sviluppo di applicazioni che utilizzano il codice XML, riducendo drasticamente la quantità di codice richiesta.
Lo scopo di questo progetto di tesi è la creazione attraverso l'utilizzo del linguaggio sopra descritto, di un player MP3 contenente una playlist cliccabile e scrollabile. La lista dovrà contenere tutti i titoli delle canzoni ed i nomi degli artisti, creati attraverso un documento XML. Ogni elemento XML sarà associato al brano corrispondente. Il vantaggio della configurazione di una playlist in linguaggio XML è la possibilità di inserire all'interno dell'elenco un numero svariato di canzoni.
La tesi si divide in tre capitoli principali. Il Capitolo 1 è una breve introduzione sui file multimediali MP3 e sull'importanza avuta nella rivoluzione musicale di questi ultimi anni. Il Capitolo 2 presenta una breve rassegna sulle tecnologie adoperate nel progetto. Per quanto riguarda l'XML verranno definiti i concetti di un documento valido e ben formato; nel contesto della validazione si andrà a sottolineare la nozione di schema e i due principali linguaggi di definizione di schemi: DTD e XML Schema. Nell'analisi del software Flash verranno presi in considerazione il suo Player ed il linguaggio a lui associato.
L'elaborato, infine, termina con la descrizione del progetto creato e con alcuni commenti finali sul lavoro svolto.

CAPITOLO 1
IL FORMATO MP3

1.1 Campionamento e Quantizzazione

Quando viene trattato l'argomento MPEG-1 Layer 3, è doveroso introdurre alcuni concetti che sono alla base del termine MP3. L'MP3 è di per sé un segnale audio digitale, ma per trasformare un segnale analogico in uno digitale è indispensabile che esista una conversione capace di mutare quest'ultimi reciprocamente. La conversione da analogico a digitale si può suddividere in tre fasi principali: campionamento, quantizzazione, codifica. Il segnale audio si presenta come segnale analogico, ovvero un segnale capace di variare nel tempo il suo andamento in modo continuo. La figura sottostante rappresenta la riproduzione grafica di un segnale analogico.

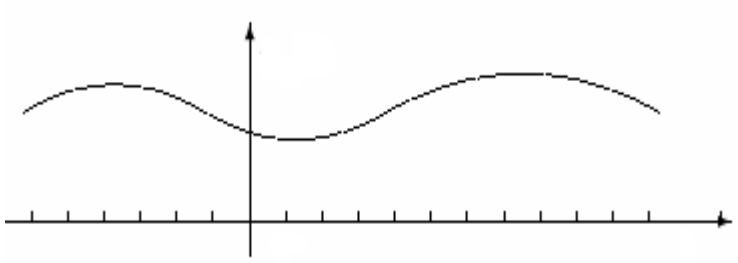

Figura 1. Riproduzione di un segnale analogico

In questo grafico è possibile notare come l'ampiezza possa assumere una serie di valori passando da un valore massimo a un valore minimo.
Come riscontro finale è possibile definire il segnale analogico come un segnale capace di assumere qualsiasi valore (all'interno di un range noto).
Differente è il caso del segnale digitale. Esso infatti può assumere due soli stati (High(1), Low(0)). Il grafico rappresenta quello appena descritto.

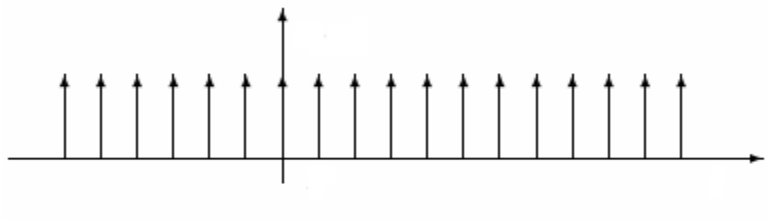

Figura 2. Riproduzione di un segnale digitale

Cercare di rappresentare un segnale audio analogico (conservando il suo andamento), con un segnale digitale (conservando l'informazione), è un nodo cruciale che sta alla base della teoria dell'informazione. Ciò è reso possibile dalla campionatura, ossia il prelievo ad intervalli regolari del valore del segnale audio. Si renderà quindi indispensabile approssimare la funzione logica con la funzione digitale.

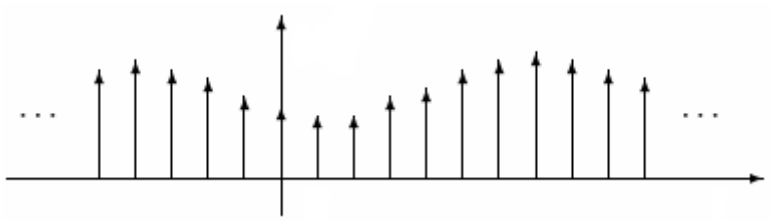

Figura 3. Campionamento di un segnale

Ad ogni istante multiplo di un valore fondamentale, detto passo di campionamento, viene assegnato uno stato che approssima la funzione in quel punto. Se l'insieme degli stati è riprodotto frequentemente, vorrà dire che la funzione di partenza è ben approssimata.
Ad esempio, se il segnale analogico è stato suddiviso in 10 stati, ognuno di durata T, significa che la durata complessiva è di 1ms e ogni passo di campionamento è 0,1ms.
Da questi valori è possibile calcolare la frequenza di campionamento, ossia il numero di campioni digitali prelevati da un segnale audio in un istante di tempo:

$$f = 1/T = 1/0{,}1 \text{ ms} = 10 \text{ KHz}$$

Poiché la digitalizzazione , ad esempio di un suono, prevede l'impiego di un numero di bit finito, è essenziale ottenere un numero finito di valori discreti. Tale processo avviene grazie alla quantizzazione. I valori possibili del suono vengono limitati tra un massimo e un minimo, in modo da definire la dinamica del quantizzatore, inoltre, i valori al di fuori del range massimo-minimo, vengono approssimati al valore più vicino già fissato. Chiaramente ciò comporta l'introduzione di errori chiamati errori di quantizzazione, dovuti al distacco tra il valore reale e il valore quantizzato.
Analizzando nel dettaglio le tracce audio di un cd è possibile notare che i file sono immagazzinati in file binari. Ogni secondo vengono prelevati 44100 campioni quantizzati a 16 bit ciascuno.

Considerando poi che si hanno a disposizione 2 canali, il calcolo del bit-rate risulta facile:

bit-rate = 44100*16*2 = 176400 byte/secondo

Questo bit-rate definisce le qualità sonore di un normale cd audio. Questo standard viene preso di riferimento quando vengono stimati i risultati ottenuti dagli algoritmi di compressione. L'orecchio umano riesce ad avvertire frequenze fino ai 20KHz. Oltre questo limite non percepisce nulla, così come sotto i 20 Hz. In breve, come risultato finale per la qualità di un cd, si è scelto di impiegare una frequenza che sia pari alla massima frequenza di interesse moltiplicata per due (teorema del campionamento), 20mila per due fa 40mila, da qui si spiega il perché del valore 44100.
Lo standard PCM , particolarmente impiegato per manipolare un segnale audio in forma digitale non compressa, adotta proprio un rate di campionamento di 44100 campioni al secondo quantizzati a 16 bit per canale. Un formato audio molto simile al PCM è il WAV.
Essendo comunque entrambi formati non compressi, generano file audio molto grandi e per questo motivo questi standard stanno perdendo popolarità con il tempo, lasciando spazio a formati compressi come appunto l'MP3.

1.2 Metodi di compressione audio

La compressione dei dati è una tecnica utilizzata per cercare di ridurre nel modo più efficiente le dimensioni di un file. Questa tecnica nasce quasi in contemporanea con l'avvento di internet, per ovviare al problema del trasporto di ingenti quantità di informazione nel minor tempo possibile. Alla base della funzione di compressione vi è un' applicazione avanzata di algoritmi matematici. Ogni anno vengono investiti importanti somme di denaro per la ricerca di nuovi algoritmi sempre più efficienti.
È fondamentale chiarire sin da subito che non esiste un algoritmo definitivo, unico e completo in grado di comprimere tutti i tipi di dati. Per fare un esempio, non è pensabile di comprimere un'immagine a colori in modo soddisfacente se si adopera un algoritmo studiato per comprimere file binari generici. Programmi come Winrar utilizzano un insieme di tecniche di compressione che si adattano a seconda del tipo di file che deve essere trattato.

Un primo metodo per classificare i modelli di compressione è quello di distinguere le tecniche che non perdono l'informazione durante il processo di compressione (lossless), da quelle che, al contrario, prevedono un abbandono dell'informazione(lossy). [1]
Le tecniche lossless, abitualmente vengono impiegate per la compressione di dati che non prevedono la perdita di nessun bit informazione quali testi, documenti, programmi. Differentemente, l'audio concede un certo livello di degradazione in quanto è un compromesso ammissibile per limitare l'occupazione della banda richiesta dal file.
Le codifiche di compressione audio sono svariate e si servono di tecniche molto diverse l'una dall'altra. Esse si distinguono in tre modelli principali: codifiche nel dominio del tempo, codifiche per modelli e codifiche nel dominio delle frequenze. Spesso le prime due vengono impiegate per comprimere un segnale vocale, mentre la terza viene adottata per la compressione della musica.
Le codifiche nel dominio del tempo vengono definite così perché riescono a sviluppare il segnale campionato senza aver estratto le frequenze spettrali. Storicamente sono le prime ad essere state inventate e, chiaramente, con il passare degli anni verranno sostituite da nuovi algoritmi. Le più importanti sono la DPCM e l'ADPCM.
La Differential Pulse Code Modulation è un codificatore di segnale che utilizza la linea di base della Pulse Code Modulation (PCM), ma aggiunge alcune funzionalità basate sulla predizione dei campioni del segnale. Questo approccio richiede quindi molti meno bit per poter spostare l'informazione, visto che si trasmette solo la differenza tra i campioni e non i valori effettivi. In un segnale audio si riduce di circa il 25% il numero di bit richiesti per campione rispetto la PCM. Questa tecnica agisce in modo favorevole con la voce umana, poiché essa gode di proprietà specifiche, ma si pone in modo negativo quando si tratta di proprietà vocali e strumentali.
Molto simile alla DPCM è l'Adaptative Differential Pulse Code Modulation(ADPCM), in cui vengono trasmessi sempre i bit differenza, ma tenendo conto dei bit passati. In modo semplicistico si cerca di prevedere in anticipo la forma dell'onda, vale a dire quali saranno i suoi campioni futuri sulla base di quelli passati.
Le codifiche per modelli si avvalgono di algoritmi di compressione legati ad una fonte sonora ben definita, che si tenta di uguagliare attraverso un modello più agevolato. Queste codifiche sono ottime per la compressione della voce, al punto che vengono impiegate per le comunicazioni telefoniche. Le più importanti sono la LPC e la CELP.
La Linear Predictive Coding è un metodo di compressione eseguito esclusivamente per comprimere il suono vocale. LPC riesce a ridurre talmente tanto il suono vocale che la

voce umana viene paragonata a quella generata da un computer. Il vantaggio principale di questa tecnica e la produzione di un bit-rate molto basso. Per il suo funzionamento LPC pone la sua attenzione su un caratteristico elemento della voce; una volta riconosciuto l'elemento, viene trasmesso il codice affiliato estratto da una tavola di riferimento. A causa di alcuni errori dell'LPC (interlocutore non identificato) si è passati allo sviluppo di una nuova tecnica definita come CELP. La Code Excited Linear Predictor è stata creata sulla base della LPC, ma a differenza di quest'ultima, migliora la qualità, dato che trasferisce perfino l'informazione dell'errore dovuto alla codificazione LPC. La buona qualità di questa codifica ha portato l'uso della presente in vari ambiti di comunicazione come la telefonia e la videoconferenza.

1.3 Codifiche nel dominio della frequenza

Queste codifiche a differenza delle precedenti non analizzano i segnali da un punto di vista temporale, ma li esaminano in base alla loro frequenza. Questo tipo di codifica viene adoperata per la compressione musicale. Solitamente il suono vocale occupa solo limitate frequenze mentre gli strumenti musicali variano il range frequenziale a seconda del tipo di strumento adottato. Ogni suono emesso da uno strumento musicale presenta una tipica impronta spettrale ben definita, ovvero, il numero di frequenze presenti all'interno di uno spettro varia al variare della sua ampiezza. In verità, l'intervallo frequenziale di uno strumento dice poco. Come si fa a distinguere un "re" suonato da una chitarra da un "re" suonato da un flauto, considerando che la frequenza del "re" è la stessa? [2]

A questa domanda è possibile rispondere grazie alla definizione di timbro. Quando viene suonato il "re", non viene emessa solamente la frequenza appartenente a quella nota, ma vengono diffuse una serie di ulteriori frequenze multiple della fondamentale, conosciute con il nome di "armoniche". È proprio la disuguale divisione di queste frequenze che riconosce un "re" prodotto da una chitarra da quello prodotto da un flauto. Anche se la frequenza centrale, sia per la chitarra che per il flauto è sempre la stessa.

Quando invece le frequenze delle armoniche rimangono invariate, ma varia la fondamentale in modo maggiore o minore, si parla di cambio di pitch. Ad esempio se si ha a disposizione un "re maggiore" e un "re minore", entrambi avranno una distribuzione delle armoniche uguali (entrambi sono dei "re"): quello che cambia è l'andamento della frequenza centrale. In qualsiasi caso, la musica e i suoni devono essere ascoltati attraverso

l'apparato uditivo, quindi è indispensabile fare una breve analisi sul modello psicoacustico per capire meglio ciò che percepisce l'uomo quando capta dei segnali audio.

L'orecchio umano è per sua natura sensibile in misura diversa alle diverse frequenze. L'intensità della vibrazione avvertita dall'orecchio (loudness) varia al variare della frequenza , ciò determina che a parità di Sound Pressare Level (SPL), due toni frequenzialmente diversi vengono compresi con intensità differenti. La figura sottostante rappresenta l'andamento della percezione del suono in un momento di calma.

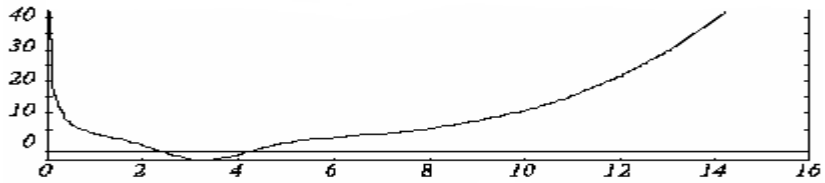

Figura 4. Capacita uditiva dell'orecchio umano

In stato di calma, tutti i valori che stanno al di sotto della curva vengono intesi dall'apparato uditivo come silenzio e quindi possono essere rimossi in fase di codifica.

Quando si è al cospetto di due toni differenti tra loro, ma prodotti contemporaneamente, quello che presenta un'intensità maggiore sovrasta sonoramente quello con intensità minore. Questo avvenimento viene chiamato mascheramento e consente di cancellare delle componenti in funzione del segnale esaminato. Il mascheramento si può suddividere in due modelli: il mascheramento frequenziale e il mascheramento temporale. [3]

Il mascheramento frequenziale avviene quando si hanno nello stesso istante di tempo un segnale forte e un segnale debole con frequenze adiacenti tra loro(facenti parte della stessa banda critica o a bande critiche confinanti). La conseguenza di tutto ciò è la produzione, da parte del segnale più forte, di una soglia di mascheramento al di sotto della quale scaturisce la mancata percezione del suono.

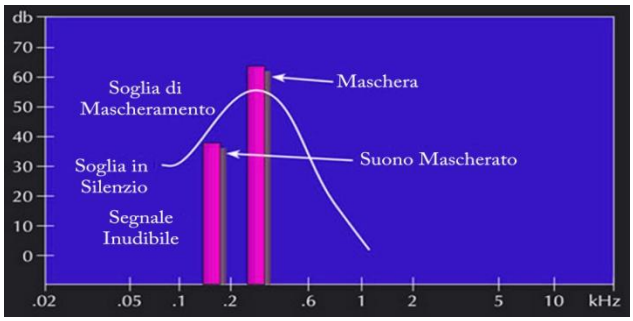

Figura 5. Rappresentazione del mascheramento frequenziale

Il mascheramento temporale fa si che la partecipazione di un tono di maschera continui a rendere non udibili i toni adiacenti anche quando questo non è fisicamente presente. Esso può verificarsi sia prima che il segnale di maschera si attivi (pre-mascheramento), sia dopo la sua disattivazione (post-mascheramento).

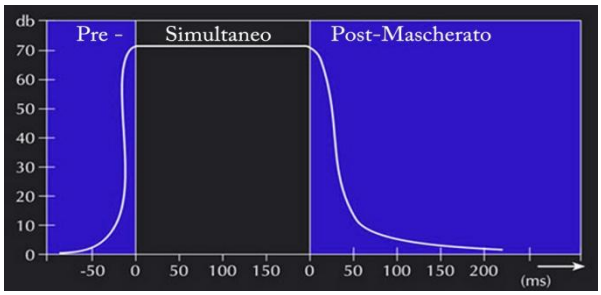

Figura 6. Rappresentazione del mascheramento temporale

Queste osservazioni vengono utilizzate negli algoritmi di compressione audio per eliminare le informazioni sulle frequenze non udibili. Gli schemi più avanzati che seguono questi principi sono il Dolby AC-3 e l'MPEG-1 (in particolare il Layer 3). Il primo è rivolto ad applicazioni multicanale, quali cinema e home theatre, mentre l'MPEG-1 è rivolto alle applicazioni audio stereofoniche.

1.4 Lo Standard MPEG

MPEG (Moving Picture Experts Group) è un gruppo di lavoro gestito dalla ISO/IEC il cui ruolo principale è quello di sviluppare nuovi di standard per la codifica audio-video digitale. [4]

Il primo progetto MPEG ebbe inizio nel 1988 e si concluse nel 1992 con la nascita dello standard internazionale MPEG-1, impiegato nei prodotti fondati su Video CD ed MP3. In seguito furono creati altri tre standard internazionali, MPEG-2, che trova il suo più grande utilizzo nelle codifiche televisive, MPEG-4 ed MPEG-7. MPEG-21 è l'ultimo progetto in lavorazione in casa MPEG: iniziato nel 2000, esso ricerca l'espansione e l'unificazione dei componenti dell'MPEG-4 e dell'MPEG-7 all'interno di un unico framework che copra tutti gli aspetti.

Una generica codifica MPEG è composta da tre entità principali legate tra loro:

- Formato di codifica: insieme di direttive definite dagli standard, che precisano il modo in cui deve essere codificata l'informazione sonora compressa.

- Encoder: applicazione il cui scopo è quello di prendere in input un file non compresso PCM e modificarlo in formato compresso, secondo lo standard di codifica MPEG adottato dall'utente.
- Decoder: applicazione il cui scopo è quello di prendere in input un formato di codifica compresso MPEG e restituirlo nel formato non compresso PCM.

Per rendere semplificabile la fase di encoding-decoding la complessità algoritmica è situata nell'encoder, in modo da rendere il più veloce possibile la funzione di decoding. Questo perché è dovere di chi amministra piattaforme multimediali attuare con l'encoder, file MP3 di elevata qualità, in modo da agevolare i compiti dell'utente finale. Esso, infatti, dovrà soltanto servirsi del decoder per ascoltare musica avendo a disposizione un'applicazione che occupi una limitata parte di memoria e sfrutti al minimo il processore del PC. Studiando nel dettaglio la funzione di encoding si osserva come esso riceva in input un segnale PCM e lo legga a blocchi di 384, 576 o 1152 campioni, in funzione del formato MPEG/Layer utilizzato. Per ciascun blocco l'encoder esegue le seguenti operazioni:

1. Vengono mutati i campioni PCM nel corrispettivo dominio frequenziale tramite il Banco di filtri polifasico seguito da una Trasfromata Coseno Modificata (il segnale passa dal dominio del tempo all'equivalente raffigurazione frequenziale).
2. Viene usato un modello psicoacustico per esaminare lo spettro del segnale e definire un livello di soglia di udibilità sfruttando i fondamenti acustici dell'apparato uditivo umano. Il modello assegna al quantizzatore l'informazione indicante quali sono le informazioni udibili e quali non.
3. Il quantizzatore codifica numericamente lo spettro in funzione dell'importanza di ciascuna banda frequenziale (se il modello psicoacustico dice che una certa banda viene poco avvertita, verrà codificata con pochi bit). Lo scopo finale è quello di avere una quantizzazione dello spettro tale per cui il rumore di quantizzazione introdotto si trovi al di sotto della soglia di udibilità data dal modello psicoacustico.
4. La codifica numerica dello spettro frequenziale generata dal quantizzatore viene impacchettata secondo la sintassi dello standard MPEG utilizzato.

Le operazioni svolte dal decoder, per i motivi spiegati in precedenza, sono meno numerose ed eseguono compiti inversi rispetto a quelle dell'encoder.

1. Effettua un'operazione di spacchettamento dei frame, leggendone le informazioni codificate ed estraendone lo spettro.
2. Attraverso il Banco dei filtri inverso prende lo spettro del segnale e genera i corrispondenti campioni PCM (384, 586, 1152) da dare in input all'hardware del PC.

L'MPEG-1 Layer 3

Qualsiasi encoder MPEG è in grado di comprimere un segnale PCM con diversi algoritmi di compressione. Lo standard MPEG consta di tre schemi di codifica (Layers) di complessità ed efficienza di compressione crescente. In tutti i casi, i dati codificati sono organizzati in trame (frame) decodificabili indipendentemente le une dalle altre:

- Layer 1: è l'algoritmo più semplice dei tre e raggiunge onesti risultati con un bit-rate pari a 384Kbit/sec per un segnale stereo. Esso associa ad un frame 384 campioni PCM. Il formato di file generato viene identificato con il nome di MP1.
- Layer 2: più articolato del primo in quanto ricollega ad un frame 1152 campioni PCM; è adatto per codifiche a bit-rate intorno ai 192-256Kbit/sec per un segnale stereo. Il formato di file creato è l'MP2.
- Layer 3: in questo Layer il formato MPEG-1 associa ad ogni frame 1152 campioni PCM mentre MPEG-2 ne associa solo 576, aumentando così la risoluzione temporale. Già con bit-rate tra 128-192kbit/sec si riesce ad ottenere un segnale stereo di qualità sufficientemente elevata.

Il Layer 3 a differenza dei primi due si avvale di tre concetti fondamentali che fanno di esso il Layer più complesso, ma anche quello che raggiunge le migliori prestazioni. In primis, vi è la presenza di un dominio frequenziale suddiviso in funzione delle bande critiche; in secondo luogo, si utilizza la codifica di Huffman per la funzione di impacchettamento finale dei dati audio e, per concludere, la tecnica del "Bit Reservoir" che permette di migliorare la qualità audio a parità di bit-rate.
Come è possibile intendere, il formato associato a questo Layer è il famoso MP3.
In un file MP3 le trame audio hanno una lunghezza variabile; ognuna di queste contiene tutte le informazioni indispensabili per la ricostruzione dei corrispettivi campioni PCM, in modo autonomo da tutto il resto del file. Ciò consente di rendere adoperabile questo formato anche nel campo dello streaming (es. web radio), in quanto la perdita di un certo numero di byte non danneggia la regolare decodifica del resto dell'informazione.

Se, per esempio, vengono smarriti i dati di un generale frame X, il decoder è comunque in capace di decodificare correttamente i rimanenti byte originando un silenzio in luogo del frame mancante.

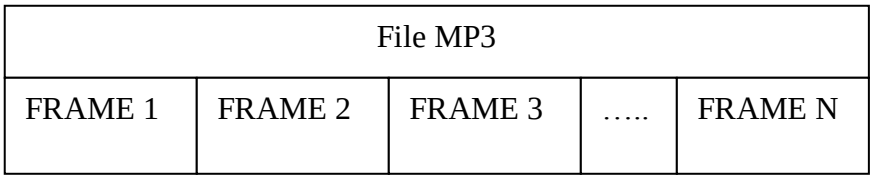

Figura 7. Struttura di uno streaming MP3

Le quattro parti basilari che rappresentano un frame sono: Header, CRC, Side Information, e Main Data e sono organizzate come mostrato in figura 8:

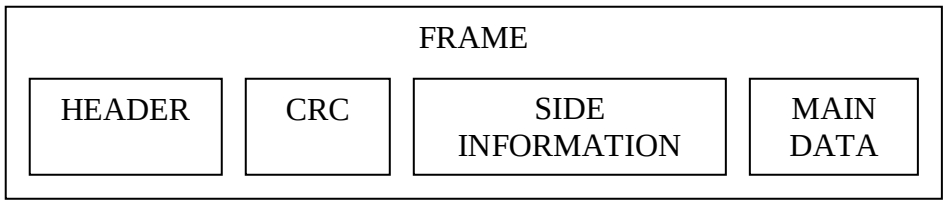

Figura 8. Struttura di un frame MP3

L'Header è un campo composto da 32 bit e contiene tutte le informazioni necessarie per la descrizione del frame. Informazioni come la versione MPEG, Layer,
bit-rate e frequenza di campionamento sono tutte contenute all'interno di esso.
Il CRC è un campo impiegato dai decoder per validare la correttezza del frame. Questa informazione ha un fondamentale rilievo nello streaming web, dove la possibilità di perdita di informazione è notevole.
Al seguito del campo Header e CRC esiste un campo chiamato Side Information. Esso contiene informazioni sulla decodifica generale del frame, ma non contiene gli effettivi campioni audio codificati, poiché quest'ultimi si trovano all'interno de campo Main Data.

CAPITOLO 2
ANALISI DELLE TECNOLOGIE

2.1 Il linguaggio XML

Il linguaggio XML, eXstensible Markup Language, contrariamente a quanto si pensa, non è un ennesimo linguaggio di marcatura, bensì un meta-linguaggio in grado di definire altri linguaggi di markup altamente specializzati e fortemente organizzati. XML è un linguaggio che permette l'interpretazione di dati semistrutturati. Con la parola, "dato semistrutturato", si intende una tipologia di dato contenente, all'interno di esso, le informazioni sulla sua struttura oltre al dato in se; questa tipologia di dati viene anche definita auto-descrittiva (self-describing) o senza schema (schemaless). [5]

L'utilizzo di dati semistrutturati consente di definire, attraverso un'unica sintassi, la struttura (schema) ed il valore (istanze) dei dati contemporaneamente; ed è proprio per questa sintesi informativa che i documenti XML vengono usufruiti per operazioni di trasferimento di dati.

Il vantaggio principale di un documento XML è la possibilità, attraverso un qualsiasi editor di testo, di esser scritto in formato testuale, un formato universale accettato da tutti i sistemi hardware e software.

Il "progetto XML" nasce alla fine degli anni novanta da una raccomandazione del W3C (Wold Wide Web Consortium), uno dei consorzi di aziende più importanti per lo sviluppo di standard web. Lo scopo del progetto era quello di produrre una versione semplificativa del linguaggio SGML (Standard Generalized Markup Language), in modo tale da offrire una maggiore libertà nella definizione dei tag, pur rimanendo nel dominio e nel rispetto di uno standard. Fu cosi che nel 1996 si costituì un gruppo di lavoro denominato XML Working Group, composto da esperti mondiali delle tecnologie SGML e da una commissione, XML Editorial Review Board, delegata alla redazione delle specifiche del progetto.

Generalmente l'XML viene confrontato con l'HTML (HyperText Markup Language), il linguaggio utilizzato per la creazione di pagine web; tuttavia i due linguaggi sono ben differenti. Mentre l'HTML definisce una grammatica atta alla descrizione e alla formattazione di pagine web, l'XML viene utilizzato per descrivere il contenuto di un documento. La difformità tra i due linguaggi è rilevante tanto che, l'HTML, può essere

esaminato come un sottoinsieme dell'XML. Le ultime specifiche di HTML sono ferme alla versione 4.0 del dicembre 1997; tutto ciò per dare spazio ad una nuovo linguaggio definito XHTML che fonde la scrittura dell'HTML alle regole dell'XML. Il W3C introduce XHTML per colmare il divario tra HTML e XML e, per far conoscere quest'ultimo a un numero più elevato di persone. XHTML in sostanza è un applicazione che simula HTML 4.0 in modo tale da poter visualizzare i risultati(veri documenti XML) nei browser web correnti.

2.2 La sintassi XML

Un documento XML è una rappresentazione testuale di informazioni identificate secondo regole sintattiche ben definite. Esso è composto da una struttura logica e da una struttura fisica collegate tra di loro intrinsecamente.

La struttura logica è contraddistinta da una ordine gerarchico ben definito; essa è composta da componenti denominati elementi, in grado di contenere al loro interno altri sottoelementi o porzioni di testo. Ogni singolo elemento rappresenta un componente logico del documento.

Ad ogni elemento è possibile associare degli attributi, ossia delle informazioni aggiuntive in grado di approfondire alcune proprietà specifiche dell'elemento interessato.

Per una corretta struttura logica è di vitale importanza che tutti gli elementi facciano capo ad un elemento genitore chiamato root element o semplicemente root o radice.

La struttura logica di un documento XML è raffigurabile tramite un albero rovesciato, generalmente noto come document tree. [6]

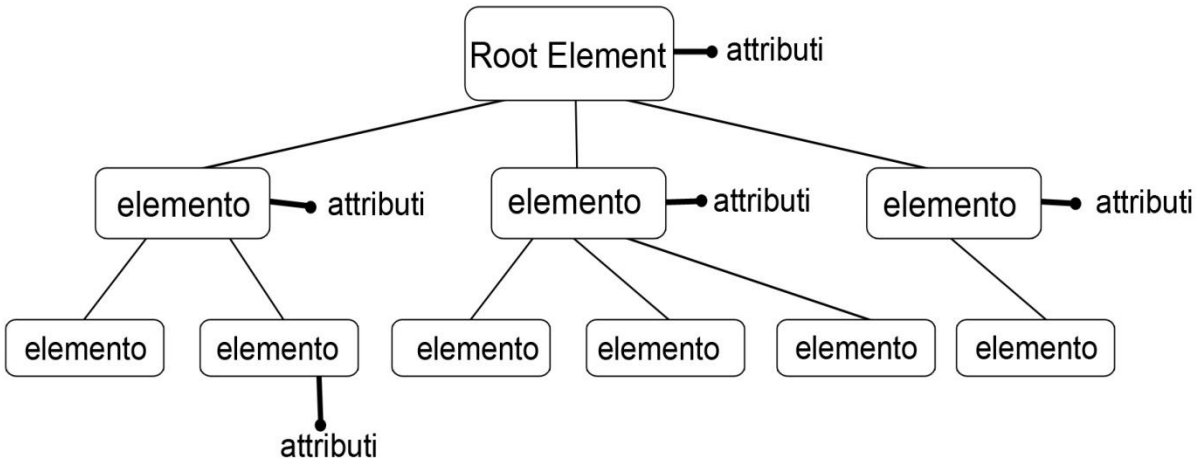

Figura 9. Struttura logica di un documento XML

Non essendoci regole prestabilite per l'organizzazione logica di un documento XML, la struttura logica dipenderà dalle scelte progettuali che l'utente è intenzionato a sviluppare; ciò vuol dire che l'organizzazione degli elementi all'interno del documento è a pura discrezione dell'autore.
La traduzione concreta della struttura logica viene chiamata struttura fisica, composta anch'essa da elementi sintattici qualificati come tag. La forma sintattica sottostante visualizza un esempio di struttura fisica di un documento XML, creata attraverso un semplice editor di testo.

```
<?xml version="1.0" ?>
<tesi titolo="Titolo della tesi">
   <capitolo titolo="Titolo del primo capitolo">
      <testo>
         Blocco di testo del primo capitolo
      </testo>
      <immagine file="immagine1.jpg">
      </immagine>
   </capitolo>
   <capitolo titolo="Titolo del secondo capitolo">
      <testo>
         Blocco di testo del secondo paragrafo
      </testo>
      <codice>
         Esempio di codice
      </codice>
   </capitolo>
   <capitolo tipo="bibliografia">
      <testo>
         Riferimento ad un testo
      </testo>
   </capitolo>
</tesi>
```

Il corpo testuale del documento è introdotto da una prima riga impiegata comunemente per identificare il documento come XML e specificandone la versione utilizzata(1.0);
in seguito vengono riprodotti gli elementi attraverso i tag, ovvero sequenze di caratteri delimitate dai simboli '<' e '>' proprio come avviene per l'HTML. A differenza dell'HTML, in cui i tag sono predefiniti, XML rende possibile la scelta dei tag da parte

dell'utente. L'assegnazione di un attributo ad un elemento si verifica inserendo il nome di quest'ultimo, con il relativo valore, all'interno del tag di apertura dell'elemento.
Alcuni elementi possono essere vuoti, cioè possono essere privi di contenuto testuale. Al contrario di quanto avviene per HTML, che autorizza l'utilizzo di elementi sprovvisti di tag di chiusura, XML presume che vengano sempre specificati i tag di apertura e di chiusura. E'questo il caso del tag immagine.
Tuttavia, XML prevede una sintassi abbreviata per gli elementi vuoti, più precisamente, la conclusione del tag di apertura con la sequenza di caratteri "/>" ammette l'assenza del tag di chiusura.

2.3 XML ben formati

I documenti XML sono soggetti a due vincoli specifici: devono essere ben formati(well formed) e validi(DTD). [7]
Secondo il consorzio W3C, un oggetto dati, per essere considerato un documento XML, deve sottostare al vincolo fondamentale di una corretta formazione. L'origine per cui il W3C impone questo vincolo, fondamentalmente, è per ostacolare i processori XML a comportarsi come i browser HTML quando interpretano il codice HTML, cercando, cioè, di risolvere i problemi sintattici autonomamente.
Perché un documento XML sia ben formato deve rispettare le seguenti regole:

- Tutti gli elementi di un documento XML devono sottostare ad un unico elemento di massimo livello (root). Le sole parti che possono rimanere esterne a quest'ultimo sono i commenti e le direttive di elaborazione.
- Ogni elemento deve avere un tag di apertura ed il corrispettivo tag di chiusura o, se vuoti, possono prevedere la forma abbreviata "/>".
- Gli elementi devono essere nidificati ordinatamente, ovvero, che i tag di chiusura seguano la disposizione inversa dei rispettivi tag di apertura.
- XML è case sensitive, ciò comporta una netta distinzione tra maiuscole e minuscole, per cui i nomi dei tag e degli attributi devono corrispondere nei tag di apertura e chiusura anche in relazione a questo aspetto.
- I valori degli attributi devono essere compresi tra singoli o doppi apici.

La violazione di una qualsiasi di queste regole comporterebbe una non corretta formazione della stesura sintattica.

I contenuti di un documento XML possono contenere qualsiasi carattere latino, cifre, punteggiature. Generalmente vengono accettati come validi i primi 128 caratteri della codifica ASCII (lettere maiuscole, minuscole, numeri, ecc), qualora un documento dovesse contenere caratteri non rientranti tra questi (simboli di valuta, lettere accentate, ecc) è necessario indicare all'interno delle direttive di elaborazione lo schema di codifica utilizzato.

Oltre alle direttive di elaborazione, in un documento XML, è possibile trovare i commenti, ovvero informazioni ignorate dal software ma visibili all'utente. I commenti XML, come per quelli dell'HTML, sono racchiusi tra le sequenze di caratteri `<!--` e `-->`
e possono essere presenti in qualunque posizione del documento.

Può capitare, a volte, che in un documento XML sia necessario inserire dei caratteri particolari, che potrebbero renderlo non ben formato. Ad esempio, i caratteri < e > non possono essere inseriti come valori poiché verrebbero confusi per elementi di apertura e chiusura di un tag.

Per ovviare a questo possibile conflitto è indispensabile utilizzare un codice, definito Entity- name , in sostituzione del carattere da inserire. La Figura 10 riporta vari Entity-name di alcuni caratteri particolari.

Entity-name	Carattere
`&amp;`	&
`&lt;`	<
`&gt;`	>
`&quot;`	"
`&apos;`	'

Figura 10. Esempi di Entity-name XML

In determinate circostanze gli elementi da sostituire con le entità possono essere tanti, il che implicherebbe una difficoltà nella lettura del testo. In questo caso, al posto di sostituire tutte le occorrenze dei simboli speciali con le corrispettive entità è possibile servirsi di una sezione CDATA. Una sezione CDATA (Character DATA) è un blocco di testo che viene considerato sempre come testo, anche se contiene codice XML o altri caratteri particolari. Per indicare una sezione CDATA si dovrà far uso della sequenza di caratteri `<![CDATA[` e `]]>`.

2.4 XML validi

Il capitolo precedente ha illustrato il ruolo fondamentale che ha la sintassi XML per la creazione di documenti ben formati. Il presente capitolo, al contrario, analizza la possibilità di associare una grammatica ad un documento XML, e quindi in un certo senso dell' opportunità di accostare una semantica ad un documento XML. Il processo con il quale viene controllata la correttezza strutturale e la correttezza sintattica di un documento XML viene definito validazione.
Le tecniche fondamentali utilizzate per definire la struttura di un documento XML sono:

- DTD (Document Type Definition)
- XSD (XML Schema Definition)

Da un punto di vista cronologico, il DTD rappresenta la prima tecnologia utilizzata per la definizione di strutture di documenti XML. Un DTD è un documento che delinea i tag usufruibili in un documento XML, il loro rapporto nei confronti della struttura del documento e ulteriori dati sugli attributi di ciascun tag.
La sintassi di un DTD viene impostata principalmente sulla presenza degli elementi e degli attributi. Per illustrare la struttura di un elemento XML all'interno di una DTD è fondamentale servirsi della parola chiave `<!ELEMENT>` attraverso la seguente dichiarazione:

```
<!ELEMENT  nome regola_DTD>
```

Dove `nome` è il nome dell'elemento XML(il tag) e la `regola_DTD` indica invece il tipo di contenuto che questo tag avrà ed eventualmente la sua relazione con altri contenuti delineati nella DTD. I valori assegnabili a una `regola_DTD` possono essere:

- EMPTY: specifica che il tag è composto solamente da attributi e non conterrà altri attributi al suo interno.
- PCDATA: specifica che il contenuto è esclusivamente di tipo testuale.
- ANY: specifica che il contenuto del tag è vuoto.

Può capitare che alcuni elementi contengano al loro interno uno o più sottoelementi, ad esempio la dichiarazione sottostante:

```
<!ELEMENT banca (clienti+) >
```

Indica che l'elemento <banca> ha come sottoelemento uno o più elementi <clienti>, mentre, il simbolo + specifica il numero di occorrenze appartenenti ad un elemento. Un insieme di caratteri speciali ha quindi lo scopo di indicare una molteplicità di occorrenze in un elemento. In particolare:

- + indica che l'elemento è presente da 1 a infinite volte.
- ? indica che l'elemento è presente zero o una volta.
- * indica che l'elemento e presente da zero a infinite volte.

Ad esempio, per determinare che l'elemento <banca> è costituito da un numero arbitrario ed eventualmente nullo di elementi <clienti> e <depositi> si adotta la seguente dichiarazione DTD:

```
<!ELEMENT  banca (clienti* , depositi*) >
```

Per definire un attributo XML è, al contrario, indispensabile l'utilizzo del termine <!ATTLIST> all'interno della seguente sintassi DTD:

```
<!ATTLIST  elemento nome_attributo  tipo  valore_default >
```

Dove elemento è il nome del tag contenente l'attributo, nome_attributo è il nominativo dell'attributo, tipo è la specifica della definizione del tipo di attributo e valore_default è un potenziale valore prestabilito. [8]
Il valore predefinito in una dichiarazione di attributo denota un valore reale o la necessità di assegnare un valore all'attributo nel documento XML. I valori predefiniti di un attributo sono i seguenti:

- REQUIRED: specifica che l'attributo deve essere reso visibile nel documento XML altrimenti si verificherà un errore di analisi.
- IMPLIED: specifica che l'attributo può essere presente nel documento XML ma, se tralasciato, non si riscontrerà errore di analisi.
- FIXED: specifica che l'attributo mostra un valore fisso nella DTD e non può essere variato o sovrascritto nel documento XML.

L'impiego dei DTD per definire la grammatica di un linguaggio di marcatura non sempre risulta appagante. Sono molte le cause che hanno portato alla definizione di metodi alternativi per comporre grammatiche per documenti XML; dalla disuguaglianza di regole nella sintassi DTD rispetto a quella XML, alla inefficace possibilità di precisare un tipo di dato per il valore di attributi, fino ad arrivare alla mancata opportunità di indicare il numero massimo e minimo di occorrenze di un tag in un documento.
Tutte queste limitazioni hanno reso possibile la creazione dell'XML Schema.
Un XML Schema è un documento XML che, attraverso l'utilizzo di un framework molto più rigoroso, si adopera per definire in modo più opportuno la struttura e il contenuto di un documento XML. Al di là di delle proprietà basilari per la gestione di elementi e attributi, gli Schema consentono agli sviluppatori di applicazioni di implementare specifiche limitazioni sulle tipologie di dati e sul contenuto dei documenti che ne fanno uso.
La creazione di uno Schema, ha come vincolo principale, quello di dover rispettare l'uso dei tag definiti dall'XML Schema Language definito dal W3C. Più precisamente un XML Schema presenta la seguente struttura generale:

```
<?xml version"1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
... descrizione della grammatica ...
</xs:schema>
```

Il tag `<xs:schema>` descrive l'elemento root, il quale specifica attraverso l'attributo `xmlns` che nel documento verranno impiegati dei tag definiti da uno determinato standard del W3C.
La principale novità importata dagli XML Schema rispetto ai DTD è data dall'opportunità di poter personalizzare un tipo di dato. I tipi di dato si possono classificare in due categorie: semplici e complesse. I dati semplici riguardano quegli elementi che non contengono altri elementi e non prevedono attributi. Al contrario i dati complessi sono relativi a quegli elementi con attributi e che sopportano la presenza di altri elementi. La spiegazione del tipo complesso consiste il più delle volte nella descrizione della struttura prevista dall'elemento. Se un elemento è capace di contenere all'interno di esso altri elementi, è possibile definire questi ultimi non più come elementi, ma come sequenza, gruppo o insieme di valori relativi.

2.5 Il software Flash

Flash è uno strumento di creazione che consente agli sviluppatori web di creare presentazioni, applicazioni e altri tipi di contenuto che consentano l'interazione degli utenti.

La sua unicità deriva dal fatto che mette a disposizione svariate funzionalità quali, disegnare, animare comunicare direttamente con Javascript ed integrare video e contenuti multimediali. [9]

Il largo utilizzo di grafica vettoriale in Flash rende questo strumento particolarmente adatto alla realizzazione di contenuto da includere all'interno di pagine web, in quanto la dimensione finale del file sarà contenuta; visualizzando una qualsiasi pagina internet è possibile individuare un esempio, più o meno vistoso, dell'utilizzo di questa tecnologia; basti pensare alle molteplici pubblicità che vengono proposte in rete, dai banner più piccoli alle animazioni più complesse.

Da notare che queste animazioni affiancano ad un primo aspetto puramente grafico, un secondo che è legato esclusivamente alla programmazione. Infatti, in Flash, la quasi totalità degli oggetti facenti parte di un'applicazione può essere accompagnata da codice che ne gestisce il comportamento nelle situazioni richieste dal programmatore. Il linguaggio utilizzato in Flash è Actionscript.

La nascita di Flash è da ricondursi al 1996 quando Macromedia acquisisce un software per animazioni vettoriali chiamato FutureSplash , per poi commercializzarlo sotto il nome di Flash 1.

Lo sviluppo sotto la casa Macromedia prosegue fino al 2005 e durante quel periodo vengono distribuite le seguenti versioni:

- Flash 2 (1997)
- Flash 3 (1998)
- Flash 4 (1999)
- Flash 5 (2000)
- Flash MX (2002)
- Flash MX 2004 e Flash MX 2004 Professional (2003)
- Flash Basic 8 e Flash Professional 8 (2005)

Successivamente, nel dicembre 2005, la società Adobe acquisisce interamente Macromedia e i suoi prodotti. Adobe inizia cosi il suo personale sviluppo di Flash e ne commercializza ad oggi le seguenti versioni:

- Flash Professional CS3 (2007)
- Flash Professional CS4 (2008)
- Flash Professional CS5 (2010) e Flash Professional CS5.5 (2011)

L'utilizzo di Flash avviene tramite diversi tipi di file, ognuno con proprietà e funzioni specifiche. I principali tipi di file utilizzati sono:

- i file FLA: sono i principali file elaborati da Flash e contengono i contenuti multimediali di base, la linea temporale e le informazioni relative agli script dei documenti Flash;
- i file SWF: sono la versione compressa dei file FLA e vengono visualizzati nella pagina web o nel Flash player; costituiscono il prodotto finale di Flash;
- i file AS: sono il file Actionscript che possono essere usati se si desidera tenere tutto o solo una parte del codice Actionscript al di fuori del file FLA; sono utili per l'organizzazione del codice e per progetti in cui più persone lavorano a parti diverse del contenuto Flash.

2.6 Il linguaggio Actionscript

Se i testi, le forme e gli strumenti di disegno sono individuabili ed usabili perché comuni a molti programmi di grafica, la sezione per l'interattività sembra più ostica. Le azioni come premere un pulsante per andare a una certa pagina o inviare una mail o fermare l'avanzamento del Flash sono possibili grazie al linguaggio di programmazione Actionscript.

Actionscript è un linguaggio di scripting, utilizzato principalmente per lo sviluppo di siti web e di software che utilizzano la piattaforma Adobe Flash Player, (sotto forma di file SWF incorporati in pagine web) ma viene anche usato in alcune applicazioni di database e in robotica di base, come con il Make Controller Kit.

AS è un linguaggio di alto livello, al contrario di assembly, è un linguaggio interpretato e non compilato, come ad esempio lo sono il C++ e il Visual Basic, inoltre, riesce a far interagire tra loro due oggetti, questo perché si basa su ECMAscript(un linguaggio molto simile al Javascript).

Il primo linguaggio di interattività inventato dagli sviluppatori Flash fu chiamato con il nome di "Action"; questo semplice codice permetteva solamente di gestire limitati movimenti come possono essere il "play", "stop", "gotoAndPlay", "getURL".
Con la creazione della versione Flash 4 del 1999, questi semplici comandi diventarono un vero e proprio linguaggio di scripting. Vennero infatti introdotte nuove funzionalità come le variabili, operazioni, costrutti if e cicli. Sebbene questo linguaggio veniva già chiamato internamente dagli sviluppatori con il termine "Actionscript", per una questione di marketing si è continuato ad usare la parola "Action".

2.6.1 Connubio tra Actionscript e Flash Player

Flash Player è il lettore che mette a disposizione gratuitamente la casa Adobe per leggere e visualizzare gli oggetti dinamici creati in Flash. Solitamente il player viene fornito in versione stand alone (capace di funzionare da solo o in maniera indipendente da altri oggetti o software, con cui potrebbe altrimenti interagire) o come per i browser.
Una negatività però di questo sistema è che la privacy dell'utilizzatore può essere a rischio poiché può immagazzinare sul suo hard disk file .sol che funzionano come cookie e che potrebbero non essere riconosciuti dai più comuni software anti-malware qualora il loro uso sia maligno.
Nello schema sottostante vengono analizzate le varie versioni di Flash Player e la contemporanea evoluzione del linguaggio AS:

- Flash Player 2: Fu la prima versione con il supporto scripting; il linguaggio "Action"includeva le azioni gotoAndPlay, gotoAndStop, nextFrame, nextScene.
- Flash Player 3: Venne ampliato il supporto di scripting, con la possibilità di caricare file esterni SWF(loadMovie).
- Flash Player 4: Primo player contenente l'intero linguaggio scripting "Action". Esso conteneva non più le azioni basilari, ma anche quelle per i costrutti if, operazioni condizionali, cicli.
- Flash Player 5: Questa versione introduce la prima versione dell'Actionscript.
- Flash Player 6: In essa viene incluso il primo supporto per la visione e l'ascolto di file in streaming.
- Flash Player 7: Vengono aggiunti i formati CSS per i testi e inoltre racchiude la prima versione di Actionscript 2.0.

- Flash Player 8: Versione ulteriormente ampliata con l'aggiunta di nuove librerie.
- Flash Player 9: Questa è la prima versione che adotta ActionScript 3.0. In essa viene aggiunta una nuova virtual machine chiamata VM2(Actionscript Vistual Machine 2)
Dopo anni di possesso da parte della casa Macromedia,questa è la prima versione chiamata con il nome Adobe.
- Flash Player 10: La versione 10 è la prima ad introdurre il movimento 3D, sfruttando l'uso degli assi X, Y, Z. Inoltre vi è un nuovo suono API che consente la creazione personalizzata di audio in Flash, cosa che non è mai stata possibile prima.
- Flash Player 11: Ultima versione creata, basata soprattutto per i dispositivi mobili e smartphone.

2.6.2 Actionscript 2.0 vs Actionscript 3.0

Con il passare del tempo, tutto prima o poi è destinato a cambiare o ad evolversi. Questo concetto si rispecchia nel passaggio dall'Actionscript 2.0 alla versione 3.0.
Piuttosto che cercare di migliorare la AS 2.0 con l'aggiunta di nuove funzionalità modificando di conseguenza le prestazioni, Adobe ha deciso di accantonare AS 2.0, per creare la versione 3.0.
Il software risultante, anche se ancora molto simile ad AS 2.0, ha un sacco di cambiamenti, il più celebre è l'aumento di velocità. La percezione generale è che AS 3.0 è di circa 10 /15 volte più veloce in esecuzione rispetto alla versione precedente.
Un altro cambiamento in AS 3.0 è la stretta aderenza alle buone pratiche di codifica.
Con AS 2.0, smussando un po'gli angoli, era possibile facilitare e velocizzare le azioni di codifica. Questo non è più possibile in AS 3.0 poiché le variabili globali sono state ridotte al minimo, costringendo le persone ad utilizzare OOP (Object Oriented Programming), che può apparire più pulito e più facile da eseguire il debug, ma è certamente più dispendioso in termini di codice.
Il principale problema che molti utenti hanno con AS 3.0 sta proprio nell'apprendimento di quest'ultimo. E'più difficile per i neofiti imparare un linguaggio più evoluto rispetto ad una versione base. Nonostante ciò, i neofiti sono incoraggiati ad imparare AS 3.0 rispetto ad AS 2.0 in quanto le aziende, in un prossimo futuro, probabilmente inizieranno a migrare verso questo nuovo linguaggio.

CAPITOLO 3
IL MEDIA PLAYER

3.1 Creazione di un elenco

In questa prima parte del progetto, vengono analizzate le basi per la creazione di un player MP3, attraverso l'utilizzo del programma Adobe Flash CS5. All'origine del programma, vi è la creazione di una componente list, ossia un elenco a scorrimento con possibilità di selezione singola o multipla. Per aggiungere voci all'elenco, la componente list utilizza due metodi: `List.addItem()` e `List.addItemAt()`. L'indice di partenza di una componente list ha sempre valore 0, in cui la voce con indice 0 è la prima voce visualizzata.

La creazione di un elenco all'interno di una list, prevede inizialmente l' inizializzazione di una variabile.

Le variabili consentono di memorizzare valori da utilizzare in un programma. Per dichiarare una variabile, è necessario utilizzare l'istruzione `var` nel suo nome. A differenza dell' Actionscript 2.0, che prevede l'uso dell'istruzione `var` solo se si utilizzano le annotazioni di tipo, l'Actionscript 3.0 utilizza quest'ultima istruzione per qualsiasi tipo di annotazione. Se si omette l'istruzione `var` nella dichiarazione di una variabile, viene generato un errore del compilatore in modalità rigorosa e un errore runtime in modalità standard.

L'associazione di una variabile a un tipo di dati deve essere effettuata al momento di dichiarare la variabile. La dichiarazione di una variabile senza indicazione del tipo è consentita, ma genera un' avvertenza del compilatore in modalità rigorosa. Per designare il tipo di una variabile si aggiunge al nome, un carattere di due punti(`:`) seguito dal tipo. Nel programma sarà presente quindi una variabile denominata `i` del tipo `int`:

```
var i:int;
```

Successivamente all'inizializzazione della variabile, il programma prevede un' istruzione di ripetizione ciclica `for`. L'istruzione di ripetizione ciclica permette di eseguire ripetutamente un blocco di codice specifico utilizzando una serie di valori o variabili. In una ripetizione ciclica, solitamente, il codice viene immesso all'interno delle parentesi graffe; seppur questo si può evitare, è preferibile avvalersi di quest'ultime poiché aumenta la possibilità che le istruzioni aggiunte in futuro vengano inavvertitamente escluse dal

blocco del codice. Se, in un secondo momento, si aggiunge un' istruzione da includere nel blocco di codice, ma si dimentica di aggiungere anche le necessarie parentesi graffe, l'istruzione verrà ignorata al momento dell'esecuzione del ciclo.
Il ciclo `for` permette l'esecuzione di iterazioni su una variabile per verificare un intervallo di valori specifico. A un' istruzione `for` è necessario fornire tre espressioni: una variabile impostata su un valore iniziale, un' istruzione condizionale che determina quando il ciclo termina e un' espressione che cambia il valore della variabile a ogni ciclo.
Nel programma comparirà quindi il seguente codice di programmazione:

```
var i:int;
for ( i=1 ; < 10 ; i++){
  list.additem ({label : " song " +i});
}
```

Il ciclo svolge 9 iterazioni. Il valore della variabile `i` inizia a 1 e termina a 9 e l'output è rappresentato dai numeri da 1 a 9, ciascuno su una riga separata. I valori delle iterazioni possono essere sostituiti dall'utente in qualsiasi momento, secondo le sue esigenze.
L'istanza `label`, visualizza una o più righe di testo semplice; nel suddetto caso verrà visualizzata la parola `song` all'interno delle celle, viceversa, `+i`, rappresenta l'indice del loop, mediante la quale il player comprende il termine del ciclo ed intraprende una nuova riproduzione.
La formattazione di un contenuto testuale è assegnata alla classe `Textformat`, più precisamente, rappresenta le informazioni relative alla formattazione dei caratteri. Per creare un oggetto `TextFormat`, prima di impostarne le proprietà (colore0xFFFFFF), è indispensabile la funzione di costruzione `new TextFormat()`. Utimata la creazione di questo oggetto, per renderla eseguibile, viene usufruita la classe `CellRenderer`. `CellRenderer` è una classe con componenti basati su list, utilizzata per gestire e visualizzare il contenuto delle celle personalizzate dell'elenco. La classe `CellRenderer` comprende numerosi stili per il controllo del formato delle celle. La formattazione degli stili usati da `CellRenderer` avviene chiamando il metodo `setRendererStyle()` dell'oggetto list.

La formattazione testuale del contenuto all'interno delle celle, sarà definita dal seguente codice di programmazione.

```
var myFormat:TextFormat = new TextFormat();
myFormat.color = "0xFFFFFF";
list.setRendererStyle("textFormat", myFormat);
```

3.2 La playlist XML

La seconda parte del progetto, prevede la creazione di un file XML mediante l'utilizzo di un altro programma facente parte del pacchetto Adobe: Dreamweaver, e l'inserimento di quest'ultimo all'interno del nostro player, in modo tale da crearne la playlist. [10]
Il documento XML prevede per ogni canzone inserita all'interno del nostro elenco, due sottoelementi `<songTitle>` e `<songArtist>`, entrambi facenti parte dell'elemento radice `<Song>`. Il primo individua all'interno del file il titolo del brano da riprodurre,
il secondo, invece, specifica il nome dell'artista che esegue il pezzo. [11]
Attraverso l'uso di un documento XML è possibile inserire all'interno della playlist un numero svariato di canzoni, ma è di fondamentale importanza che i sottoelementi e i file concreti abbiamo una sintassi equivalente.
Una volta creata la playlist virtuale, si è in grado di inserirla al'interno del player. Per fare ciò è indispensabile inizializzare le seguenti variabili:

```
var trackToPlay:String;
var i:uint;
var pausePosition:int = 0;
```

Dove la variabile indicizzata come `trackToPlay` è una variabile di tipo stringa impiegata per la rappresentazione di dati di tipo testuale . Quando si vuole rappresentare un numero intero senza segno, vale a dire un numero intero non negativo si utilizza l'indice `uint`, il tipo di dato `uint` è memorizzato internamente come numero intero a 32 bit e
comprende la serie di numeri interi da 0 a 4.294.967.295 (232 - 1). La variabile `pause Position`, infine, settata a 0 sta ad indicare che il lettore è in attesa di una riproduzione musicale. Al fine di caricare il file XML esterno è obbligatorio eseguire le seguenti operazioni:

1. Creare una variabile per contenere un'istanza della classe *XML*

2. Creare un'istanza della classe `URLLoader` per caricare il file XML
3. Passare il contenuto del file XML per la variabile di istanza `XML` una volta che il file ha completato il caricamento.

La creazione di una variabile che contiene un'istanza della classe *`XML`* può essere fatto facilmente utilizzando l'operatore `var`:

```
var myXML: XML; = new XML();
```

L'operatore due punti (`:`) viene utilizzato per bloccare il tipo della variabile in XML. Ciò significa che questa variabile non conterrà nessun altro dato se non in XML. Il passaggio di un qualsiasi altro tipo di dato diverso da XML creerà un errore di compilazione. [12]
Il passo successivo consiste nell'assegnare un identificativo (`XML_URL`) alla variabile che specifica il nome del file XML, per poi creare successivamente un'istanza della classe `URLLoader` capace di caricare quest'ultimi.

```
var XML_URL:String = "mp3_playlist.xml";
var myXMLURL:URLRequest = new URLRequest(XML_URL);
var myLoader:URLLoader = new URLLoader(myXMLURL);
```

La classe `URLLoader` è la classe responsabile del caricamento di tutti i dati binari e testuali ed ha bisogno dell'URL da passare come un'istanza della classe `URLRequest`. Questo secondo oggetto contiene le informazioni sulla risorsa da caricare e sui dati. Tuttavia, per elaborare il file XML, è necessario che esso sia stato completamente caricato, per questo motivo bisogna creare una `Listener` capace di verificare il processo di caricamento per poi completare l'elaborazione dei contenuti. `Listener` è da attribuire all'istanza della classe `URLLoader` nel modo illustrato di seguito.

```
myLoader.addEventListener("complete", xmlLoaded);
```

Il codice soprastante indica la forma per attivare la funzione `Listener`, ma non sono stati specificati i contenuti della presente. E'il caso di prevedere una "reazione" del player al caricamento del file XML. [13]

Questa reazione sarà contenuta all'interno di una funzione:

```
function xmlLoaded(event:Event):void
myXML = XML(myLoader.data);
```

Questa funzione prende il nome di `xmlLoaded` e per funzionare ha bisogno di ricevere un argomento che è definito tra parentesi tonde, si chiama `event` ed è di tipo `Event`. In AS 3.0 la logica di programmazione è fortemente strutturata ad eventi. Invece di scrivere il codice nell'ordine in cui eseguirlo, si preferisce creare dei comportamenti ed associarli a degli eventi. In seguito all'inserimento dei dati XML nel player, l'ordine di riproduzione e di visualizzazione dei brani viene stabilito attraverso la sintassi:

```
var firstSong:String = myXML..Song.songTitle[0];
var firstArtist:String = myXML..Song.songArtist[0];
songURL = new URLRequest("mp3_files/" + firstSong + ".mp3");
```

Inizialmente viene segnalata la traccia di default, con la quale il lettore deve cominciare l'esecuzione: nel caso in cui si voglia partire da una traccia differente, basta modificare il numero della canzone di riferimento all'interno delle parentesi `[]`.
Per poter eseguire tutte le canzoni del documento XML è strettamente necessario servirsi del `for each..in`, ActionScript 3.0 include l'istruzione `for each..in` per eseguire iterazioni sugli oggetti XML.

```
for each (var Song:XML in myXML..Song) {
    i++;
    var songTitle:String = Song.songTitle.toString();
    var songArtist:String = Song.songArtist.toString();
    list.addItem( { label: i+". "+songTitle+" - "+songArtist, songString:
    songTitle, Artist: songArtist, songNum: i } );
}
```

Così come per il semplice ciclo `for`, anche per il ciclo `for each.in`, l'istruzione `i++` determina un incremento della variabile ogni qualvolta quest'ultima termina il suo ciclo; in questo modo si evita la numerazione delle tracce nel file XML. Per concludere l'inserimento della playlist all'interno del player, è preferibile a fini prettamente estetici che le canzoni dell'elenco siano evidenziate una volta in riproduzione;

ciò comporta la creazione di una variabile denominata `myArray`.

```
var myArray = new Array (0,0);
list.selectedIndices = myArray;
```

All'apertura del player, la traccia uno sarà evidenziata di default. Le canzoni successive verranno evidenziate una volta cliccate.

3.3 Riproduzione del file

La terza parte del progetto illustra le specifiche che il player deve adottare per riprodurre concretamente un brano una volta caricato. Verranno creati due ulteriori frame oltre a quello di caricamento visto nella parte precedente. Il frame 2 è utilizzato esclusivamente per indicare la canzone da estrarre dalla cartella una volta cliccata una cella.

```
songURL = new URLRequest("mp3_files/" + trackToPlay + ".mp3");
```

Il frame 3 contiene la parte di codice atta ad intraprendere l'esecuzione del brano. [14]
Tutti i 3 frame presentano la stessa timeline, ciò comporta che quando l'applicazione dà inizio all'esecuzione, verrà intrapreso il processo di caricamento (frame 1) il quale, una volta terminato, passerà automaticamente al frame 3 evitando il frame 2. Questo avviene perché, in precedenza, è stato assegnato al frame 1 il comando `gotoAndStop(3)`, ossia l'ordine di eseguire una volta concluso l'upload, la canzone di default. Da questo punto in poi, il player non tornerà mai più sul frame 1, bensì intraprenderà un continuo interscambio tra frame 2 e frame 3 ogni volta che una traccia termina o ne viene cliccata una differente.
Quando si lavora con l'audio in ActionScript, è probabile che vengano utilizzate diverse classi del pacchetto Flash. La classe `Sound` è la classe utilizzata per ottenere l'accesso alle informazioni sonore; essa ha il compito di caricare un file audio e iniziarne la riproduzione. Una volta iniziata la riproduzione di un suono, Flash fornisce l'accesso a un oggetto `SoundChannel`. Dal momento che un file audio caricato potrebbe essere un suono tra i tanti che vengono riprodotti sul computer di un utente, ogni singolo suono che viene riprodotto utilizza il proprio oggetto `SoundChannel`; ciò che viene riprodotto dagli altoparlanti del computer è l'output combinato di tutti gli oggetti `SoundChannel` combinati. Questa istanza `SoundChannel` viene utilizzata per controllare le proprietà del suono e per interromperne la riproduzione. Infine, se si desidera controllare l'audio combinato, la classe `SoundMixer`

fornisce il controllo sull'output combinato.
La riproduzione dell'audio caricato può consistere semplicemente nella chiamata al metodo `Sound.play()` di un oggetto `Sound`, come indicato di seguito:

```
var snd:Sound = new Sound();
var channel:SoundChannel;
var context:SoundLoaderContext = new SoundLoaderContext(5000, true);
snd.load(songURL, context);
channel = snd.play(pausePosition);
```

La classe `SoundLoaderContext` fornisce le verifiche di sicurezza per i file che caricano l'audio. Gli oggetti `SoundLoaderContext` vengono passati come argomenti alla funzione di costruzione e al metodo `load()` della classe `Sound`. Il primo valore 5000 indica la proprietà `bufferTime`, ovvero il numero di millisecondi per pre-caricare l'audio in streaming in un buffer prima dell'avvio dello streaming; il secondo parametro `true`, invece, indicizza la proprietà `checkPolicyFile`. Solitamente viene impostato il valore `true` quando viene caricato un file audio dall'esterno del dominio del file che esegue la chiamata e il codice nel file che esegue la chiamata richiede un accesso di basso livello ai dati audio. Il metodo `load`, infine, sarà il responsabile del caricamento del file esterno indicato dall'URL di riferimento.
Anche in questo caso è presente un metodo `list.addEventListener` capace di specificare l'evento da eseguire una volta che viene cliccata una canzone differente all'interno della list.

```
list.addEventListener(Event.CHANGE, itemClick);
function itemClick (event:Event):void {
         channel.stop();
         trackToPlay = event.target.selectedItem.songString;
         gotoAndPlay(2); }
```

L'evento da compiere è chiamato con il nome personalizzato `item Click` e dovrà essere in grado di fermare la canzone in esecuzione per poi riprodurre il nuovo brano scelto dall'utente. Per eseguire questa operazione avverrà un eventuale trasferimento nel frame 2 del player, per il quale viene intrapreso il passaggio descritto all'inizio del capitolo, secondo la quale nel frame 2 viene estratto dalla cartella il brano indicato dal fruitore.

3.4 L'equalizzatore

La creazione di questo elemento si suddivide in due parti ben distinte. La prima, introduce la creazione dell'equalizzatore da un punto di vista grafico attraverso la realizzazione delle singole barre che compongono quest'ultimo; la seconda, invece, riguarda l'animazione delle singole barre, il che contempla uno sviluppo di codice di programmazione. [15]
L'equalizzatore è stato concepito come un insieme di 6 barre singole che, a loro volta, sono state costruite attraverso la creazione di 10 rettangoli, come si può vedere nella foto sottostante.

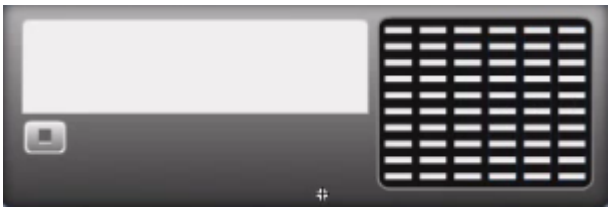

Figura 11. L'equalizzatore

Per effettuare l'animazione di ogni singola barra è di fondamentale importanza che vengano convertite come simboli d'animazione. L'animazione delle barre prevede una configurazione della timeline, ovvero quanti rettangoli devono essere visualizzati in ogni istante di tempo(frame). La configurazione dell'equalizzatore sarà data da 1 singolo rettangolo per ogni millisecondo, fino ad arrivare a un massimo di 10 millisecondi che corrispondono a 10 rettangoli.
Il susseguirsi del movimento delle barre è dovuto alla configurazione di un evento chiamato `onEnterFrame`. Esso ha il compito di ricreare un loop del frame in base alle indicazioni settate dall'utente, i cosiddetti Fps (frame per second).

```
addEventListener(Event.ENTER_FRAME, onEnterFrame);
function onEnterFrame(event:Event)
```

La programmazione delle singole barre è assegnata a una classe specifica definita `Math`.
La classe `Math` contiene i metodi e le costanti che rappresentano le funzioni matematiche e i valori comuni. Tutte le proprietà e i metodi della classe `Math` sono statici e per chiamarli è necessario utilizzare la sintassi `Math.method`.
Il metodo utilizzato per l'animazione delle barre è `round()`.

```
eqBarsx1.gotoAndStop (Math.round(channel.leftPeak * 10) );
eqBardx1.gotoAndStop (Math.round(channel.rightPeak * 8) );
eqBarsx2.gotoAndStop (Math.round(channel.leftPeak * 7) );
eqBardx2.gotoAndStop (Math.round(channel.rightPeak * 7) );
eqBarsx3.gotoAndStop (Math.round(channel.leftPeak * 8) );
eqBardx3.gotoAndStop (Math.round(channel.rightPeak * 10) );
```

Il metodo `round` arrotonda il valore del parametro `leftPeak o rightPeak` al numero intero più vicino per eccesso o per difetto e restituisce il valore finale. Il numero di rettangoli visibili in ciascuna barra è a completa discrezione dell'utente.

3.5 Regolazione del volume

Per la creazione dell'elemento grafico dello slider del volume, vale lo stesso identico concetto visto precedentemente per l'equalizzatore. Sostanzialmente l'elemento di regolazione del volume viene visto come elemento unico, ma in realtà è un insieme di singoli elementi realizzati ognuno indipendentemente.

L'elemento è creato da un rettangolo generale di 100 pixel che rappresenta la base sulla quale viene sviluppato l'intero componente. Uno slider di 10 pixel che, una volta cliccato, permette di abbassare e alzare il volume a discrezione dell'utente e infine da 33 rettangoli da 3 pixel ciascuno.

Il set iniziale della visualizzazione grafica viene definita dall'utente secondo un ordine logico; sapendo che il rettangolo totale è di 100 pixel è consigliabile posizionale lo slider in una posizione di offsett di 90 pixel. Lo stesso suggerimento vale per l'illuminazione del primo rettangolo che comunemente viene inserito nella posizione opposta allo slider. [16]

Questi parametri iniziali possono essere settati definitivamente secondo il codice:

```
volumeLightBar.width = knob.x;
```

I parametri `.widht` e `.x` corrispondono alle coordinate specifiche da intraprendere all'interno del rettangolo.

Il movimento laterale dello slider prevede che ad esso vengano imposti determinati limiti dimensionali in cui svolgere la sua funzione.

Questi limiti corrispondo sostanzialmente alle dimensioni del rettangolo iniziale.

```
var knobWidth:Number = knob.width;
```

```
var rectWidth:Number = rect.width;
var rectX:Number = rect.x
var boundWidth = rectWidth - knobWidth;
var boundsRect:Rectangle = new Rectangle(rectX, 0, boundWidth, 0);
```

Impostati i limiti di movimento è possibile ricreare il movimento concreto del cursore. In Actionscript 3.0 per avviare e fermare il trascinamento, vengono utilizzati gli eventi `MouseEvent.MOUSE_DOWN` (sul cursore) e `MouseEvent.MOUSE_UP` (sullo stage)
associati ai metodi `start drag` e `stop drag`. Il metodo `star drag` viene richiamato quando l'utente preme(`MOUSE_DOWN`) sull'istanza movie clip; in questo modo si è capaci di iniziare a trascinare il cursore. Per disabilitare l'esecuzione viene utilizzato il metodo `stop drag`. E'possibile richiamare questo metodo ogni volta che l'utente rilascia(`MOUSE_UP`) il pulsante del mouse, in modo tale da interrompere il trascinamento contemporaneamente al rilascio del pulsante del mouse.
In questo linguaggio di programmazione il trascinamento della MovieClip viene controllato passando un' istanza della classe `Rectangle` (inizializzato in precedenza) al nostro metodo `start Drag`, ciò vuol dire che il metodo stesso accetta solo 2 parametri:

- lock center: valore booleano che specifica, mentre la MovieClip viene spostata, se il suo centro è il centro del mouse (`true`) oppure se il centro è il punto dove l'utente esegue il click (`false`).
- bounds: consente di creare un contorno rettangolare che restringe l'azione di trascinamento dell'oggetto.

```
knob.addEventListener(MouseEvent.MOUSE_DOWN, Drag);
function Drag(event:MouseEvent) {
        knob.startDrag(false, boundsRect);
}
stage.addEventListener(MouseEvent.MOUSE_UP, Drop);
function Drop(event:MouseEvent) {
        knob.stopDrag();
}
```

Per la regolazione del volume Actionscript 3.0 utilizza una classe chiamata `Soundtrasform`; questa classe è responsabile dell'alterazione del volume e del panning del suono (manipolare l'equilibrio tra i diffusori sinistro e destro).

Il settaggio del volume, però, è affidato alla proprietà `volume` della classe `Soundtrasform`.
La proprietà `volume` ha la caratteristica di accettare esclusivamente valori compresi tra 0 (audio disattivato) e 1(audio al massimo).

```
var volumeLevel = channel.soundTransform;
var newLevel:Number = (volumeSlider.knob.x)/100 ;
volumeLevel.volume = newLevel;
channel.soundTransform = volumeLevel;
volumeSlider.volumeLightBar.width = volumeSlider.knob.x;
```

I valori del volume vengono associati alle misure adottate nel realizzare il set per la regolazione, in questo modo è possibile notare come il volume del suono venga immediatamente aggiornato basandosi sulla posizione *x* del cursore (i valori per posizionare il cursore vanno infatti da 0 a 100 in coincidenza della barra di 100 pixel).
Si evidenzi come il valore *x* recante la posizione del cursore venga diviso per 100, questo perché come specificato in precedenza il range di valori per il volume in Actionscript 3.0 va da 0 a 1.

3.5 Comandi di riproduzione

Questa sezione del progetto rende comprensibile l'utilizzo dei comandi comunemente utilizzati per svolgere le funzioni basilari di un player. Inizialmente, però, verrà analizzato il modo in cui le tracce cambiano volontariamente al termine di ciascuna e in particolare, il procedimento secondo il quale, una volta arrivato all'ultima traccia, il player tornerà automaticamente alla traccia iniziale. [17]
Il cambio di canzone automatico prevede l'impiego dei controlli condizionali `if` e `else`.
L'istruzione condizionale `if..else` permette di eseguire un blocco di codice se la condizione esaminata risulta verificata, al contrario verrà eseguito un blocco di codice differente.

```
function newTrack():void {
      if (list.selectedItem.songNum == i) {
          channel.stop();
          var selectFirst = new Array (0,0);
          list.selectedIndices = selectFirst;
```

```
            list.scrollToIndex(0)
            trackToPlay = list.selectedItem.songString;
            gotoAndPlay(2);
        } else {
            channel.stop();
            var sn:uint = list.selectedItem.songNum;
            var selectNext = new Array (sn,sn);
            list.selectedIndices = selectNext;
            list.scrollToIndex(sn)
            trackToPlay = list.selectedItem.songString;
            gotoAndPlay(2);
        }
```

Nelle prime sei righe dell'istruzione condizionale `if`, viene richiesto al player di rieseguire e sottolineare la traccia iniziale una volta conclusa la riproduzione di tutti i brani. Il player, nel momento in cui individua che, il numero indicante la canzone in riproduzione è pari alla somma di tutte le tracce, percepisce l'obbligo di dover tornare ad eseguire il brano iniziale della playlist (`list.selectedItem.songNum == i`).

Al contrario, se il brano in riproduzione non è uguale alla somma dei brani, la voce `else` entra in funzione. In questo caso di fondamentale importanza è la variabile definita `sn`. Il player individua nell'acronimo `sn` l'imposizione di dover selezionare e riprodurre la traccia successiva a quella terminata.

A queste due funzioni viene inoltre aggiunto un metodo definito come `scrollIndex`.

Lo `scrollIndex` serve a scorrere l'elenco automaticamente fino alla voce in corrispondenza dell'indice specificato. In altre parole, questo metodo fa muovere automaticamente la barra di scorrimento una volta che la canzone cambia. Per lo stesso e identico concetto spiegato in precedenza, la barra di scorrimento torna in posizione iniziale nel momento in cui termina l'esecuzione dell'ultima canzone.

I pulsanti di riproduzione presentano ognuno una funzione diversa all'interno del player, il che vuol dire ricreare quattro funzioni differenti, ciascuna assegnata ad un pulsante corrispondente.

```
function stopPlayer (event:MouseEvent):void {
            channel.stop();
            isPlaying=false;
            playBtn.gotoAndStop(1);
            pausePosition = 0; }
```

La funzione `stopPlayer` viene richiamata ogni volta che l'utente clicca sul pulsante di stop. Questa funzione non fa altro che interrompere la riproduzione del brano. Nel caso in cui non ci sia nessun brano in riproduzione, in quel momento l'istruzione `channel.stop` viene eseguita ugualmente, senza però modificare niente.
La funzione `playPause` viene invocata ogni volta che l'utente clicca sul pulsante play. Questa funzione svolge due compiti:

- Mette in pausa il brano.
- Fa riprendere la riproduzione della canzone.

```
function playPause(event:MouseEvent):void {
      if (isPlaying == true) {
          pausePosition = channel.position;
          channel.stop();
          playBtn.gotoAndStop(1);
          isPlaying = false;
      } else {
          channel = snd.play(pausePosition);
          playBtn.gotoAndStop(2);
          isPlaying = true;}
}
```

Per decidere quale delle due operazioni svolgere il metodo controlla, attraverso il costruttore `if`, il valore della variabile `isPlaying`. Nel caso in cui questa variabile sia `true` allora significa che il player sta riproducendo una canzone che bisogna fermare; nel caso in cui la variabile sia `false`, allora significa che c'è una canzone in pausa che il player deve far riprendere.
Per mettere in pausa un brano, il player, prima di tutto memorizza nella variabile `pausePosition` la posizione dal quale far ripartire la canzone, dopo di che ferma la riproduzione, fa passare il bottone `playBtn` al frame 1 in modo tale da modificarne l'aspetto grafico e setta la variabile `isPlaying` a `false` in modo tale da segnalare che c'è un brano in pausa.
Per far riprendere la riproduzione di un brano il player come prima cosa, avvia la riproduzione della canzone dal punto indicato dalla variabile `pausePosition`, dopo di che fa passare il bottone `playBtn` al frame 2 per modificare il suo aspetto grafico ed infine setta la variabile `isPlaying` a `true`.

La funzione `playNext` viene utilizzata per mandare in esecuzione la canzone successiva. Questo metodo non fa altro che richiamare la funzione `newTrack()` creata in precedenza.

```
function playNext(event:MouseEvent):void { newTrack();}
```

Un discorso più complesso riguarda la spiegazione della funzione `previousSound`. La funzione `previousSound` viene lanciata ogni volta che l'utente vuole mandare in play il brano precedente a quello attualmente in esecuzione. Come prima cosa questo metodo istanzia delle variabili che utilizzerà più avanti. Queste variabili sono: `prevNum` `selectPrevSong`, `indexMinus1`, `lastSongInList`.
`prevNum` e `selectPrevSong` verranno utilizzate nel caso in cui il brano che è in riproduzione non è il primo della lista e verranno inizializzate in modo tale da mantenere in memoria la canzone precedente, invece `indexMinus1` e `lastSongInList` verrà utilizzato nel caso in cui il brano che si sta riproducendo è il primo e verranno quindi inizializzate in modo tale da tenere in memoria l'ultima canzone presente nella lista.

```
function previousSound(event:MouseEvent):void {
     var prevNum:uint = list.selectedItem.songNum - 2;
     var selectPrevSong = new Array (prevNum,prevNum);
     var indexMinus1:uint = i -1;   var lastSongInList = new Array
     (indexMinus1,indexMinus1);

      if (list.selectedItem.songNum == 1) {

                  if (snd.bytesLoaded != snd.bytesTotal) {
                       channel.stop();
                       snd.close();
                       list.selectedIndices = lastSongInList;
                       list.scrollToIndex(indexMinus1);
                       trackToPlay = list.selectedItem.songString;
                       gotoAndPlay(2);
                  } else {
                       channel.stop();
                       list.selectedIndices = lastSongInList;
                       list.scrollToIndex(indexMinus1);
                       trackToPlay = list.selectedItem.songString;
                       gotoAndPlay(2);
                  }
```

```
        } else {
                    if (snd.bytesLoaded != snd.bytesTotal) {
                        channel.stop();
                        snd.close();
                        list.selectedIndices = selectPrevSong;
                        list.scrollToIndex(prevNum);
                        trackToPlay = list.selectedItem.songString;
                        gotoAndPlay(2);
                    } else {
                        channel.stop();
                        list.selectedIndices = selectPrevSong;
                        list.scrollToIndex(prevNum);
                        trackToPlay = list.selectedItem.songString;
                        gotoAndPlay(2);
                    }
        }
        pausePosition = 0;
}
```

La funzione utilizza l'istruzione `if (list.selectedItem.songNum == 1)` per differenziare il caso in cui si sta riproducendo il primo brano dal caso in cui si sta riproducendo una qualsiasi altra canzone. Infatti la differenza sta nel fatto che nel caso in cui si sta riproducendo la prima canzone non si può semplicemente mandare in riproduzione il brano precedente in quanto non esiste, e questo genererebbe un errore. In questo caso si è scelto di mandare in riproduzione l'ultima canzone della lista.

L'istruzione `if (snd.bytesLoaded != snd.bytesTotal)` confronta i byte caricati con i byte totali della canzone. Se sono diversi significa che l'MP3 non è stato ancora caricato del tutto e che quindi è necessario richiamare la funzione `close` della variabile `snd`.

Dopo aver effettuato questo controllo, seleziona dalla lista il brano da mandare in play, richiama il metodo per lo scroller automatico in modo da far visualizzare all'utente il brano che è in play e richiama il frame 2 del filmato in modo da mandare in caricamento l'MP3 selezionato.

CONCLUSIONI

Il seguente lavoro è stato realizzato con lo scopo di sviluppare un media player dinamico adattabile a qualsiasi situazione nel campo web. In questa contesto non è stato possibile ricreare alcune circostanze pensate all'inizio della pianificazione del lavoro, questo perché Flash non permette la modifica in locale di file XML ma esclusivamente la sola lettura.
Ciò però è pensabile se il player viene implementato all'interno di siti web, creando uno script lato server (realizzato anche in PHP) che carica il documento XML remoto e lo fornisce in output o ne crea uno nuovo. In questo caso la provenienza del documento XML è la stessa di quella del filmato SWF, in quanto risiedono sullo stesso server. Con questo tipo di approccio è possibile far svolgere al player varie funzioni come, ad esempio, il caricamento di file MP3 all'interno del lettore e di conseguenza la rielaborazione del linguaggio XML in automatico.
La gamma di applicazioni e contenuti per Actionscript si è spostato in modo significativo negli ultimi anni, mentre il linguaggio Actionscript 3.0 rimane praticamente invariato dalla sua introduzione nel 2006. Adobe sta pensando, che sia ora di rivedere il linguaggio espandendo la sua ulteriore evoluzione verso una maggiore espressività, nonché guadagni di produttività e prestazioni.
In primo luogo, Adobe prevede di procedere ad aumenti significativi delle prestazioni a breve termine con l'obiettivo di continui miglioramenti delle prestazioni a lungo termine.
Allo stesso tempo, la casa americana mira ad aumentare la produttività degli sviluppatori, semplificando il linguaggio, migliorando strumento di supporto, e rafforzando la prevenzione bug. Infine, avendo ridotto inutili complessità, l'utente sarà in grado di innovare nei runtime Flash molto più rapidamente. Questa nuova release verrà chiamata quasi sicuramente Actionscript 4.0.

BIBLIOGRAFIA

[1] Goffredo Haus, *Compressione audio senza perdita di informazione*, dispensa del corso di Informatica applicata alla Musica, Università degli studi di Milano, 1994.

[2] Gianclaudio Floria, Giorgio Sitta, *Organizza la tua raccolta MP3*, Fag Digital Life Style, 2005.

[3] Martin Ruckert, *Understanding MP3: syntax, semantics, mathematics, and algorithms*, Gwv-Vieweg, 2005.

[4] Stefania Colonnese, *Lo Standard MPEG*, dispensa del corso di Comunicazioni Multimediali II, Università di Roma la Sapienza, 2008.

[5] Ciaccaglini Gianni, *Open XML. Guida allo sviluppo*, Fag Digital Life Style, 2008.

[6] Møller Anders, Schwartzbach Micheal, *Introduzione a XML*, Pearson, 2007.

[7] Luca A. Ludovico, Maurizio Longari, *Rappresentazione di informazione musicale simbolica mediante linguaggi di markup*, dispensa del corso di Informatica applicata alla Musica, Università degli studi di Milano, 2005.

[8] Elliotte Rusty Harold, W. Scott Means, *XML Guida di Riferimento*, Apogeo, 2001.

[9] Evandro Raffaella, Scataglini Carlo, *Informatica facile.Vol. 2: Laboratorio cooperativo per potenziare le competenze multimediali*, Centro Studi Erickson, 2007.

[10] Vasta Davide, De Marco Andrea, *Dreamweaver CS5*, Apogeo, 2010.

[11] Karlins David, *Adobe Dreamweaver CS5. 100 tecniche essenziali*, Mondadori Informatica, 2008.

[12] Castrofino Nicola, Gioffrè Bruno, *Adobe Flash CS4. Guida pratica*, Mondadori Informatica, 2011.

[13] Adobe Creative Team, *Adobe Flash CS4 professional. Classroom in a book. Corso ufficiale Adobe*, Pearson, 2009.

[14] Adobe Systems Incorporated, *Programmazione Actionscript 3.0*, guida ufficiale cartacea prodotta dagli sviluppatori, 2008.

[15] Salvaggio Alessandra, Testa Gualtiero, *Flash CS3 e Actionscript 3.0*, Fag Digital Life Style, 2008.

[16] Adobe Systems Incorporated, *Uso dei componenti Adobe Actionscript 3.0*, guida ufficiale cartacea prodotta dagli sviluppatori, 2008.

[17] David Stiller, Rich Shupe, Jen deHaan, Darren Richardson, *Actionscript 3.0. Le risposte per i professionisti Flash*, Tecniche Nuove, 2009.

RINGRAZIAMENTI

A conclusione del mio lavoro di tesi, desidero rivolgere un ringraziamento a chi mi ha aiutato a sviluppare e redigere questo contenuto. Uno speciale ringraziamento va al prof. Luca Andrea Ludovico e al dott. Adriano Baratè per la costante disponibilità dimostrata.
Al di là della tesi, mi sento di ringraziare innanzitutto tutti i miei familiari in particolare i miei genitori e mio fratello per avermi sostenuto in questi anni di studio e per essere stati un supporto ben più importante di quanto loro possano immaginare.
Ringrazio inoltre tutti i miei amici e i compagni di università per aver condiviso con me questo percorso.
Infine, un pensiero particolare, va al mio amico Diego che mi ha insegnato a non mollare mai nei momenti di difficoltà.