Università degli Studi di Milano

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE e NATURALI

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE DELLA COMUNICAZIONE MUSICALE

"RAPPRESENTAZIONE IN FORMATO PIANO ROLL DI DOCUMENTI IEEE 1599"

Tesi di Laurea di:

Daniele Norton

Matricola: 709162

Relatore:

Prof. Luca Andrea Ludovico

Correlatore:

Dott. Adriano Barate'

Anno Accademico 2008/2009

Indice

Indice delle figure

Introduzione

L'informazione musicale è caratterizzata da molti aspetti eterogenei. Ai suoi estremi possiamo individuare: da un lato l'aspetto simbolico, che la rappresenta nella sua forma originale (cioè tramite simboli), con le note e tutti i possibili segni della notazione (pause, alterazioni, chiavi, ecc.), e, dall'altro, l'audio, che riporta la performance della forma scritta. Vi sono poi una serie di informazioni legate ad un singolo brano: oltre ai dati di catalogazione (titolo, autore, genere, periodo storico,ecc.), è possibile individuare aspetti riguardanti la struttura, come la ripetizione di temi musicali, di sequenze o di segmenti particolari (il canone). In alcuni casi, poi, sono da curare anche le registrazioni video, stampe o libretti d'opera lirica connesse al brano stesso.

Come si può notare, l'informazione musicale è molto ricca e complessa. Tuttavia, in campo informatico, la maggior parte dei programmi finora sviluppati in questo ambito si concentrano solo su un singolo aspetto tra quelli appena citati.

Il formato IEEE 1599, standard approvato a livello mondiale nel giugno 2007 dalla IEEE Standard Association, è pensato per trattare la musica, e le relative informazioni ad essa collegata, nella sua globalità. Ciò ha reso necessario progettare un nuovo linguaggio che consentisse una gestione efficace di metadati complessi. Questo linguaggio, basato su XML, è in grado, infatti, di descrivere in maniera esauriente contenuti musicali eterogenei all'interno di un unico file, dove gli elementi presi in considerazione vengono collegati reciprocamente e sincronizzati attraverso una gestione spazio-temporale degli stessi.

Essendo un formato basato su XML, il singolo documento IEEE 1599 è scritto, di per sé, con un codice di codifica. Quest'ultimo non è adatto ad una fruizione musicale dal punto di vista visivo e dell'ascolto; quindi è necessario "trasformare" questi documenti, in modo tale da poter fornire un prodotto immediatamente utilizzabile dall'utente finale. Per fare questo, è possibile sviluppare e implementare diverse applicazioni in grado di leggere e utilizzare i documenti IEEE 1599.

Obbiettivi della tesi

L'obbiettivo del presente elaborato è quello di creare un plug-in, da integrare all'interno del frame work MX, in grado di tradurre automaticamente i file in formato IEEE 1599 in una corrispondente struttura musicale, che riproduce graficamente, e in modo sequenziale, un brano da cima a fondo. Le due principali strutture adottate in campo informatico sono la partitura multimediale e il piano-roll. A differenza della partitura, il piano-roll interpreta la musica in modo alternativo, ossia con un approccio visuale (rappresentazione vettoriale su un piano cartesiano) piuttosto che teorico. Il principio consiste, infatti, nell'utilizzo di barre (che rappresentano, di fatto, le note) poste all'interno di un piano: l'asse x indica la durata delle note, l'asse y, sul quale è posta l'immagine di una tastiera, indica la loro altezza.

La maggior parte dei piano-roll dispongono, inoltre, di una funzione che consente una trasposizione automatica nel loro formato partendo da altri, sia di tipo audio (wav, midi, ecc.), sia di tipo grafico (Finale, MusicXml, ecc.). Ad oggi non esiste ancora un piano roll in grado di tradurre automaticamente i file in formato IEEE 1599; il presente elaborato consiste proprio nello sviluppo di tale applicazione e nell'analisi di tutte le tecnologie necessarie per il suo funzionamento.

Durante l'attività di tirocinio svolta all'interno del LIM, Laboratorio di Informatica Musicale presso il Dipartimento di Informatica e Comunicazione dell'Università degli Studi di Milano, è stato possibile acquisire le nozioni adeguate per lo svolgimento dell'elaborato finale:

- sviluppo dell'interfaccia grafica di un'applicazione.
- importazione e manipolazione di un documento XML all'interno di un linguaggio di programmazione a oggetti, tramite l'utilizzo di un documento strutturalmente neutro per il linguaggio e per la piattaforma: il DOM, Document Object Model.

Il linguaggio di programmazione ad oggetti (per lo sviluppo di applicazioni software) adottato per la presente sperimentazione è C#, implementato con il software di editing Visual Studio Professional (VSP).
Per lo sviluppo dell'interfaccia grafica sono stati utilizzati gli strumenti e le funzionalità fornite dal software VSP, implementando diversi oggetti all'interno della finestra principale dell'applicazione (il form). Contenitori di immagini, pannelli, barre di scorrimento e forme, sono stati disegnati effettuando un inserimento manuale delle istruzioni di codice all'interno dell'editor, oppure, in alcuni casi, utilizzando i tool messi a disposizione dal software.
Successivamente, si è posta l'attenzione sull'utilizzo degli strumenti di implementazione, navigazione e modifica di documenti XML all'interno delle suddette interfacce, utilizzando un modello DOM per C#. Grazie a questa tecnica, è stato possibile accedere ai valori dei nodi presenti all'interno di un documento IEEE 1599 e utilizzarli in vari modi, facendo uso dei metodi forniti dal linguaggio di programmazione C#. I nodi in questione sono quelli riguardanti la partitura, in modo particolare ciò che indica l'altezza delle note, la loro durata e il posizionamento che viene assegnato a ciascuna di esse all'interno del pentagramma. Queste costituiranno le coordinate per il posizionamento delle barre all'interno dell'interfaccia grafica, e il valore stesso che verrà assegnato a ciascuna di esse.

Riassumendo, gli aspetti principali su cui la tesi ha posto l'attenzione sono i seguenti:

- Descrizione delle caratteristiche di un piano-roll, facendo un confronto con la notazione musicale utilizzata all'interno di una partitura.
- Confronto con la versione analogica: cenni storici e stato dell'arte.
- Tecnologie di riferimento.
- Analisi della struttura e dei tag che designano note, pause, chiavi, e tutti gli altri simboli che fanno parte della notazione musicale dei file in formato IEEE 1599.
- Analisi tecnico-funzionale di tutte le parti del programma sviluppato: il Form che lo caratterizza, le parti di codice più importanti e gli algoritmi principali.

Capitolo 1: Il formato piano roll

Con “rappresentazione in formato piano-roll” si intende una rappresentazione grafica alternativa dello spartito musicale, che permette di visualizzare la struttura musicale di un brano seguendo la sua effettiva sequenza temporale con un approccio meno teorico. Per chiarire meglio questo concetto è necessario introdurre alcuni aspetti della teoria musicale, e, successivamente, confrontare questa tipologia di notazione con la rispettiva rappresentazione in formato piano-roll.

1.1 LA NOTAZIONE MUSICALE

Secondo S. Giovanni Damasceno, la musica è “una successione di suoni che si chiamano l’un l’altro”. Ogni suono è caratterizzato da tre proprietà: altezza, intensità e timbro. L’altezza di un suono è il suo essere più acuto, più grave, oppure uguale rispetto ad un altro; l’intensità è la forza che percepiamo da esso (forte, piano,pianissimo); il timbro è la qualità che ci permette di distinguere quale sia la fonte sonora da cui proviene.

Nella sua forma scritta, la musica viene descritta in base all’altezza dei suoni che la compongono e alla relativa durata.

1.1.1 L’ALTEZZA

- **Le note musicali**

I segni grafici che rappresentano i suoni sono detti *Note*, e se ne possono contare sette: DO, RE, MI, FA, SOL, LA, SI, DO. Ognuna di queste viene scritta sul *Rigo* musicale, anche chiamato pentagramma (pente = cinque, e gramma = linea) in quanto formato da 5 linee orizzontali e da 4 spazi che intercorrono tra le linee. La posizione che ciascuna note assume sul pentagramma viene fissata dalla notazione: una nota posizionata più in alto rispetto ad un’altra è più acuta, mentre invece se posizionata più in basso è più grave. L’immagine sotto mostra ad esempio la scala di do.

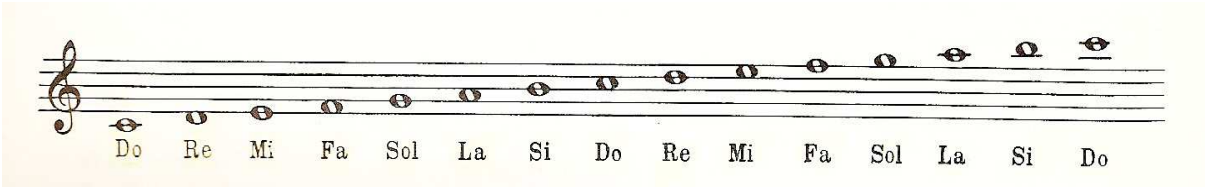


Fig.1 – Posizione delle note all’interno di un pentagramma.

Dato che in natura ogni suono può assumere molte altezza diverse, non è possibile rappresentarle tutte all’interno di un singolo pentagramma. Per questo motivo, la notazione utilizza, in alcuni casi, pentagrammi doppi e consente di scrivere le note al di fuori del rigo musicale usando i cosiddetti “tagli addizionali”, cioè dei trattini posti sotto o in mezzo alla nota. In questo modo il pentagramma viene, di

fatto, prolungato verso l'alto e verso il basso. L'immagine successiva chiarisce meglio questo concetto.

Fig.2 – Pentagramma doppio e tagli addizionali

Il fatto di aver dato alle note solo 7 nomi, inoltre, ha implicato il fatto di raggrupparle per ottave; ognuna è caratterizzata dalle note comprese tra due che hanno lo stesso nome. Se ad esempio partiamo de un do, l'ottava comprende tutte le note utilizzate per arrivare al do successivo. Lo strumento musicale che mostra efficacemente questa suddivisione è il pianoforte. L'immagine sotto riassume tutti i concetti trattati finora.

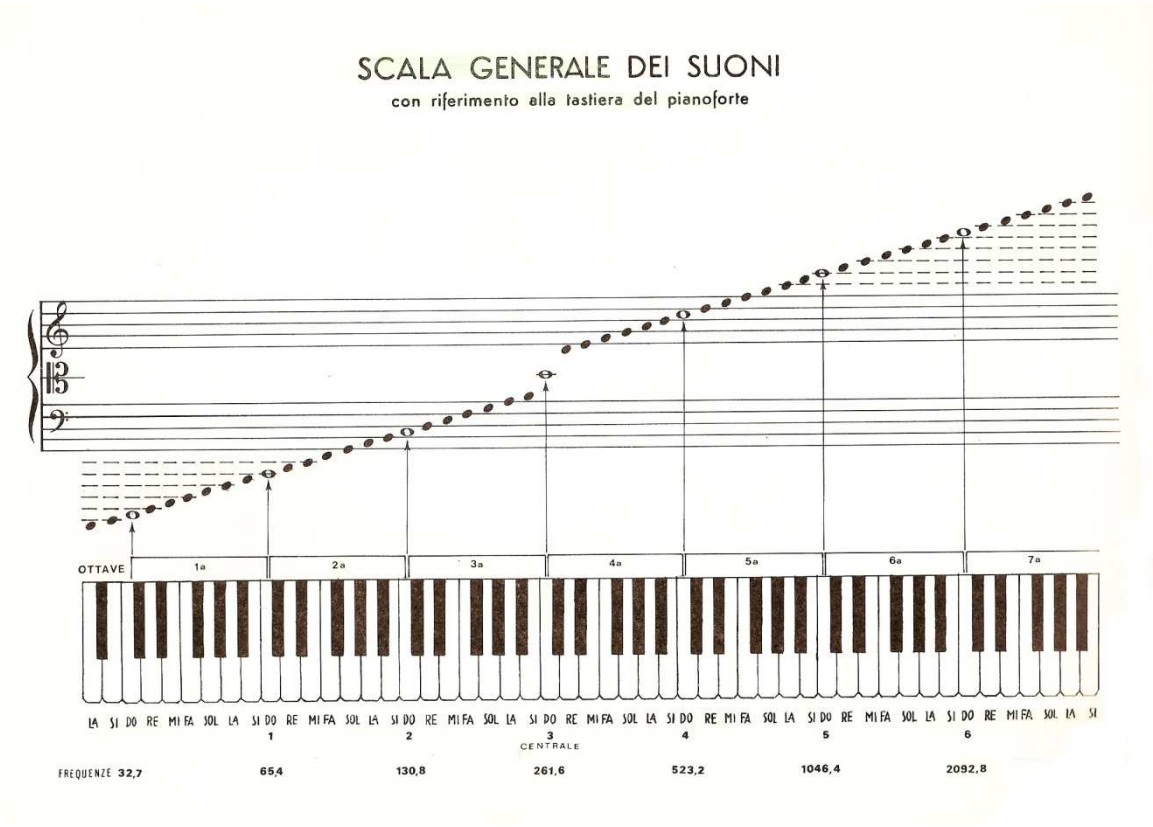


Fig.3 – Scala generale dei suoni

- **Le chiavi musicali**

All'inizio di ogni pentagramma è sempre presente un simbolo che indica la chiave musicale con cui il brano va letto. Le chiavi musicali sono state introdotte nella notazione per evitare l'utilizzo di un numero eccessivo di tagli addizionali. Se ad esempio prendiamo in considerazione strumenti dai suoni

perennemente acuti (come l'ottavino), senza l'ausilio di una chiave adeguata la loro partitura verrebbe scritta interamente al di sopra del rigo musicale (o al di sotto nel caso di strumenti gravi).

Nella pratica musicale ci sono tre tipi di chiave: chiave di SOL, chiave di DO e chiave di FA. Poste in diverse posizioni sul rigo musicale, le chiavi ci daranno la possibilità di posizionare sul pentagramma la maggior parte delle note formanti l'estensione dei suoni più o meno gravi o più o meno acuti di cui ogni strumento è dotato. Seguendo questo principio, le chiavi sono diventate sette (una chiave di SOL, quattro chiavi di DO e due chiavi di FA) e formano nel loro insieme il cosiddetto setticlavio. Nell'immagine sotto viene mostrata la posizione che la nota DO assume nelle diverse chiavi.

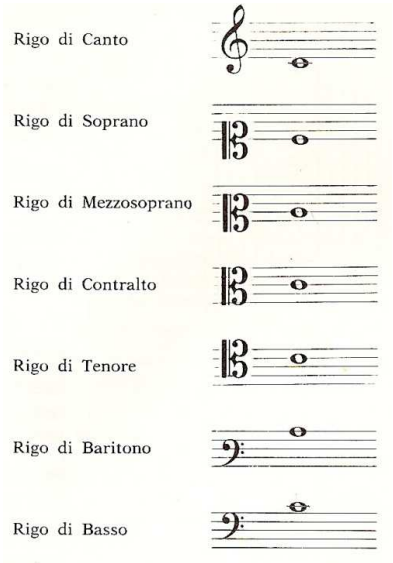


Fig.4 – Posizione del do all'interno del setticlavio

- **Le alterazioni**

Le *Alterazioni* sono segni grafici che, posti davanti a una nota, modificano verso l'alto o verso il basso l'intonazione della nota stessa. Sono cinque e vengono chiamate: *diesis*, *bemolle* (alterazioni semplici), *doppio diesis*, *doppio bemolle* (doppie) e *bequadro.* Vediamole singolarmente nell'immagine sottostante.

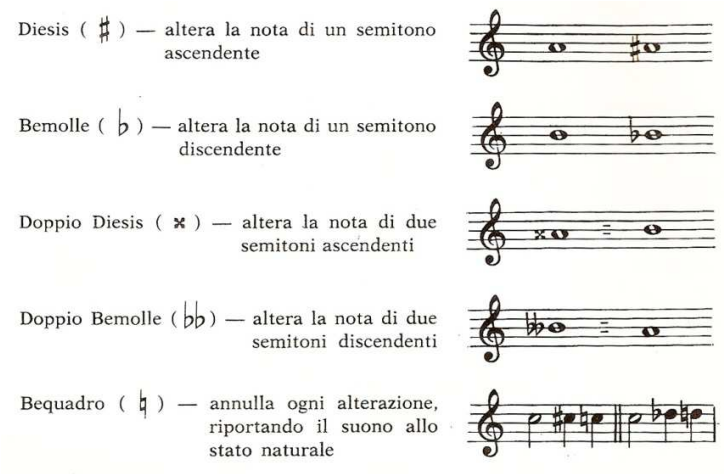


Fig.5 – Tipologie di alterazioni

Sul pianoforte, le principali note con alterazione semplice sono rappresentate dai tasti neri (es. DO#, RE#, ecc.), con qualche eccezione che li assegna a un tasto bianco. Bisogna inoltre sottolineare che ad ogni tasto corrispondono due note; secondo la teoria musicale, infatti, i suoni omofoni (oppure omologhi) sono quei suoni che hanno la stessa altezza ma nome diverso (es. DO# = RE*b).* Lo stesso discorso vale per le note con alterazioni doppie (es SOLx = LA).

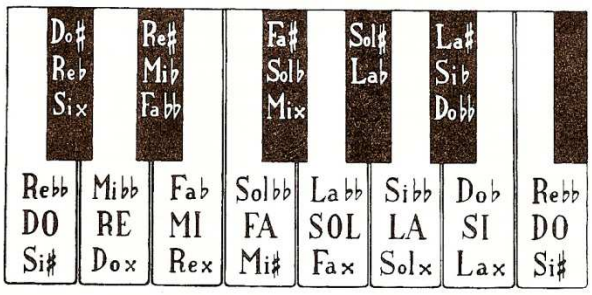


Fig.6 – I suoni omofoni

1.1.2 LA DURATA

- **Valori e pause musicali**

La durata dei suoni e dei silenzi è stabilita e rappresentata con precisione da particolari simboli grafici di forma diversa chiamati Figure musicali (o Valori) e Figure di silenzio (o Pause).

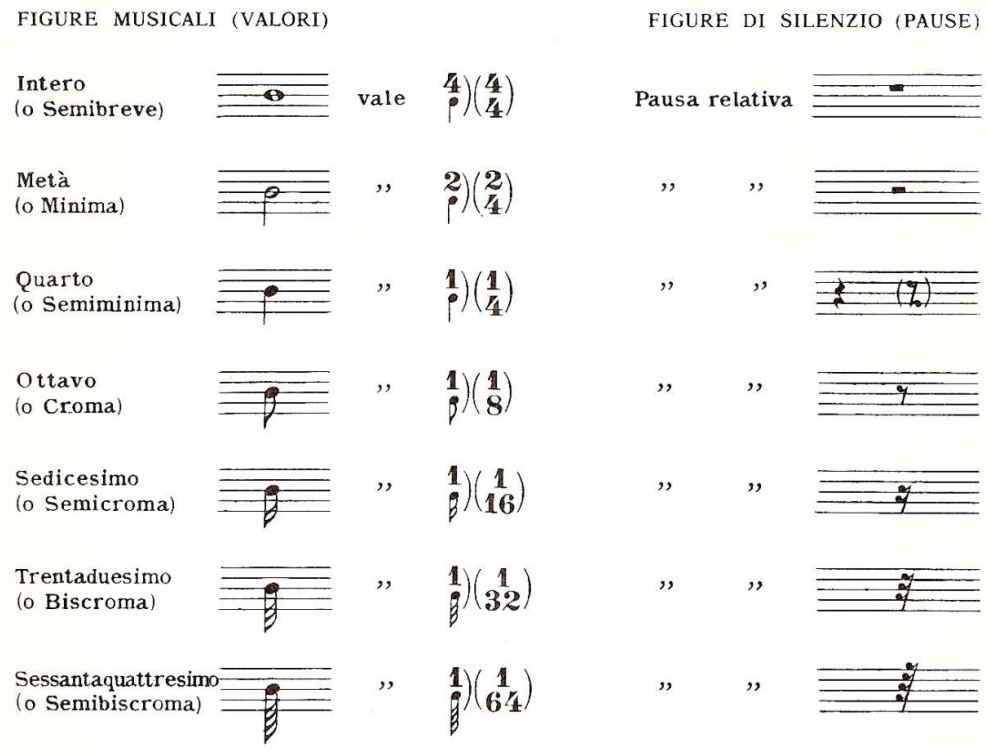


Fig.7 – Segni grafici dei valori di note e pause musicali

Tutti i valori e le pause hanno un preciso rapporto numerico: ogni figura e ogni pausa valgono infatti la metà della figura o della pausa precedente mostrate nell'immagine precedente. Ad esempio, 2 *Metà* formano un *Intero*, 2 *Quarti* formano una *Metà*, e così via. Di seguito viene mostrato il prospetto comparativo delle figure e delle pause.

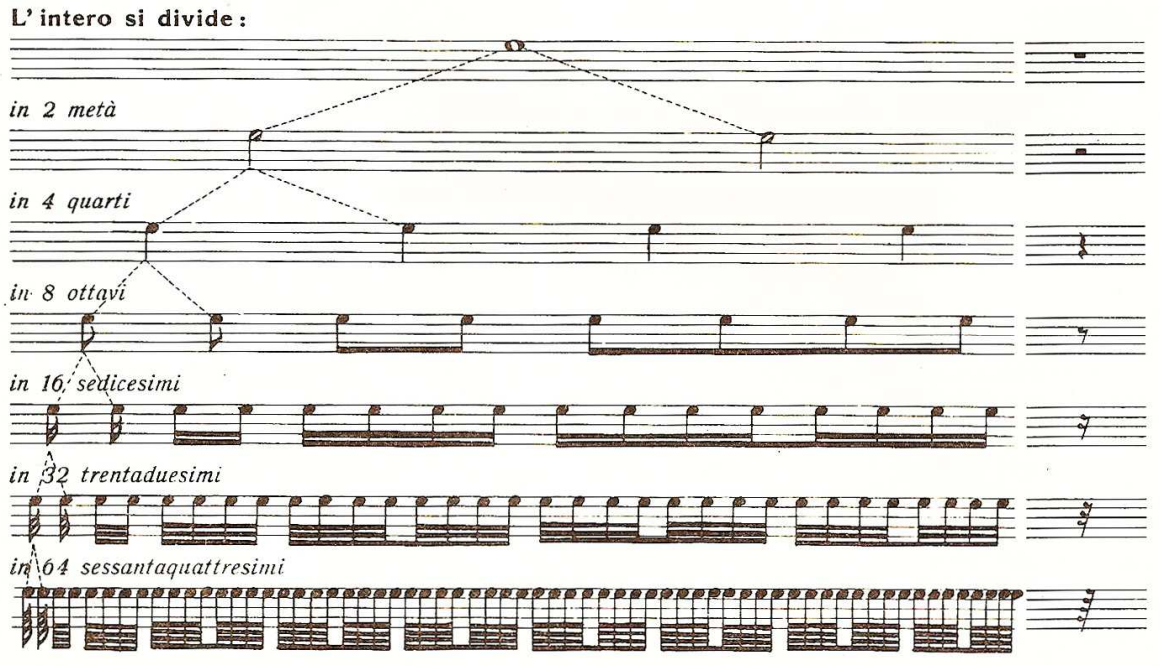


Fig.8 – Prospetto comparativo delle figure e delle pause

- **Il punto di valore**

Il punto di valore aumenta la nota (o la pausa) di metà del suo valore

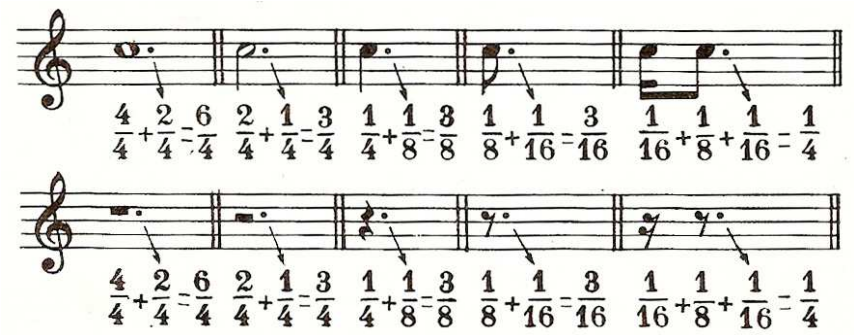


Fig.9 – Il punto di valore

- **La Battuta**

Per facilitare la lettura ritmica, ogni brano viene diviso in tante parti di eguale durata chiamate Misure o Battute; queste vengono disegnate attraverso due stanghette poste verticalmente sul pentagramma che indicano rispettivamente il loro punto di inizio e di fine. La stanghetta che separa le battute è

costituita da una linea semplice (salvo alcune eccezioni in cui è composta da due), mentre quella che indica la fine del brano presenta due linee: una sottile e una in grassetto.
Ogni battuta contiene un determinato numero di valori, note o pause, la somma dei quali è stabilita dal tempo segnato all'inizio del pentagramma. L'immagine sotto riporta 2 battute: una in tempo di 4/4 ed una in tempo di 2/4.

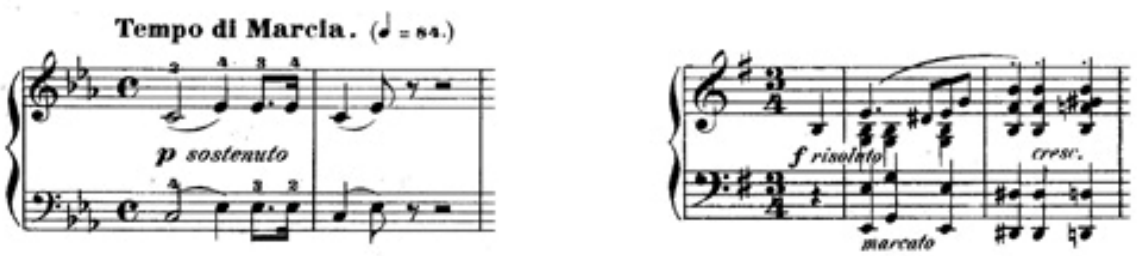


Fig.10 – Battute di 3/4 e 4/4

1.1.3 POSIZIONAMENTO DELLE NOTE

Per concludere il paragrafo sulla notazione musicale, viene presentato un breve frammento di partitura che mostra come le note vengono posizionate all'interno di un pentagramma seguendo le regole di altezza e durata descritte finora.

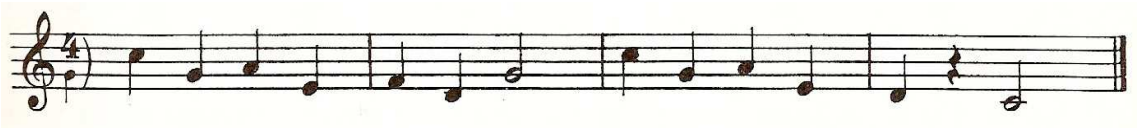


Fig.11 – Breve frammento musicale in tempo di 4/4

1.2 RAPPRESENTAZIONE IN FORMATO PIANO-ROLL

Come la partitura, anche il piano-roll è in grado di fornire tutte le componenti sopra citate dei singoli suoni. Inoltre, questo tipo di rappresentazione risulta molto più semplice da decifrare in fase di lettura. Agli occhi di un "non-musicista", infatti, la partitura risulta indecifrabile, ed è necessario acquisire diverse nozioni della teoria musicale per comprendere tutti i segni disegnati al suo interno. Nel caso del piano-roll, invece, la lettura risulta molto più semplice: basta semplicemente conoscere le note di un pianoforte ed il loro valore in termini di durata. Naturalmente questo tipo di rappresentazione non è in grado di sostituire completamente una partitura, in quanto sono troppi i segni notazionali che possibile incontrare durante la lettura di quest'ultima (segni di dinamica, di interpretazione, di chiave); tuttavia ne offre una valida alternativa.

Nel corso del presente paragrafo verranno inseriti degli screenshot del programma Visual Studio Professional nello sviluppo di un applicazione piano-roll che metteranno in evidenza alcuni aspetti di questa tipologia di rappresentazione.

1.2.1 ASPETTO GRAFICO DI UN PIANO-ROLL

Come già accennato nell'introduzione, un piano-roll opera una trasposizione della musica su due vettori:

- L'asse y indica l'altezza delle note;
- L'asse x indica il tempo e la durata delle note.

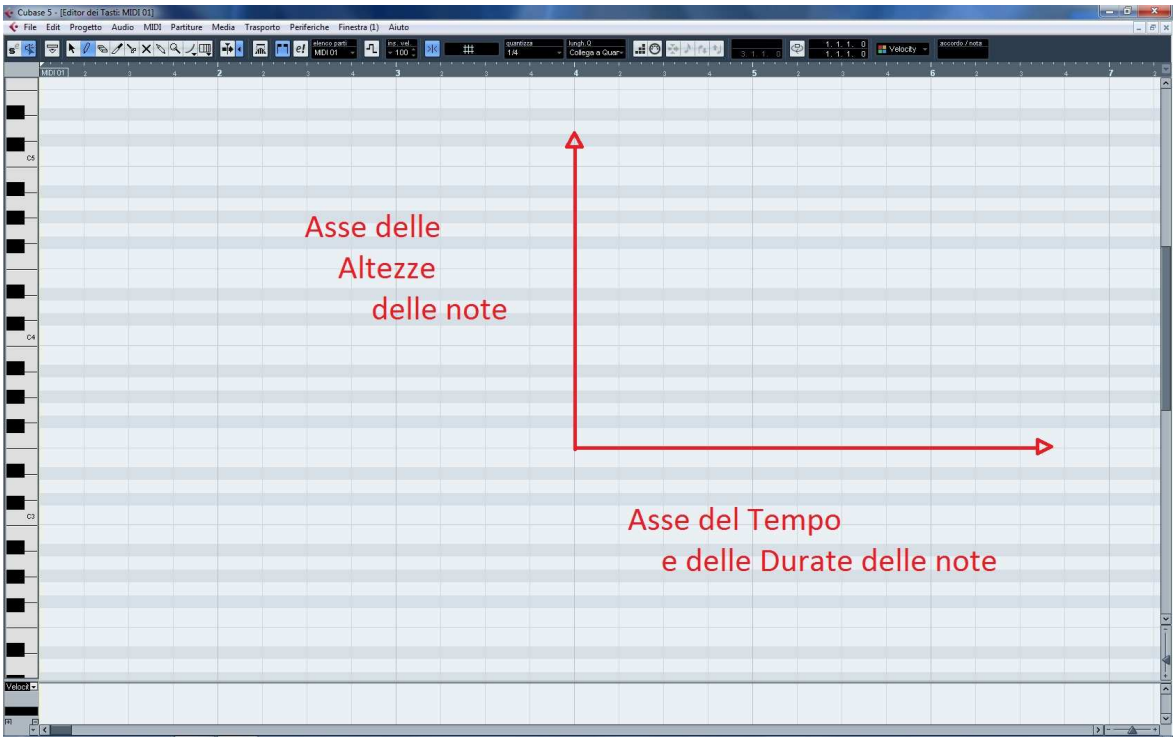


Fig.12 – Interfaccia grafica del piano-roll di cubase

All'interno di un piano-roll, i segni grafici che indicano le note sono delle barre. Queste vengono inserite all'interno della griglia che caratterizza il piano, dove assumono significati diversi in base alla posizione e alla dimensione che gli viene attribuita.

1.2.2 L'ALTEZZA

Per distinguere le varie note, ogni valore dell'asse y del piano corrisponde all'altezza specifica della nota (ad esempio il primo valore dell'asse y equivale alla nota DO dell' ottava 1). La corrispondenza valore - altezza nota viene effettuata accostando all'asse y l'immagine della tastiera di un pianoforte, in modo tale che l'assegnazione risulti semplificata dal punto di vista visivo e della comprensione.

La maggior parte delle griglie usate nei piano roll vengono disegnate con due colori diversi: uno per i valori che corrispondono ai tasti bianchi della tastiera, ed un altro per i tasti neri. Inoltre, sull'asse y della griglia sono spesso presenti delle linee maggiormente marcate rispetto ad altre per evidenziare i cambi di ottava.

1.2.3 LA DURATA

L'asse x del piano roll rappresenta la temporizzazione del brano. In questo caso la griglia è scandita in linee che indicano la durata minima (ad esempio 1/16, valore che viene scelto dall'utente) ed il cambio di battuta (linee più marcate rispetto alle altre). La posizione della barra sull'asse x indica quindi l'istante in cui una determinata nota si verifica all'interno del brano.

La durata della nota viene invece stabilita dalla lunghezza della barra. In base al numero di durate minime ricoperte dalla barra, viene attribuito il valore della nota in termini di durata.

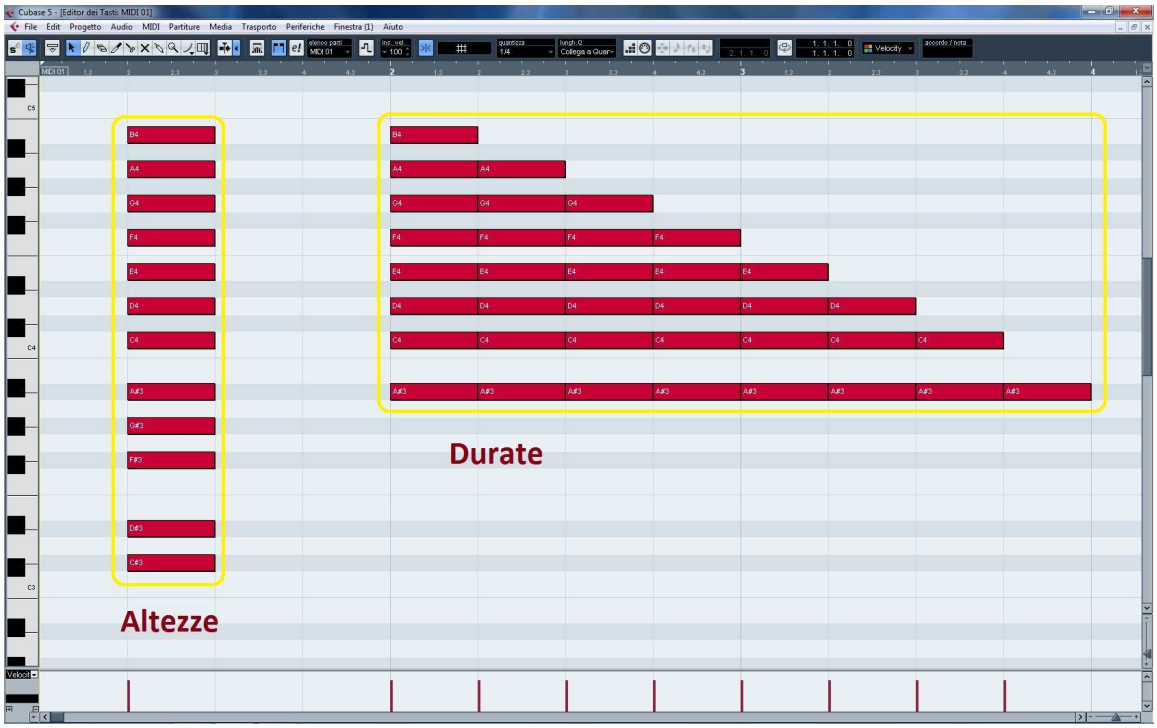


Fig.13 – Altezze e durate delle note di un piano roll

L'immagine sopra mostra come vengono raffigurate le note in un piano roll. Per chiarire le modalità di rappresentazione di altezza e durata, nell'esempio troviamo selezionate: sulla sinistra del piano le note dei tasti bianchi e neri (altezze) e sulla destra note con altezza casuale e durata progressivamente maggiore.

Capitolo 2: Nascita ed evoluzione del piano roll

La versione analogica del piano roll nasce intorno alla fine dell'ottocento, e consiste in un mezzo di riproduzione della musica azionato all'interno di un pianoforte meccanico. La funzione principale di questo strumento era quella di contenere le performance musicali dei musicisti dell'epoca. Il piano-roll moderno deriva dal principio di funzionamento di questo medium musicale.
In continua produzione di massa dal 1896 fino ai giorni nostri, questo mezzo di comunicazione ha subito nel corso degli anni numerose evoluzioni, fino a giungere all'attuale versione informatica.
In questo capitolo saranno brevemente analizzate le fasi evolutive del piano-roll e gli ambiti in cui oggi viene utilizzato.

2.1 IL PIANO-ROLL ANALOGICO

Prima dell'invenzione della radio, esistevano diversi mezzi di comunicazione in grado di riprodurre musica. Tra i più diffusi c'erano il Carrillon ("scatole musicali" cilindriche con ingranaggi a manovella), i cilindri (supporti in cera o acetato con un riproduttore dedicato) e, appunto, il piano roll.
A differenza dei cilindri, supporti che richiedevano meccanismi di registrazione sofisticati e consentivano solo un numero di copie limitate, il pano roll è stato il primo medium che ha permesso una produzione di massa della musica. Grazie alla semplicità con cui veniva creato e copiato industrialmente, questo strumento permetteva di fornire al cliente finale musica contemporanea in modo semplice e veloce.

- **Cenni storici**

Il piano-roll deve la sua nascita al suo mezzo di riproduzione: il "piano riproduttore".
Comunemente conosciuto con il termine "pianola", questo strumento è in grado di suonare la musica senza l'intervento di un pianista, grazie a dei marchingegni addizionali aggiunti al suo interno. Questi meccanismi hanno il compito di leggere le performance musicali, impresse su un apposito rullo e scritte sotto forma di codice (cioè il piano-roll), e di azionare di volta in volta i tasti corrispondenti sul pianoforte.
L'idea di un pianoforte in grado di suonare da solo nasce dal francese Jacquard Mills, che nel 1800 disegna su un telaio dei motivetti musicali sulla base di un diagramma di schede perforate. Nel 1863, un altro francese, Fourneaux, crea il primo "piano player" utilizzando il meccanismo del pianoforte pneumatico: con un breve rotolo di metallo forato avvolto su un bastone, le note del piano vengono suonate in base alla posizione specifica dei fori stessi. Questa invenzione, battezzata " il Pianista", è stata resa pubblica nell'Esibizione Centenaria di Philadelphia del 1876.
Pertanto, i primi piano rolls furono in metallo e in grado di suonare solo brevi canzoni.
L'invenzione vera e propria di questo supporto, intesa come rullo di carta perforato, è da attribuire invece al tedesco Edwin Welte, un costruttore di organi e "piano player": il primo rullo di carta fu

utilizzato in una orchestrion[1] della Welte & Sons nel 1883(87); su questo era possibile imprimere brani più lunghi grazie alla maggiore quantità di fori praticabili sul supporto cartaceo.
Edwin Scott Votey inventò un pianoforte che suonava con rotoli di carta grazie ad un meccanismo di pedali a motore, brevettato nel 1900. Nel 1911, ha inizio negli Stati Uniti la produzione su vasta scala dei piano-rolls cartacei, che ebbero una considerevole diffusione nel mondo tra il 1900 e il 1927.
Passati di moda durante il periodo della Grande Depressione, i pianoforti player divennero nuovamente popolari dopo la fine della seconda guerra mondiale. Molti pianoforti furono restaurati e ricominciò la produzione dei piano roll.
Esistono ancora copie di piano rolls, ma sono estremamente rari da trovare; per di più, sono costosi e utilizzabili con difficoltà tecniche non indifferenti per la loro estrema fragilità.

2.1.1 IL SUPPORTO

La struttura fisica di un piano roll consiste in un rullo di carta sul quale vengono impressi dei piccoli fori. La posizione e la lunghezza di questi fori determina rispettivamente l'altezza e la durata della nota che verrà suonata sul pianoforte.
Inizialmente il rullo aveva una dimensione in altezza tale da poter contenere 58 buchi sul piano verticale; successivamente è stato espanso a 65, fino ad arrivare a 88 fori, uno per ogni tasto del pianoforte. Dopo che centinaia di aziende di questo settore in espansione avevano prodotto rotoli di pianoforte differenti in termini di dimensioni e di perforazione, nel 1909 i produttori americani accettarono uno standard definito nella convenzione Buffalo.

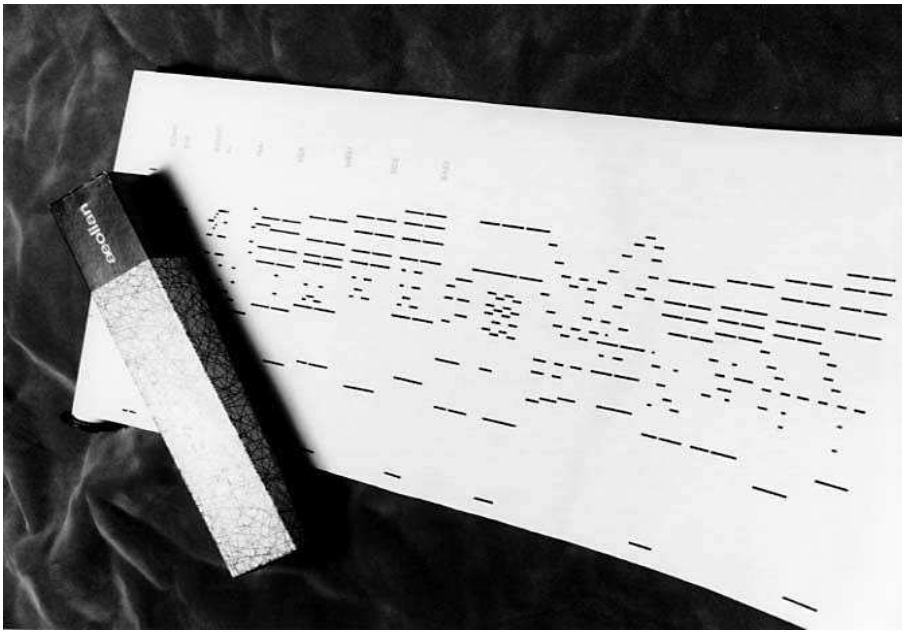


Fig.14 – Rullo di carta perforato

[1] Nome generico che indica una macchina in grado di simulare i suoni di un'orchestra o di una band.

Le modalità di perforazione del rullo erano diverse, ma ne venivano utilizzate principalmente due. Il primo metodo consisteva nel tagliare manualmente (con un coltello) dei buchi nella carta, tenendo a portata di mano lo spartito come guida ("arranged rolls"); il risultato era quello di un suono un po' meccanico.

Fig.15 – Incisione manuale di un rullo di carta

In alternativa, era possibile imprimere i fori direttamente durante la performance musicale, utilizzando dei "pianoforti registratori" ("hand played" rolls). Questa registrazione manuale veniva realizzata con rotoli in grado di catturare i suoni prodotti da un pianista, grazie ad un dispositivo collegato ad un pianoforte meccanico che traccia la carta in tempo reale.

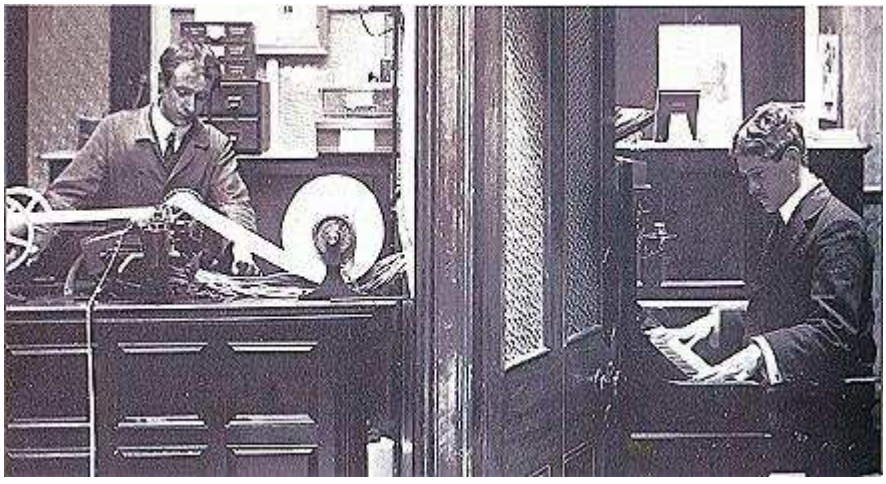


Fig.16 – Registrazione – incisione di una performance musicale

Alcuni rulli venivano prodotti sfruttando una combinazione dei due metodi: la carta marcata (forata) automaticamente dal piano veniva utilizzata come una guida, e su questa potevano essere aggiunte successivamente altre note e cancellati gli eventuali errori dopo il processo di registrazione. Questo

metodo era già usato nel 1904 dalla società Welte in Germania. Con il loro piano riproduttore Welte-Mignon, furono eseguite parecchie registrazioni con pianisti famosi come Camille Saint-Saëns, Richard Strauss, Alexander Scriabin e George Gershwin, ricavandone registrazioni di inestimabile valore storico (e di qualità superiore a molte registrazioni audio dell'epoca).
Tuttavia, questa modalità forniva ai produttori la possibilità di creare una musica talmente articolata che sarebbe stata impossibile da riprodurre per l'uomo; per questo motivo, l'editing di post produzione doveva essere contenuto e proporzionato alle reali capacità di prestazione dei musicisti.

2.1.2 LA PIANOLA E IL PIANOFORTE MECCANICO

Pianola, un termine mutuato dal linguaggio popolare, fu registrato come marchio dalla The Corporation Eolie di New York nel 1890 e divenne ben presto un nome generico per indicare l'auto-riproduzione nei pianoforti. Molte altre aziende produttrici di pianoforti meccanici provarono ad introdurre un marchio che identificasse il prodotto, ma soltanto il nome commerciale "Phonola", utilizzato dalla compagnia tedesca Hupfeld, riuscì nell'intento, e solo in Europa occidentale.
Il principio di funzionamento di un pianoforte meccanico consiste null'utilizzare l'aspirazione e la pressione dell'aria generata da due pedali (o da un motore elettrico) collegati ad un mantice di grandi dimensioni posizionato nella parte inferiore del pianoforte. Nella stessa zona è presente il bar 'tracker' (in italiano "barra inseguitrice), una barra di metallo caratterizzata da 88 fori, ognuno dei quali è collegato, nella parte posteriore, ad un tubo. Ogni tubo è a sua volta collegato ad un sistema che aziona la singola nota (pertanto, anche i tubi sono 88, uno per ogni tasto del pianoforte).
Quando l'operatore spinge i pedali, una porzione d'aria riempie il mantice e un'altra porzione mette in movimento il rotolo di carta perforato che scorre sul tracker bar. Mentre il rullo scorre, l'aria del mantice viene soffiata sul foglio: se questa incontra una perforazione sul rotolo, attraversa il foro corrispondente sulla barra di tracker, entra nel tubo ad esso collegato e viene immessa nel sistema che aziona la nota (ossia il martelletto che percuote la corda associata al tasto).

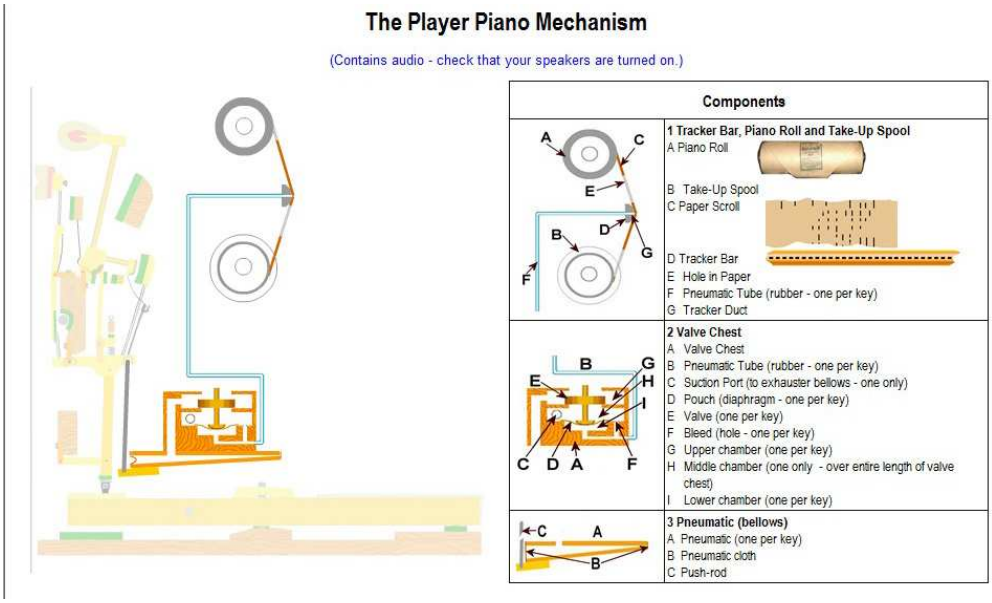


Fig.17 – Meccanismo di un pianoforte meccanico

La variazione del volume d'aria, indispensabile per variare l'espressione delle note, si otteneva variando la forza e la velocità di spinta del piede sui pedali. Allo stesso modo, poiché le liste musicali sono perforate a un ritmo costante, tutte le fluttuazioni di fraseggio e rubato dovevano essere introdotte per mezzo di una leva di tempo (di solito gestite dalla mano destra), in modo da controllare la velocità del rullo di carta.

2.2 IL PIANO-ROLL DIGITALE

Sfruttando a piene mani la filosofia di fondo che ha ispirato i piano-rolls, i file MIDI (acronimo di Musical Instrument Digital Interface) rappresentano il modo più moderno di salvataggio-conservazione delle performance musicali; anche l'aspetto esteriore è lo stesso: i software per l'editing di questi file forniscono una modalità di rappresentazione dalla musica in formato piano-roll. Approfondiamo questi concetti.

2.2.1 IL FILE MIDI

Il file MIDI esegue digitalmente ed elettronicamente ciò che un piano-roll faceva meccanicamente. Trattandosi di un file (cioè un codice binario letto da un computer), può essere salvato in tutti i moderni supporti informatici di salvataggio, dal floppy-disk al CD-ROM. Di conseguenza, anche il mezzo di riproduzione utilizzato è diverso: non si tratta più di pianoforti meccanici (e quindi analogici) ma di strumenti musicali digitali che supportano questa tecnologia. Il principio di funzionamento è però lo stesso: questi strumenti, prendendo in input un file codificato in MIDI, sono in grado di riprodurre autonomamente una performance musicale.
Gli strumenti musicali MIDI prendono in input i file in due modalità alternative. Se dotati di un lettore dedicato (floppy-disk o CD) eseguono la performance memorizzata all'interno del supporto in cui è stata salvata. Altrimenti, attraverso un hardware chiamato "interfaccia MIDI" collegato ad un computer, esegue le informazioni che gli vengono passate da una macchina.
Le funzioni e le caratteristiche fornite da questi file sono molteplici, ma non riguardano ciò che stiamo discutendo; per questo motivo, ci soffermeremo esclusivamente sul legame riguardante il piano roll.

2.2.2 IL FORMATO PIANO ROLL

Quasi tutti i software di editing (cioè di scrittura) MIDI utilizzano una rappresentazione grafica del brano che ricalca per filo e per segno la modalità di scrittura delle note di un piano-roll.
Karl Steinberg è stato il primo ad utilizzare la visualizzazione di performance musicali in formato piano-roll all'interno di un software da lui inventato: Cubase, un programma dedicato principalmente alla composizione e registrazione di audio professionale. Data la semplicità e l'immediatezza che forniva, questo tipo di rappresentazione fu subito adottato da molti altri software dello stesso settore.
Per la pagina principale di editing di cubase, Steinberg si ispirò agli stessi rotoli di carta perforata che facevano funzionare i pianoforti meccanici un secolo fa, mantenendo le regole di sintassi che questi

utilizzavano (fori che indicano altezza e durata delle note), oggi alla base della rappresentazione degli eventi midi negli editor di tutti i software musicali.
L'immagine che segue mette a confronto il Grid Editor (una modalità particolare di rappresentazione delle note) usata nel programma Steinberg Pro 24 (antenato di Cubase) e il primo Piano Roll di Cubase 1.0/Atari.

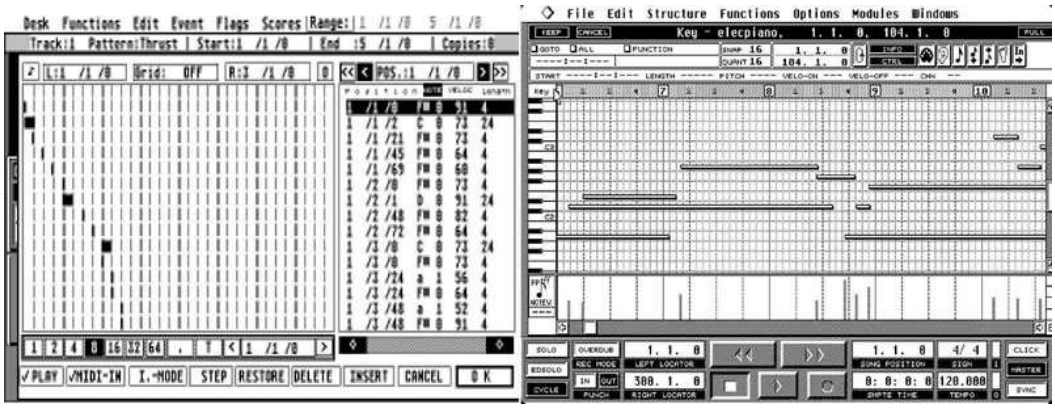


Fig.18 – Grid Editor e Piano Roll a confronto

Durante l'analisi del piano roll analogico nel capitolo precedente sono state illustrate le caratteristiche fisiche e le modalità di scrittura – modifica del sopporto cartaceo. Dato che l'aspetto grafico di un piano roll digitale è già stato esposto nel primo capitolo, nel paragrafo successivo verrà analizzato lo stato dell'arte e le funzionalità che la sua versione informatica mette a disposizione dell'utente. Il software che fornisce il maggior numero di funzionalità ad un piano roll è cubase; pertanto, prenderemo quest'ultimo come modello da esaminare, anche se non tutti i software di questo settore prevedono le funzionalità esposto qui di seguito.

2.3 STATO DELL'ARTE

Come anticipato in precedenza, la maggior parte dei software di editing MIDI attualmente in circolazione sfrutta una rappresentazione della musica in formato piano roll. Nel seguente capitolo verranno presentati alcuni di questi software per fornire una panoramica dello stato dell'arte attuale riguardante il piano roll.

- **Editing di un piano roll**

Il piano-roll digitale, così come avveniva con la versione analagica, consente all'utente di intervenire sulla musica scritta nel "rullo": oltre alla fase di scrittura, consente anche una fase di revisione e modica. Le due componenti principali sono il disegno di una tastiera e una griglia contenente le note (Display Note), dove, contrariamente a quello che ci si potrebbe aspettare, la tastiera disegnata sulla sinistra non serve ad inserire le note, ma ne indica solo l'altezza. Tuttavia, questo strumento risulta molto utile durante questo processo: la pressione di un tasto con il mouse attiva l'audio con l'altezza della nota (grazie ad un "feedback acustico") ed evidenzia la linea corrispondente sulla griglia.

- **Fase di scrittura**

Le modalità di scrittura di un file in formato piano-roll sono di due tipologie: inserimento manuale (con il mouse) delle singole note, inserimento automatico di una performance suonata in tempo reale.

L'inserimento manuale viene effettuato utilizzando il tool matita messo a disposizione dalla barra degli strumenti del software, attivato da un altro tool chiamato snap. Tenendo il disegno della tastiera come riferimento e facendo un semplice click all'interno della griglia, viene inserita una nota dell' altezza corrispondente, con un valore di durata minimo. Quest'ultimo, che corrisponde graficamente ad un quadratino, viene impostato dall'utente variando il cosiddetto valore di quantizzazione: se ad esempio viene assegnato il valore di un ottavo, ad ogni quadratino il medesimo valore. Se si tiene premuto il tasto del mouse (dai punti inizio e fine nota) e si scorre orizzontalmente la griglia, è quindi possibile inserire note della durata desiderata.

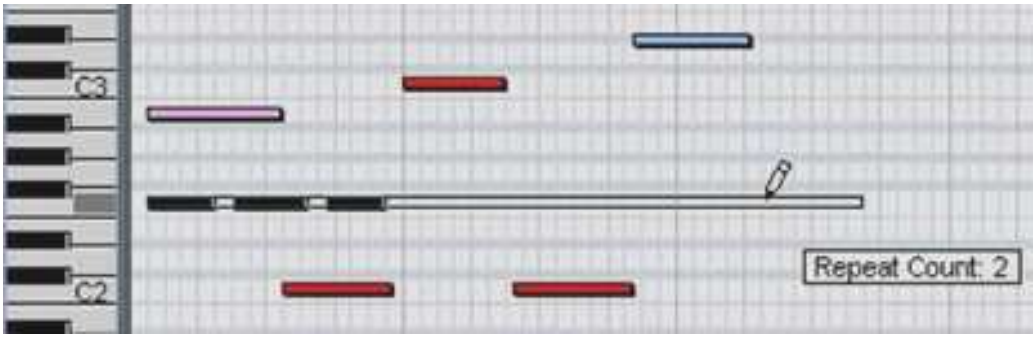


Fig.19 – lo strumento matita del piano roll di cubase

L'inserimento automatico di una performance segue lo stesso principio dei rulli registratori analogici: l'acquisizione avviene tramite il collegamento di uno strumento musicale MIDI ad un computer, che trascrive le note all'interno del piano-roll in tempo reale.

Una variante di questa tipologia è l'inserimento step-by-step (passo dopo passo). Questa funzionalità viene attivata da un tool (in cubase si chiama registrazione step) che permette di scandire con estrema precisione la durata delle note. Impostando un determinato valore di quantizzazione, la nota che viene registrata rimane entro i limiti definiti da questo valore e, ogni volta che il tasto viene rilasciato, il cursore si sposta in avanti di una posizione (sempre seguendo le impostazione di quantizzazione). Se ad esempio viene impostato un valore di quantizzazione pari ad un ottavo, nel momento in cui viene inserita una nota che dura un quarto e un sedicesimo, quest'ultimo viene letteralmente tagliato. Volendo inserire valori diversi, è sufficiente modificare volta per volta le impostazioni di quantizzazione. Si tratta di un processo un po' macchinoso, molto utile, però, per musicisti alle prime armi. Grazie a questa funzione, infatti, è possibile inserire le note di uno spartito all'interno della griglia senza dover necessariamente andare a tempo.

I parametri di quantizzazione vengono anche utilizzati per definire le dimensioni e le caratteristiche della griglia, al fine di gestire in modo efficace la durata delle note. Partendo dal valore di un tick, cioè la minima suddivisione possibile per un evento sonoro, è possibile definire la dimensione di un quadratino e, di conseguenza, della battuta. Per fare un esempio, in Cubase il numero massimo di ticks supportato per battuta è 3840 (cioè divide una battuta in altrettanti punti), un quarto di battuta corrisponde a 960 ticks, un ottavo a 480 e così via. La suddivisione in ticks è indispensabile alla precisione della scansione ritmica: con un valore di BPM di 120, una battuta dura esattamente 2 secondi. In questo lasso di tempo, Cubase è in grado di posizionare un evento con una precisione di +/- 0,5 millisecondi (2000ms/3840 ticks). Una indicazione di 3.2.4.120, ad esempio, vuol dire che il nostro evento è posizionato a battuta 3, secondo quarto, quarto sedicesimo, primo trentaduesimo. Questo concetto, che spesso si dà per scontato, è alla base del funzionamento di un qualsiasi programma di hard disk recording; in mancanza di esso, semplicemente il programma non potrebbe funzionare.

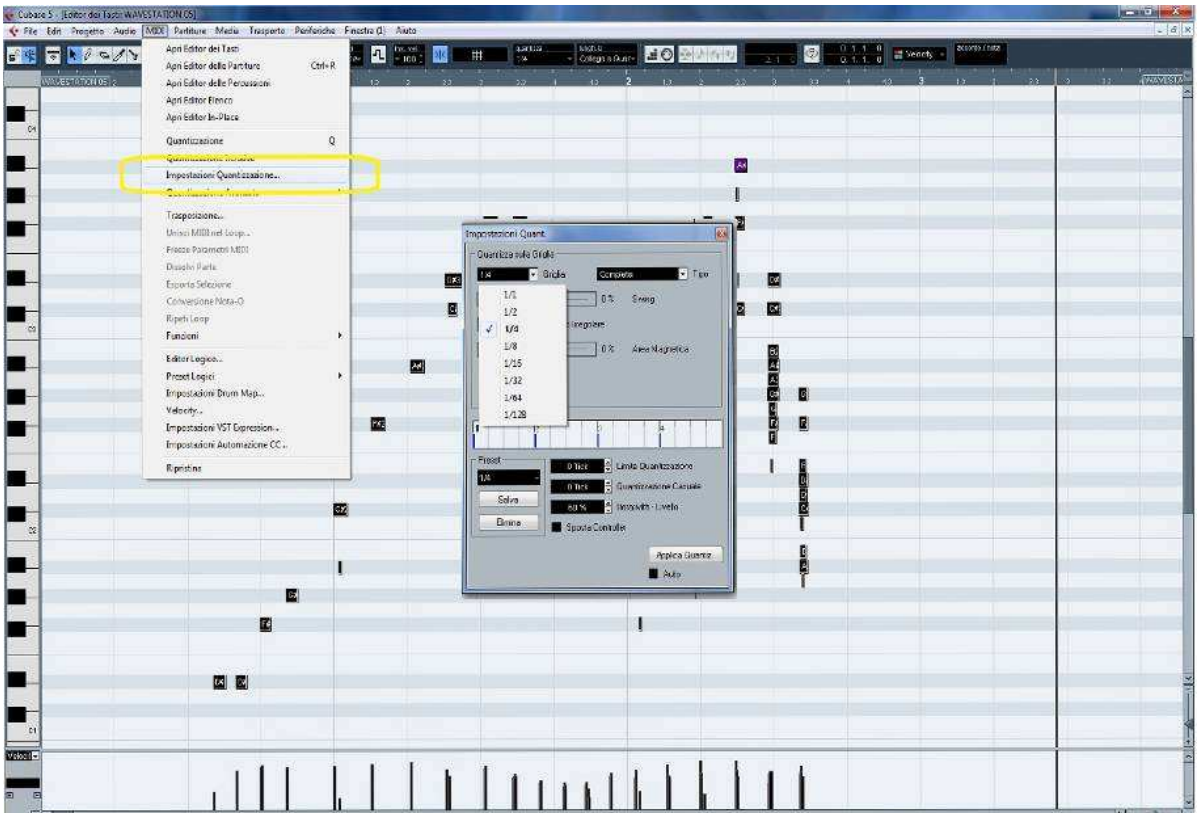


Fig.20 – Parametri di quantizzazione

- **Fase di modifica**

Come avveniva nel supporto cartaceo, anche il piano-roll digitale permette di modificare le performance in un momento successivo alla registrazione: rimuovere errori, inserire nuove note, spostarle e modificarle. Ognuna di queste operazione viene messa a disposizione nella barra degli strumenti (tool) del software che utilizza il piano-roll. Le icone più utilizzate sono: puntatore, matita, gomma, trim, forbici, mute, colla, lente di ingrandimento, linea, warp.

Lo strumento **trim** ridimensiona la durata delle note. Il tool **linea** permette, invece, di disegnare una serie di note con varie modalità: la figura sotto mostra un esempio di inserimento di una scala di note discendenti utilizzando proprio questo strumento.

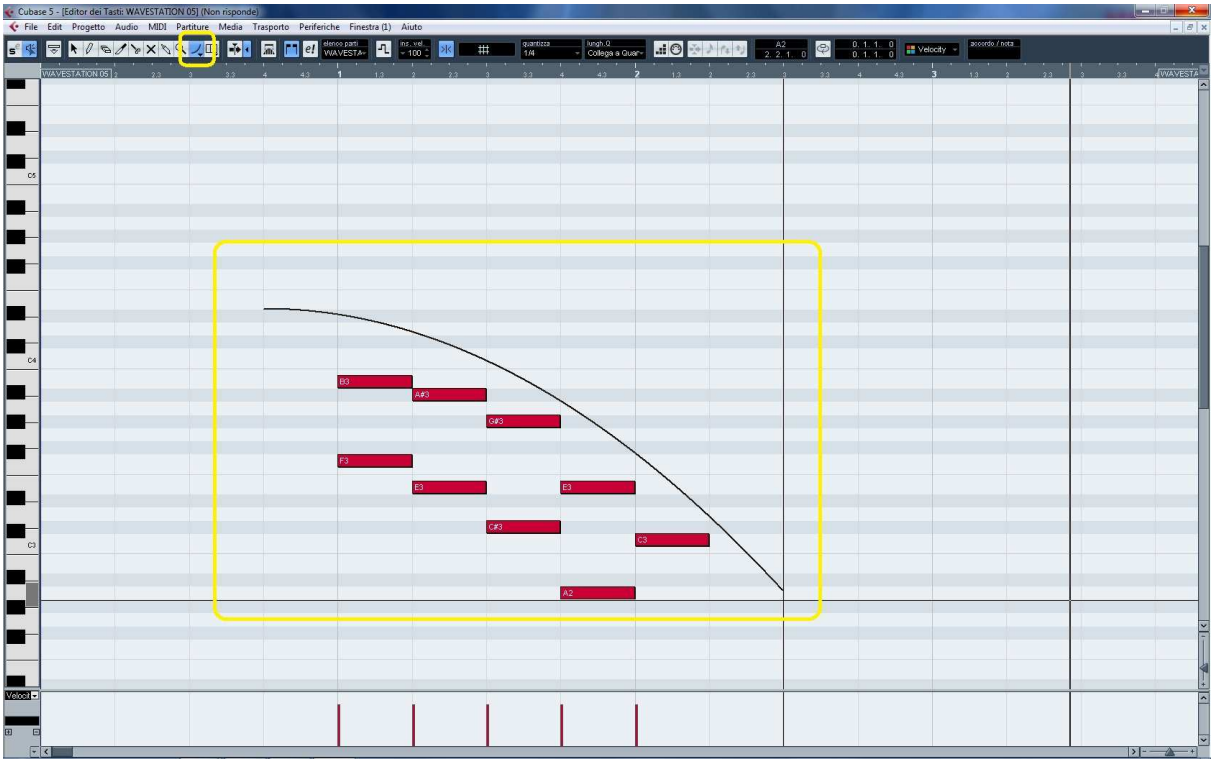


Fig.21 – lo strumento linea del piano roll di cubase

La selezione degli eventi da modificare può avvenire in diversi modi: click sulla singola nota, strumento rettangolo per la selezione di più note, tasti freccia della tastiera del PC e il menu **seleziona**. Quest'ultimo è sicuramente il più interessante, in quanto fornisce una varietà di comandi utili a facilitare il lavoro di selezione eventi, come ad esempio tutte le note della stessa altezza o appartenenti ad una determinata ottava. Una volta selezionato l'evento (o gli eventi), le operazioni di modifica possibili sono molteplici. Tra le principali troviamo:

- la duplicazione delle note in un punto vuoto della griglia;
- la trasposizione delle note selezionate in termini di semitoni o di ottave;
- la ripetizione di un evento per n volte: puntando il mouse sull'estremità destra di una selezione e scorrendo verso destra si ottiene la sua ripetizione;
- il trascinamento delle note e l'inversione di due note contigue;
- la modifica della durata delle note, stirando orizzontalmente l'estremità destra della nota selezionata;
- l'editing via MIDI: attivando l'icona **MIDI INPUT** nella toolbar e suonando su una MIDI keyboard, l'altezza della nota selezionata sul piano-roll verrà modificata.

Capitolo 3: Tecnologie di riferimento

Nello sviluppo dell'applicazione Piano roll sono state coinvolte diverse tecnologie; nel seguente capitolo ne verrà fatta una breve presentazione.

3.1 C# (sharp)

Il linguaggio di programmazione scelto per lo sviluppo dell'applicazione è C#. Si tratta di linguaggio di programmazione orientato agli oggetti, creato dalla Microsoft ed estendibile su ogni tipologia di piattaforma.
La sintassi del C# prende spunto da altri linguaggi Object Oriented, in particolare da: Daelphi (hanno il medesimo autore, ovvero Anders Hejlsberg), C++, Java e Visual Basic, per gli strumenti di programmazione visuale e per la sua semplicità (meno simbolismo rispetto a C++, meno elementi decorativi rispetto a Java). Come tutti questi linguaggi, C# viene usato per creare una varietà di applicazioni, compresi siti web, strumenti di sviluppo (development tool) e compilatori.
L'implementazione del C# è stata fatta con l'ausilio del software di editing Visual Studio Professional (VSP). Grazie alle funzionalità fornite da questo software, la definizione delle classi e dei metodi C# viene semplificata dalla "casella degli strumenti": gli "oggetti" vengono visualizzati graficamente in un elenco (in base alla necessità), inseriti nella finestra principale dell'applicazione (il form) e implementati, successivamente, con C#.
Durante lo sviluppo dell'applicazione, contenitori di immagini, pannelli, barre di scorrimento e forme, sono stati disegnati effettuando un inserimento manuale delle istruzioni di codice all'interno dell'editor, oppure, in alcuni casi, utilizzando i tool messi a disposizione dal software.
C scharp ha inoltre la possibilità di interagire con altri linguaggi. Tra questi c'è XML, con il quale può comunicare tramite l'utilizzo di un documento strutturalmente neutro per il linguaggio e per la piattaforma: il DOM (Document Object Model). Una volta importato il file, la navigazione e la manipolazione dei documenti XML avviene tramite delle query Xpath.
Vediamo le motivazioni che portano all'utilizzo di queste tecnologie e in cosa consistono.

3.2 XML e la musica

I file che vengono presi in input dal'applicazione piano roll (cioè i documenti IEEE 1599) sono in formato XML. E' bene quindi chiarire innanzitutto cos'è di fatto XML e il motivo per cui può essere utilizzato nella definizione di formati che trattano informazione musicale.

- **Presentazione di XML**

La definizione di XML, acronimo di eXtensible Markup Language, è quella di meta-linguaggio di markup, cioè un linguaggio che permette di definire altri linguaggi di markup. Di fatto, però, non è altro

che un insieme standard di regole sintattiche, in grado di modellare la struttura di documenti e dati. Questo insieme di regole (dette specifiche) definiscono, infatti, le modalità secondo cui è possibile crearsi un proprio linguaggio di markup, senza imporre tag predefiniti. Non serve per definire pagine Web, né per programmare, ma esclusivamente per definire altri linguaggi: è uno standard utilizzato per la descrizione di documenti. I linguaggi definiti tramite XML presentano molti vantaggi, tra i quali vi sono:

- Indipendenza dalla piattaforma hardware e software;
- Interscambiabilità in rete;
- Struttura gerarchica dell' informazione;
- Intelligibilità dell'informazione codificata;
- Estensibilità;
- Disponibilità di tool per l'implementazione del formato e per la validazione dei file prodotti.

Intorno ad XML ruotano una serie di tecnologie e linguaggi che supportano l'uso di questo meta-linguaggio e lo rendono affidabile e flessibile per gli usi più disparati: dalla definizione della struttura di documenti allo scambio di informazioni tra sistemi diversi, dalla rappresentazione di immagini alla definizione di formati di dati.

- **Formati musicali XML**

Anche l'informazione musicale risulta essere fortemente strutturata e gerarchizzata; per questo motivo, XML è stato più volte utilizzato nella definizione di linguaggi markup legati a questa materia, soprattutto rivolti alla rappresentazione di informazione musicale simbolica.
I principali formati musicali che sono stati definiti con la tecnologia XML sono:

- MusicXML, sviluppato da Michael Good, è un linguaggio markup che modella la struttura di documenti di tipo partitura, compatibile con il software di editing FINALE sia in fase di scrittura che in fase di lettura;
- MusiXML, di Gerd Castan, per la separazione della forma dal contenuto;
- MusiCAT/MDL di Perry Roland, che gestisce le informazione di catalogazione legate ad un brano;
- MPEG7-Audio, usato per propositi MIR (Music Information Retrivial), cioè per la localizzazione di brani all'interno di un database tramite input musicale.

3.3 IL FORMATO IEEE1599

Anche il formato IEEE 1599 è un linguaggio basato su XML. Data la sua complessità, le struttura di questo formato verrà analizzata in dettaglio nel prossimo capitolo, mentre invece di seguito saranno descritte le sue funzioni principali e la sua struttura generale.

- **Funzionalità**

IEEE 1599 è un formato finalizzato ad una descrizione esauriente dell'informazione musicale. Questo formato è stato sviluppato principalmente presso il LIM (Laboratorio di Informatica Musicale - Dipartimento di Informatica e Comunicazione - Università degli Studi di Milano).
Il linguaggio è una meta-rappresentazione dell'info musicale per descrivere, collegare e sincronizzare l'informazione musicale all'interno di un ambiente multi-livello. Lo scopo è realizzare un'integrazione tra diversi livelli di rappresentazione: simbolico, strutturale, notazione, computer performance e digital sound.
Dal Settembre 2008, il formato diventa uno standard internazionale, approvato dall'organo **IEEE-Standard Association**. Lo standard proposto dovrebbe integrare la rappresentazione musicale con gli standard già esistenti ed essere supportato da qualsiasi genere di software che tratta informazione musicale, ad es. applicazioni per l'editing di spartiti, per la performance musicale multimediali, per la composizione, per la ricerca musicologica, database e OMR (Optical Music Recgnition).

- **Struttura generale**

Gli autori del formato IEEE 1599 hanno individuato sei diversi strati (*layers*) per descrivere la musica nella sua interezza, ognuno dei quali rappresenta un grado di astrazione diverso dell'informazione musicale. I livelli sono: general, music logic, structural, notation, performance e audio.

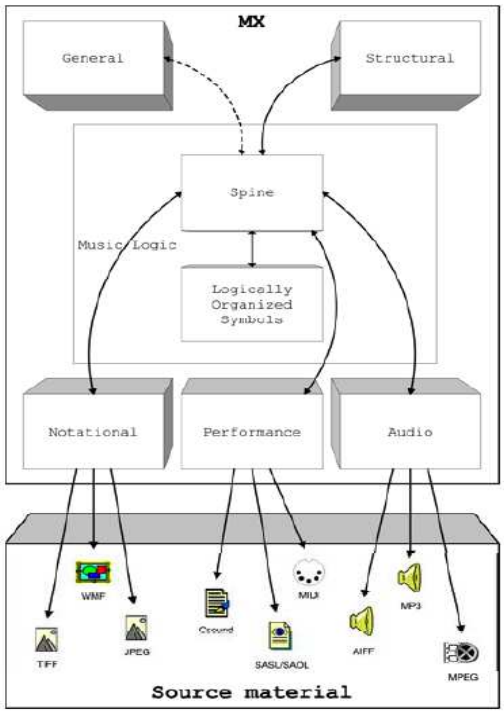


Fig.22 – Grafico che rappresenta la struttura di un doc IEEE1599

Ogni tipo di rappresentazione musicale utilizza diversi formati di codifica (ad esempio MP3 e WAV per l'audio, MIDI e MPEG per la performance, TIFF e JPG per la notazione e così via), e per poter sincronizzare un brano nella forma di spartito al relativo audio, video, e così via, è necessaria una struttura che rappresenti la relazione spazio-temporale implicita della musica. Il formato IEEE 1599 individua tale struttura nello spine, che mette in relazione i differenti formati di file per ottenere una descrizione completa dell'informazione musicale. Tutti i layer, ad eccezione del layer general, fanno riferimento ad esso, in una sorta di struttura a stella.
Un brano musicale può dunque essere descritto nella sua interezza (oggetti musicali, scansioni della partitura, esecuzioni sintetizzate, registrazioni,…), e questo porta indubbi benefici nel campo dell'analisi e dell'organizzazione dell'informazione, in quanto costringe a una sua visione strutturata.

Analizzando la sintassi XML di un documento IEEE 1599, si può notare che i singoli livelli sono indipendenti l'uno dall'altro.

```
<ieee1599>
        <general>…</general>
        <logic>…</logic>
        <structural>…</structural>
        <notational>…</notational>
        <performance>…</performance>
        <audio>…</audio>
</ieee1599>
```

Fig.23 – Sintassi XML della struttura generale di un doc IEEE 1599

3.4 DOM E METODI XPATH

Per sviluppare un'applicazione in grado di gestire documenti in formato IEEE 1599 è necessario utilizzare una tecnologia in grado di entrare nel documento stesso, prelevare i valori ritenuti interessanti e, in un secondo momento, utilizzarli. C# è in grado di fare questo guardando i file XML come **Document Object Model** (**DOM***)*, cioè sotto forma di documenti strutturati come modello orientato agli oggetti, neutrali sia per il linguaggio che per la piattaforma.

La classe DOM è, di fatto, una particolare rappresentazione in memoria di un documento XML. Attraverso la classe **XmlDocument** è possibile l' implementazione del DOM, mentre invece la classe di implementazione **XPath** permette una navigazione al suo interno ed una eventuale modifica a livello di codice.

Consideriamo una porzione di documento IEEE 1599:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ieee1599 SYSTEM "http://standards.ieee.org/downloads/1599/1599-
2008/ieee1599.dtd"[]>
<ieee1599 version="1.0" creator="Luca A. Ludovico">
        <general> …  </general>
        <logic>
            <spine>
                        <event id="clef_1" timing="0" hpos="1" />
        </spine>
        <los> ... </los>
        </logic>
        <performance> … </performance>
        <audio> … </audio>
        …
</ieee1599>
```

Nella figura seguente viene illustrato come è strutturata la memoria quando questa porzione di documento IEEE1599 viene letto nella struttura del DOM.

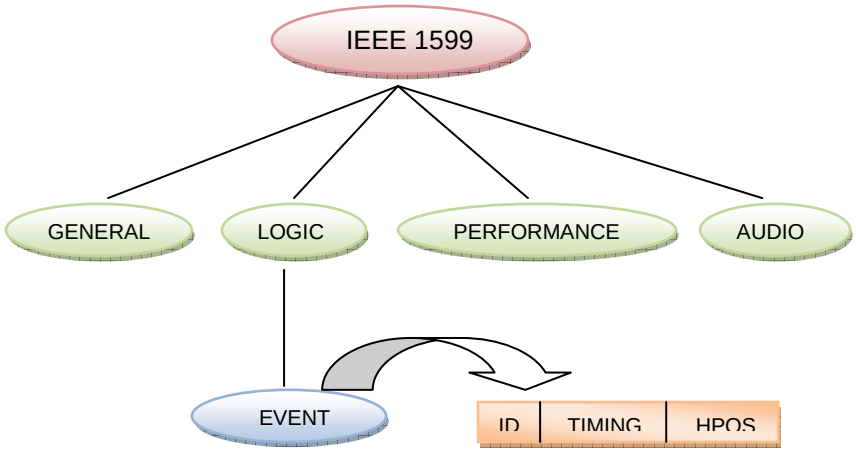


Fig.24 – struttura DOM di un documento IEEE 1599

All'interno della struttura del documento XML, ogni cerchio di questa figura rappresenta un nodo, denominato oggetto **XmlNode**, che costituisce l'oggetto base nella struttura DOM. La classe XmlDocument, che estende XmlNode, supporta i metodi per l'esecuzione di operazioni sul documento nella sua totalità, ad esempio il caricamento del documento in memoria o il salvataggio del documento XML in un file.

XmlNode e XmlDocument dispongono di metodi e proprietà per accedere e apportare modifiche ai nodi specifici del DOM, o addirittura recuperare interi nodi e informazioni contenute nel nodo (numeri, testo, ecc.).

Anche gli oggetti **Node** hanno dei metodi propri e sono caratterizzati da aspetti ben definiti:

- Ogni nodo presenta un singolo nodo padre, ovvero quello al livello immediatamente superiore. Gli unici nodi che non presentano nodi padre sono al livello radice del documento (nel nostro caso, il nodo root è Ieee1599), in quanto si tratta dei nodi di primo livello contenenti il documento stesso.
- La maggior parte dei nodi può presentare più nodi figlio, ovvero quelli al livello immediatamente inferiore.
- I nodi che si trovano sullo stesso livello (ad esempio i nodi los e spine) sono nodi di pari livello.

Una caratteristica del DOM è rappresentata dalla gestione degli **attributi**, raffigurati nella struttura sopra all'interno di un rettangolo. Gli attributi non sono nodi che fanno parte delle relazioni tra nodi padre, nodi figlio o nodi di pari livello, ma sono considerati una proprietà del nodo e sono costituiti da una coppia composta da nome e valore (nell'esempio precedente, il nodo event ha tre attributi di nome id, timing e hpos, con i rispettivi valori clef_1, 0 e 1). Per recuperare l'attributo timing del nodo event, è necessario chiamare il metodo **GetAttribute** quando il cursore si trova in corrispondenza del nodo in questione.

Per selezionare un insieme di nodi con lo stesso nome viene utilizzato il metodo **XmlNodelist**, il quale crea una lista di nodi xml seguendo la query Xpath inserita al suo interno. E' possibile scorrere singolarmente tutti nodi della lista; per ognuno di questi, è inoltre possibile selezionare altri nodi collegati ad esso, tra cui tutti i suoi nodi figli con lo stesso nome (metodo SelectNodes), un singolo nodo figli (SelectSingleNode), il nodo padre (ParentNode), i suoi attributi (GetAttribute), e molto altro ancora.

Capitolo 4: Il layer Music Logic

Come già anticipato nell'introduzione del presente testo, per poter sviluppare un'applicazione che prende in input un documento IEEE 1599 è necessario accedere ai nodi del documento stesso e, successivamente, utilizzare i valori letti tramite i metodi forniti dal linguaggio di programmazione C#.

Nel caso del piano roll, le informazioni che interessano all'applicazione sono quelle riguardanti la partitura, e in modo particolare ciò che indica l'altezza delle note, la loro durata, e il posizionamento che viene assegnato a ciascuna di esse all'interno del pentagramma. Questi aspetti sono raccolti all'interno del layer MUSIC LOGIC; per questo motivo analizzeremo dettagliatamente solo questo livello e tralasceremo tutti gli altri.

4.1 COMPONENTI

Il layer music logic è il livello principale dell'IEEE 1599, in quanto contiene le informazioni dello spine e di partitura, indispensabili per la natura multi-livello dello formato stesso.
Presenta tre sottolivelli (elementi figli):

- SPINE = contiene la funzione di mappatura spazio-temporale
- LOS = descrive gli oggetti in partitura
- LAYOUT = si occupa di rappresentare il layout del brano

Dato che le informazioni di LAYOUT non vengono utilizzate all' interno dell'applicazione piano-roll, verranno esaminati esclusivamente i livelli SPINE e LOS.

4.2 LO SPINE

Lo spine può essere visto come una funzione di mappatura bidimensionale tra i domini dello spazio e del tempo che funge da collante tra i diversi strati. Si compone di eventi, ciascuno dei quali presenta un riferimento nel dominio dello spazio e del tempo, con l'obiettivo di costruire una struttura astratta cui fanno riferimento tutti gli strati che descrivono le proprietà del materiale originario.
L'utilità dello spine deriva dalle differenti codifiche adottate per l'informazione musicale che rappresenta il materiale di base; si rende dunque necessario un punto di riferimento unico per tutte le istanze appartenenti a layer diversi (ad esempio, file MIDI, TIFF e WAVE) o anche allo stesso layer (ad esempio, gli MP3 con esecuzioni differenti della stessa partitura). Due eventi fisici diversi (quali due note) in due istanze distinte (quali due file che contengono la registrazione dello stesso pezzo) possono così far riferimento allo stesso evento logico, ossia allo stesso simbolo in partitura. Analogamente, l'immagine scansionata di una porzione di partitura e l'audio della sua esecuzione possono essere emanazioni degli stessi eventi logici.

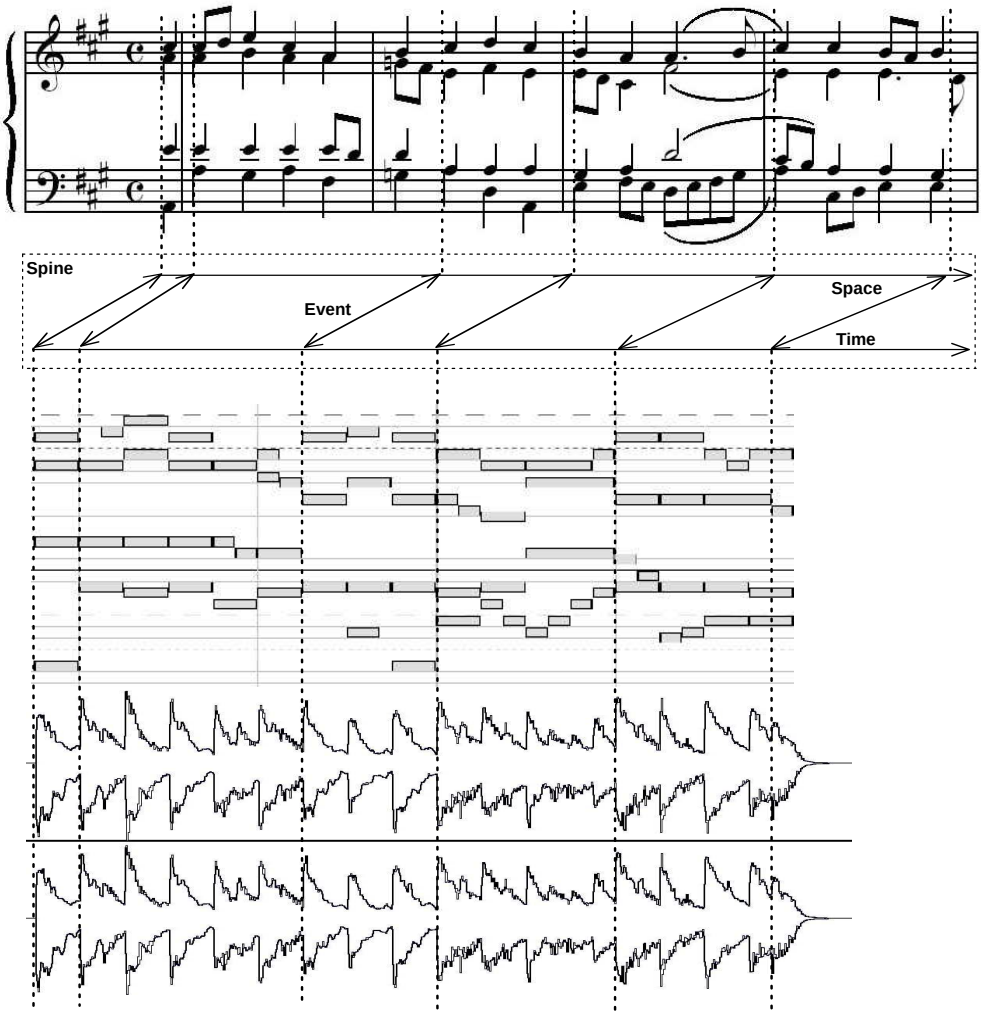


Fig.25 – Rapporto spine – spartito musicale

Nello spine vengono etichettati tutti gli eventi “significativi” in partitura: note, pause, segnature di chiave e di tempo, armature di chiave, segni di agogica, di articolazione e via dicendo. La quantità di questi eventi viene definita in modo arbitrario: tutto ciò di cui si vuole far riferimento nella codifica IEEE 1599 deve avere una riga corrispondente nello spine, senza necessariamente includere le informazioni ritenute inutili.

Da una presentazione generale dello spine (figura sotto) si può notare che al suo interno viene definito un ordinamento stretto tra eventi, che possono essere descritti anche in modo disordinato (ad esempio, la nota 25 di una voce può essere descritta prima della nota 1 nella partitura). Questo è possibile in quanto lo spine ricrea l’ordinamento esatto attraverso l’attributo di identificazione univoco dell’evento (eventi id = ”evento n”); id duplicati costituiranno, quindi, errore, e il file IEEE 1599 non risulterebbe valido.

```
<ieee1599>
        …
        <logic>
                <spine>
                        <event id="evento0" timing="NULL" hpos="NULL"/>
                        <event id="evento1" timing="0" hpos="6"/>
                        <event id="evento2" timing="1000" hpos="0"/>
                </spine>
                …
        </logic>
        …
</ieee1599>
```

L'attributo **timing** costituisce una temporizzazione virtuale in VTU (*virtual timing units*). Esistono delle norme per scegliere l'unità di misura, che non è fissata: i VTU non sono millisecondi, decimi o quarti d'ora, ma l'utente sceglie che significato temporale attribuire ai VTU di volta in volta. Un possibile assegnamento può essere ad esempio:

croma = 1 semiminima = 2 minima = 4

L'**hpos**, invece, tiene conto della spazializzazione orizzontale (sempre virtuale). Il valore contenuto al suo interno indicherà, quindi, la posizione in cui la nota verrà posizionata all'interno del pentagramma. La partitura va concepita tutta di seguito e senza ritorni a capo e, anche in questo caso, l'unità di misura può essere stabilita arbitrariamente.

Entrambi gli attributi **timing** e **hpos** supportano i valori numerici nulli, interi positivi e il valore speciale NULL, che va inserito quando non ha senso parlare di temporizzazione o di spazializzazione (come nel caso dell'armatura di chiave). Inoltre, i due attributi sono relativi, e non assoluti: ogni valore fa riferimento a quello immediatamente precedente. La contemporaneità spaziale e/o temporale viene denotata dal valore 0.

4.3 IL LOS

Le informazioni simboliche di partitura vengono raggruppate all'interno del layer LOS (Logical Organized Symbols) : è qui che troviamo *note, pause, chiavi, segni di articolazione, dinamiche,* e, in generale, tutto il contenuto informativo simbolico di un brano.

```
<ieee1599>
        …
        <logic>
                <spine> … </spine>
                <los>
                        PUNTO DI INSERIMENTO DELLA PARTITURA
                </los>
        </logic>
        …
</ieee1599>
```

I documenti IEEE 1599 stabiliscono una gerarchia delle informazioni presenti in partitura. Il livello più alto è quello costituito dalla partitura stessa (nella sua interezza), mentre il livello più basso è rappresentato da simboli quali note e pause.

La struttura completa è la seguente:

- una **partitura** è formata da più **accollature**
- un'**accollatura** è formata da più **pentagrammi**
- un **pentagramma** contiene (o meglio, può contenere) più **parti**
- una **parte** può essere suddivisa in più **voci**
- una **voce** intera viene considerata **battuta per battuta**
- una **battuta** contiene **accordi** e **pause**
- un **accordo** è formato da più **note** o **pause**

Con il termine voce si intende la linea melodica eseguita da un singolo strumento musicale o da una cantante. La parte, invece, indica la categoria di esecutori che andranno a suonare il brano, e può contenere una o più voci. Per fare un esempio, la parte del violino può contenere le voci di violino 1 e violino 2. La notazione musicale permette di inserire all'interno di un pentagramma una o più voci, e, talvolta, anche una o più parti. Talvolta in letteratura è possibile trovare anche contro esempi problematici, come la migrazione di una voce da un pentagramma a un altro (presente in numerosi pezzi pianistici per le voci interne), o il ribaltamento della gerarchia sopra descritta, nel senso che non è il pentagramma a contenere più parti bensì una parte ad abbracciare più pentagrammi (come è consueto ad es nella musica per tastiera).
Le battute sono descritte parte per parte: prima i contenuti delle misure da 1 a n della parte 1 (voce 1, voce 2,...) poi quelle delle misure da 1 a n della parte 2 (voce 1, voce 2,...) e così per tutte le altre parti. Quindi, IEEE 1599 non mette in risalto i rapporti di sovrapposizione verticale o di sincronizzazione temporale, o almeno non sono evidenti "a occhio nudo". Per rendere l'informazione

simbolica fruibile ad un musicista, e poter quindi trasformare il formato di codifica IEEE 1599 in un formato di presentazione, sarà necessario ricorrere ad un editor grafico o ad un viewer.
Per via della posizione gerarchica in cui il termine compare, per accordo si intende la sovrapposizione di note all'interno di una parte e di una voce specifica, e non quello che scaturisce dall'esecuzione contemporanea di linee monodiche differenti. Allo stesso modo, anche la singola nota viene vista e codificata come accordo, anche se in realtà non si sovrappone verticalmente ad alcun altro suono.

4.4 LE GERARCHIE DI UN DOCUMENTO IEEE 1599

Analizziamo ora la struttura di un doc IEEE 1599 e i tag utilizzati al suo interno.

```
<los>      <!-- descrizione dell'accollatura -->
        <staff_list>       <!-- descrizione delle proprietà dei pentagrammi -->
                <staff id="staff_1">       … </staff>
                <staff id="staff_2">       … </staff>
        </staff_list>
        <part id="flauto" dfstaff_ref="staff_1">      <!-- descrizione delle singole parti -->
                <voice_list>      <!--elenco delle voci della parte -->
                        <voice_item id="voce_flauto"/>
                </voice_list>
                <measure number="1"> … </measure>
                ...
        </part>
        <part id="pianoforte" dfstaff_ref="staff_2">
                <voice_list>
                        <voice_item id="voce1"/>
                </voice_list>
                <measure number="1"> … </measure>
                …
        </part>
        <!-- descrizione delle parti restanti… -->
</los>
```

- **Staff_list:** Rappresenta l'accollatura, ossia l'elenco di tutti gli **Staff**. Uno **Staff_list** deve contenere almeno un pentagramma.

- **Staff:** È il pentagramma. Ciascun pentagramma contiene le informazioni su:
  chiave musicale (clef), indicazione di tempo (time_signature) e armatura di chiave (key_signature).

- **Clef**: Contiene diverse informazioni riguardanti le chiavi. Si tratta di un elemento vuoto, ossia di un elemento che si apre e si chiude in un unico tag:

```
<clef type="G" staff_step="2" octave_num="-8" event_ref="e1">
```

Gli attributi di questo elemento sono:

- **Type**: rappresenta il tipo di chiave. E' stato definito l'intero setticlavio (chiavi di Sol, di Fa e Do), la chiave per le percussioni e quella per le intavolature.

Canto		<clef type="G" staff_step="2" event_ref="clef_1"/>
Soprano		<clef type="C" staff_step="0" event_ref="clef_1"/>
Mezzosoprano		<clef type="C" staff_step="2" event_ref="clef_1"/>
Contralto		<clef type="C" staff_step="4" event_ref="clef_1"/>
Tenore		<clef type="C" staff_step="6" event_ref="clef_1"/>
Baritono		<clef type="F" staff_step="4" event_ref="clef_1"/>
Basso		<clef type="F" staff_step="6" event_ref="clef_1"/>
Intavolatura		<clef type="tabguitar" staff_step="?" event_ref="clef_1"/>
Percussioni		<clef type="percussion" staff_step="0" event_ref="clef_1"/>

- **Staff_step**: serve a definire la posizione della chiave sul pentagramma. Il valore *"0"* rappresenta la prima riga dal basso del pentagramma, *"1"* il primo spazio e via dicendo. Per rappresentare le diverse chiavi del setticlavio che hanno la stessa forma grafica, l'attributo **Type** sarà fisso, mentre il valore di **Staff_step** cambia (*Chiave di soprano Staff_step = "0", Chiave di mezzosoprano Staff_step = "2", Chiave di Tenore Staff_step = "6",* e così via).

- **Octave_num**: questo attributo denota i numerini posti solitamente sopra o sotto le chiavi per definire l'innalzamento o l'abbassamento di una o più ottave delle note scritte nel corrispondente rigo. I valori ammessi sono *8 (ottava sopra), -8 (ottava sotto), 15 (due ottave sopra), -15 (due ottave sotto) o 0 (valore di default).*

- **Event_ref**: costituisce un puntatore per il rispettivo evento presente nello *Spine*. Questo attributo, obbligatoriamente richiesto, è fondamentale nella rappresentazione della partitura.

- **Time_signature**: indica il tempo iniziale del brano. Gli attributi di questo elemento sono **num** e **den**, e indicano il numeratore e il denominatore della frazione utilizzata per le segnature di tempo. Inoltre, alcune combinazioni (ad es. *num = "4"* e *den = "4"*) danno come risultato la scrittura di un simbolo in partitura (la la *C tagliata*): basta aggiungere un attributo **Char_type** e assegnare a questo il valore **"yes"**. Come l'elemento precedente, anche **Time_signature** è un elemento vuoto :

```
<time_signature event_ref="timesig_1" num="4" den="4" vtu_amount="4000"/>
```

- **Key_signature**: definisce l'armatura di chiave. A differenza dei due elementi precedenti, questo non è vuoto, ma prevede un tag di apertura e uno di chiusura:

```
<Key_signature event_ref="Keysig_1">
        <flat_num number="2"/>
</Key_signature>
```

Gli elementi figli di questo elemento sono **flat_num** (per i *bemolli*), **sharp_num** (per i *diesis*) e **natural_num** (per i bequadri). Tutti e tre hanno un attributo **number** che può assumere valori da 1 a 7, in base alla quantità di alterazioni scritte in armatura di chiave. Il numero che verrà inserito, infatti, seguirà l'andamento del circolo delle quinte, cioè lo stesso criterio utilizzato dalla notazione musicale. Per fare un esempio, se **sharp_num = "4"** il brano avrà in chiave *FA#*, *DO#*, *SOL#*, *RE#*,

I tag descritti finora sono tutti definiti all'interno del nodo staff_list, che si occupa sostanzialmente delle caratteristiche riguardanti la tonalità e il tempo di un brano. Tutti gli altri aspetti, invece, vengono trattati all'interno del nodo **Part**: qui troviamo la indicazioni riguardanti le voci (nodo <voice_list>) e quelle riguardanti le battute (nodo <measure>).

- **voice_list**: indica l'elenco delle voci in cui si divide una parte.
- **voice_item**: indica la singola voce e contiene attributi utili alla descrizione della stessa all'interno dell'elemento measure. L'immagine sotto mostra un esempio di come gli attributi id vengono poi riutilizzati all'interno delle singole battute.
- **measure**: descrive tutte le voci che compongono la parte. Ha due attributi:
  - **number**: cioè il numero di battuta
  - **id**: attributo facoltativo utile per identificare una battuta particolare all'interno del brano

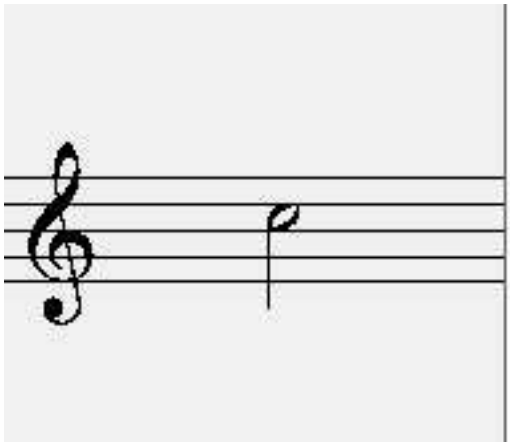
L'elemento **measure** è senza dubbio il più importante, in quanto rappresenta tutti gli eventi musicali simbolici all'interno del brano.

```
<measure number="1">
        <voice voice_item_ref="soprano_voice1">
                <rest event_ref="v1_e0" staff_ref="staff_1">
                        <duration num="1" den="2"/>
                </rest>
                <chord event_ref ="v1_e1">
                        <duration num="1" den="2"/>
                        <notehead staff_ref="staff_1">
                                <pitch step="C" octave="6"/>
                        <notehead/>
                <chord/>
        <voice/>
<measure/>
</ieee1599>
```

Gli elementi figli sono:

- **voice_ref**: riferimento all'id di voice_item (nodo figlio di voice_list all'interno di part) che indica la voce a cui la battuta si riferisce (es. nell'immagine sopra sono "manodx" e "manosx").
- **rest**: elemento opzionale usato per inserire pause nella battuta. Gli attributi di questo nodo sono i riferimenti per il posizionamento nello spine (**event_ref**) e nel pentagramma (**staff_ref**).
- **duration**: indica la durata dell'evento musicale tramite i due attributi **num** e **den.** Questo elemento è presente come elemento figlio sia all'interno di **rest**, sia all'interno di **chord**.

- **chord**: viene usato per inserire gli eventi nota e accordo nella battuta. E' caratterizzato dall'attributo **event_ref** che serve de riferimento allo spine e da due nodi figli (duration e notehead).
- **notehead:** rappresenta, di fatto, la nota. Oltre ad avere il consueto attributo per il riferimento al pentagramma (staff_ref), presenta il nodo figlio **pitch**, che definisce l'altezza della nota (attributo **step**) e l'ottava in cui si colloca (attributo **octave**):
  - **step** può assumere valori char che vanno dalla lettera A alla lettera G (usando, quindi, la nomenclatura in lettere delle note); per cui ad A corrisponde la nota LA, a B la nota SI, a C la nota DO, e via dicendo.
  - **Octave** può assumere valori da 1 a 7, ricoprendo tutte le ottave in cui la singola nota può collocarsi. Le ottave maggiormente usate in musica vanno dalla numero 4 alla numero 6.



```
<chord event_ref="v1_e1">
  <duration num="1" den="2" />
  <notehead staff_ref="staff_1">
    <pitch step="C" octave="6" />
  </notehead>
</chord>
```

Fig.26 – codice XML di una nota e rispettiva rappresentazione grafica

Capitolo 5: Analisi tecnico - funzionale dell'applicazione "Piano Roll"

L'applicazione "Piano Roll" è stata sviluppata con il supporto di diverse tecnologie: dal linguaggio di programmazione a oggetti C#, che sta alla base di tutto, all'implementazione XML tramite struttura DOM.
La sua funzione principale è quella di estrarre una serie di informazioni presenti all'interno dei documenti IEEE1599 e di trasformarle in maniera tale da poter fornire una corrispondente rappresentazione in formato piano roll. Quest'ultima consiste in un disegno grafico realizzato sul form (cioè la finestra principale dell'applicazione), pertanto, i valori che verranno presi in considerazione saranno convertiti in pixel.

Dopo una prima definizione dell'interfaccia, che ha permesso di studiare le modalità da utilizzare nello sviluppo del programma, si è passati alla progettazione vera e propria dell'applicazione. Come si è visto nel capitolo precedente, i documenti IEEE1599 sono caratterizzati da una quantità considerevole di informazioni musicali; tuttavia, gli elementi strettamente necessari alla rappresentazione delle note sul piano roll sono tre: l'altezza, la durata e l'istante in cui ognuna si presenta all'interno del brano. Le prime due informazioni sono raggruppate all'interno del layer LOS, mentre invece l'aspetto temporale, cioè l'ordine con cui gli eventi si presentano all'interno della partitura, coincide alla sequenza di nodi definita nel layer SPINE. I valori di ogni evento dislocati nei due layer vengono collegati tramite dei riferimenti; inoltre, utilizzano unità di misura eterogenee, che vanno dal VTU (Virtual Time Unit) alla nomenclatura delle note. Per questi motivi sarà inevitabile considerare, oltre alle informazioni di pura natura musicale, anche alcuni aspetti legati alla sintassi dei documenti IEEE1599.

Nel corso del seguente capitolo verranno analizzate: in un primo momento le modalità utilizzate per estrarre le informazioni dai documenti IEEE1599 e, successivamente, le componenti dell'interfaccia grafica.

5.1 METODI DI ACCESSO AI NODI XML

L'accesso ai nodi dei documenti XML IEEE1599 avviene tramite delle query Xpath su una struttura DOM di questo tipo:

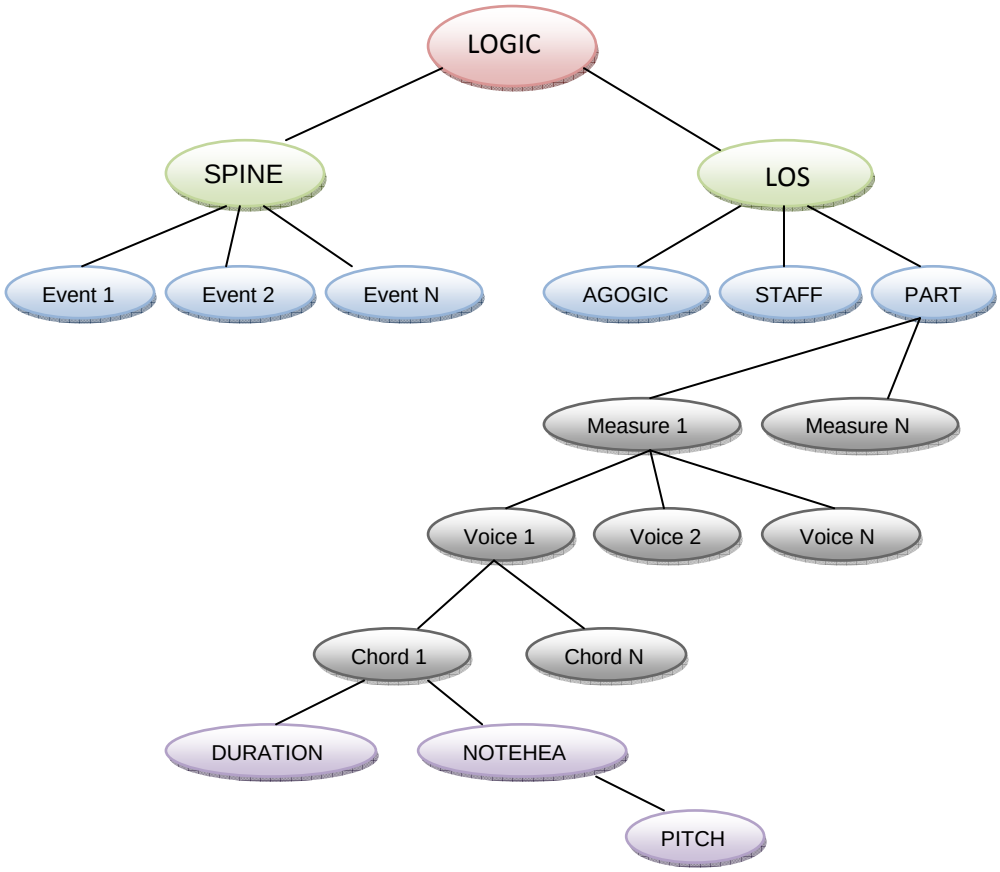


Fig.27 – struttura DOM del layer logic

La maggior parte delle informazioni utilizzate dall'applicazione "Piano Roll" verranno estratte partendo dalla selezione di tutti i nodi che presentano lo stesso nome, effettuata tramite query Xpath. Questi nodi formeranno una lista di nodi xml e, per ogni nodo riscontrato, verranno eseguite diverse operazioni. I valori estratti da ogni nodi verranno quindi assegnati a delle variabili definite all'interno di cicli for e foreach.

Questo sistema di selezione comporta però un problema fondamentale: le variabili, con i relativi valori, non sono accessibili all'esterno dei cicli for. Una possibile soluzione che le rende accessibili è quella di inserire i valori all'interno di un array ma, dato che non è possibile conoscere a priori la quantità di nodi presenti in ogni lista, anche questa opzione è stata scartata. La soluzione al problema sono invece le liste: oltre a non richiede la definizione delle proprie dimensione nel momento in cui vengono inizializzate, le liste permettono di accedere alle variabili dall'esterno dei cicli for. Per questi motivi, quest'ultime verranno spesso utilizzate.

Inizialmente il codice è stato sviluppato prendendo in input un documento IEEE1599 che rappresenta il brano Gottes Match di Ludvig Van Beethoven.

```
XmlDocument xmldoc = new XmlDocument();
xmldoc.Load("C:/Users/Daniele/Desktop/esempi_partiture/GottesMacht.xml");
```

Nella fase di ottimizzazione del programma, invece, questa porzione di codice è stata modificata in modo tale da abilitare l'applicazione a ricevere in input qualsiasi documento in formato IEEE 1599.

5.2 DEFINIZIONE DELLE NOTE

Nell'applicazione piano roll, le note vengono rappresentate tramite dei rettangoli disegnati all'interno di una griglia con il metodo DrawRectangle(). Seguendo la legenda della rappresentazione in formato piano roll, le 4 variabili che definiscono le coordinate e le dimensioni di ogni rettangolo/nota sono:

- Coordinata x = istante in cui si verifica l'evento (ad es. la prima nota della terza battuta);
- Coordinata y = altezza dell'evento (ad es. nota do dell'ottava numero 3);
- Larghezza del rettangolo = durata dell'evento;
- Altezza del rettangolo = altezza del quadratino che compone la griglia del piano roll.

L'altezza del rettangolo è un valore che rimane fisso per tutti gli eventi e viene quindi definito solo una volta (20 pixel); tutti gli altri valori, invece, variano di evento in evento. Per gestire in modo ordinato tutti i dati di un singolo evento, ogni valore, dopo essere stato convertito in pixel, viene inserito all'interno di una lista dedicata.

Fig.28 – liste per la definizione delle note

Come mostra la figura precedente, le liste utilizzate sono tre:

- RettX, con tutte le coordinate x
- Pithc, con tutte le coordinate y
- Durate , con tutte le larghezza del rettangolo

Una volta riempite le liste, il metodo DrawRectangle prende i valori dalle stesse scorrendole dall'inizio alla fine (Lista.Cuont – 1) attraverso un ciclo for. Successivamente, con i metodi DrawRectangle e FillRectangle, i rettangoli vengono disegnati, colorati all'interno (diventando, così, delle barre) e collocati in posizioni specifiche all'interno della griglia.

```
//Rettangoli note
for (int r = 0; r <= RettX.Count - 1; r++)
  {
    Rectangle eventi = new Rectangle(RettX[r], Pitch[r], Durate[r], 20);
    Color MyColor;
    ColorBar = Color.Green;
    MyColor = Color.FromArgb(255, ColorBar);
    SolidBrush myBrush = new SolidBrush(MyColor);
    g.FillRectangle(myBrush, eventi);
    g.DrawRectangle(nota, eventi);
  }
```

Il punto di partenza per l'estrazione dei dati e la conversione in pixel è una XmlNodelist che contiene tutti gli eventi dello spine. Ad ognuno di questi nodi viene estrapolato il valore dell'attributo ***id***, che viene assegnato ad una variabile String chiamata *Event_ref* utilizzata come filtro per la selezione dei nodi ***chord*** e ***rest***.

```
//coordinate rettangolini
XmlNodeList xTiming = xmldoc.SelectNodes("/ieee1599/logic/spine/event");
 foreach (XmlNode xT in xTiming)
   {
      String Event_ref = xT.Attributes["id"].Value.ToString();

  XmlNodeList xChord =
  xmldoc.SelectNodes("/ieee1599/logic/los/part/measure/
     voice/chord[@event_ref='" + Event_ref + "']/notehead/pitch");
```

Il codice mostra che la lista xChord viene riempita di nodi chord che hanno come valore dell'attributo event_ref una stringa che corrisponde al valore dell'attributo ***id*** corrente (variabile Event_ref). La stessa cosa avviene per i nodi rest.

- **DEFINIZIONE DELLE ALTEZZE**

L'altezza delle note viene definita all'interno dell'applicazione "Piano Roll" sull'asse x. La griglia è stata suddivisa con un'unità di misura fissa (quadratini da 20 pixel) e ad ogni unità è stata associata l'alternativa tra due immagini: un tasto del pianoforte (per indicare i tasti bianchi) oppure la congiunzione tra due tasti (per i tasti neri).

L'esempio riportato nella pagina successiva mostra la scrittura delle note localizzate sui tasti neri (primo riquadro della griglia) e sui tasti bianchi di un pianoforte (secondo riquadro).

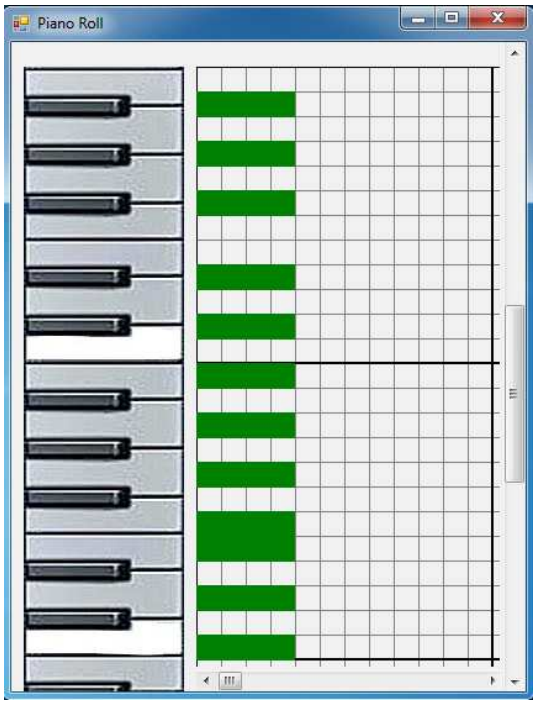


Fig.29 – Indicazione dell'altezza in "Piano Roll"

Come si può vedere, in questo modo è possibile rappresentare tutte le note riscontrabili all'intero di una partitura: sia naturali che con alterazioni semplici o doppie. La linea nera orizzontale in grassetto segnala il cambio di ottava.

Il codice che definisce il valore dell'altezza di ogni nota e traccia le barre nella posizione esatta inizia dalla selezione dei nodi pitch. Per ogni nodo ***pitch*** vengono prelevati i valori degli attributi ***octave*** e ***step***, che indicano rispettivamente l'ottava in cui la nota si colloca e la sua altezza. Tuttavia, questi due dati non sono sufficienti a rappresentare tutte le note possibili, in quanto resterebbero escluse quelle caratterizzate da alterazioni (diesis e bemolli). Di conseguenza, bisogna prendere in considerazione anche l'attributo opzionale chiamato ***actual_accidental***, che indica appunto la presenza di un'alterazione.

Il principio utilizzato per determinare il valore delle altezze in pixel consiste nel sommare, per ogni nota corrente, il valore degli attributi octave e step dopo che questi abbiano subito a loro volta una conversione in pixel. Vediamo in che modo.

L'attributo ***octave*** può assumere valori che vanno da 1 a 7 (cioè le ottave di un pianoforte) e, visto che la coordinata Y del piano roll parte da un valore uguale a zero, è necessario trovare un sistema che permetta di poter disegnare le barre in tutto il range di valori possibili. Stando alle dimensioni del piano roll, ogni ottava è composta da 240 pixel, cioè l'altezza di un quadratino (20) moltiplicata per il numero di note presenti all'interno di un ottava (12). Il range di valori del piano roll va quindi da 0 (ultimo tasto della tastiera) a 1680 (primo tasto, cioè 240 * 7).

La soluzione che è stata adottata consiste nel sottrarre al valore di octave sette unità, ossia il valore massimo che quest'ultimo può assumere, ottenendo un risultato che avrà sempre un valore negativo o uguale a zero. Il tutto viene poi moltiplicato per -240, in modo da rendere il valore nuovamente positivo e poter ricoprire l'intero range (compreso lo 0: (7 - 7) * – 240).

```
int oct;
oct = (Int32.Parse(xC.Attributes["octave"].Value.ToString()) - 7)* -240;
```

Anche l'attributo step può assumere sette valori diversi (dalla A alla G), mentre invece actual_accidental ne può assumere cinque (cioè tutte le alterazioni possibili). A differenza delle ottave, i dati riguardanti questi due attributi non sono numeri, ma stringhe di caratteri, ed è quindi necessario convertire i valori in numeri. La conversione viene fatta basandosi sul seguente schema.

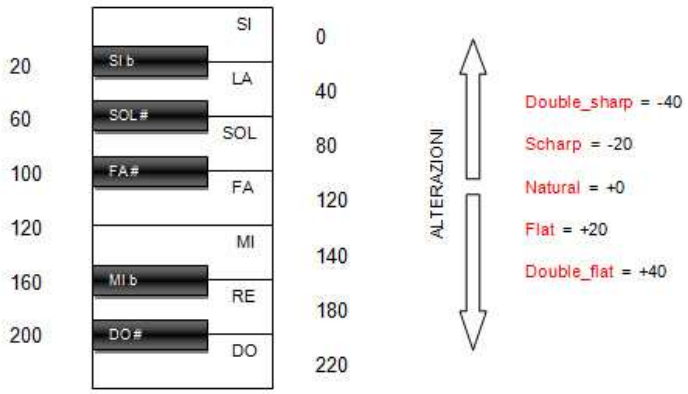


Fig.30 – conversione altezze – pixel

Il codice corrispondente inizializza il valore della variabile *altezza* 0 e valuta, tramite la funzione if/else, il valore dell'attributo ***actual_accidental***; in base a quest'ultimo, viene definita un'altra variabile di tipo string chiamata *alterazione*: se è uguale a null (cioè actual_accidental non è presente) alterazione è posta uguale a 0, se invece actual_accidental è presente il valore indicato (duoble-sharp/flat, sharp/flat o natural) diventa il valore di *alterazione*.

```
int altezza = 0;
string alterazione;
if (xC.Attributes["actual_accidental"] == null)
 {
    alterazione = "0";
 }
else
 {
     alterazione = xC.Attributes["actual_accidental"].Value;
 }
```

Utilizzando sempre delle funzioni if/else, viene assegnato un determinato valore alla variabile altezza, che varia in base al valore dell'attributo step e alla presenza, o meno, di alterazioni. Di seguito è riportato l'esempio del SI (attributo step = 'B').

```
if (xC.Attributes["step"].Value == "B")
   {
      //nessuna alterazioni
      altezza = 0;
     //possibili alterazioni
    if (alterazione == "sharp")
       {altezza = - 20;}
    if (alterazione == "flat")
       {altezza = 20;}
    if (alterazione == "double_sharp")
       {altezza = - 40;}
    if (alterazione == "double_flat")
       {altezza = 40;}
    if (alterazione == "natural")
       {altezza = 0;}
  }
```

Sommando per ogni nota considerata il valore di octave e di step (trasformati in pixel) si ottiene la coordinata Y del rettangolo che, come già detto, corrisponde all'altezza esatta della nota. Quest'ultimo viene quindi aggiunto alla lista chiamata Pitch.

```
int pit = altezza + oct;
Pitch.Add(pit);
```

Volendo fare una verifica per il primo e ultimo tasto della tastiera possiamo notare che i valori calcolati in questo modo ricoprono l'intero range del piano roll.
Il primo tasto è il DO in ottava 1; nel codice il valore in pixel del DO naturale è uguale a 220, quindi il calcolo che viene fatto è: ((1-7) * – 240) + 220 = 1440 + 220 = 1660. Aggiungendo 20 pixel di altezza della barra si ottiene il valore massimo, cioè 1680. L'ultimo tasto è invece il SI in ottava 7: ((7-7) * – 240) + 0 = 0.
Per quanto riguarda l'altezza delle pause si è optato per l'attribuzione di un valore fittizio, per cui ogni barra che presenterà quel determinato valore verrà cancellata tramite il metodo Clear().

- **DEFINIZIONE DELLE DURATE**

La definizione della durata di ogni evento avviene attraverso un confronto con un altro valore. Quest'ultimo rappresenta un metro di misurazione che corrisponde alla porzione più piccola del piano roll (il singolo quadratino) e, allo stesso tempo, alla durata minima presente all'interno del brano.
In questo modo, ogni nota riscontrata viene suddivisa in durate minime così da ottenere il numero esatto di quadratini da colorare.

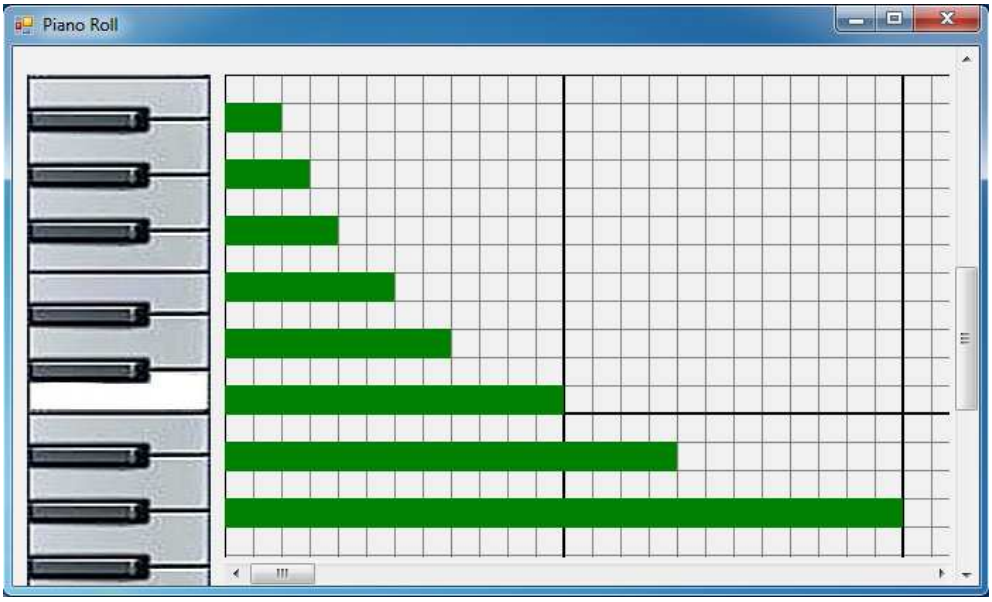


Fig.31 – Indicazione della durata in “Piano Roll”

La ricerca della durata minima viene effettuata analizzando tutte le durate dei nodi ***chord*** e ***rest***, confrontate a loro volta con una durata minima inizializzata ad un valore di 1/1 (cioè il massimo valore riscontrabile). Nel momento in la durata di un evento è inferiore alla durata minima, la prima viene subito assegnata alla struttura *Minima*; ripetendo questo procedimento per ogni durata si arriva a determinare il valore più piccolo.
Di seguito viene mostrato il codice che svolge le funzioni appena citate riguardante i nodi rest.

```
 XmlNodeList xRestdur =
xmldoc.SelectNodes("/ieee1599/logic/los/part/measure/voice/rest/duration");
 foreach (XmlNode xR in xRestdur)
   {
      int num = Int32.Parse(xR.Attributes["num"].Value.ToString());
      int den = Int32.Parse(xR.Attributes["den"].Value.ToString());
      float dur = ((float)num / (float)den);

      if (dur < ((float)Minima.num / Minima.den))
        {
           Minima.num = num;
           Minima.den = den;
        }

   }
```

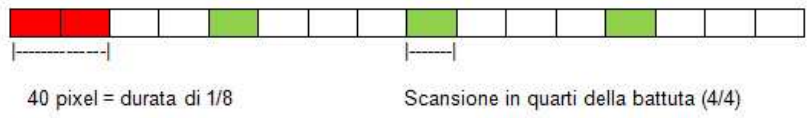
Una volta stabilita la durata minima si passa alla definizione delle durate vera e propria. Questa avviene in due modi differenti: per i nodi rest viene semplicemente estratto l’attributo duration e convertito in pixel; per i nodi chord bisogna invece considerare anche altri aspetti. Nella pagina successiva viene riportato il codice utilizzato per gestire la durata delle pause.

```
XmlNodeList xRest =
xmldoc.SelectNodes("/ieee1599/logic/los/part/measure/voice/
   rest[@event_ref='" + Event_ref + "']/duration");
   foreach (XmlNode xR in xRest)
     {
        int num = Int32.Parse(xR.Attributes["num"].Value.ToString());
        int den = Int32.Parse(xR.Attributes["den"].Value.ToString());
        float dur = ((float)num / (float)den);

        DurEvent = (int)(dur / Rettmin) * PXL; //Rettmin = Durata Minima
         Durate.Add(DurEvent);

     }
```

Dopo aver preso i valori degli attributi **num** e **den** del nodo **duration** e averli assegnati ad una variabile di tipo float chiamata *dur*, la durata dell'evento in pixel corrisponde a: (dur / Rettmin) * PXL. Se ad esempio la durata minima è 1/16 e la durata del nodo corrente è 1/8, il valore in pixel è uguale a 40 ((1/8)/(1/16)) * 20), cioè due quadratini.



Nel caso dei nodi chord il codice utilizzato risulta diverso: mentre la durata delle pause ha sempre un valore fisso (definito dalla coppia di variabili num e den), la durata dei nodi chord può variare dalla presenza di nodi supplementari che fanno parte del nodo stesso. Di conseguenza, il valore espresso nel nodo **duration** non corrisponde sempre alla durata effettiva della nota:

- il nodo ***augmentation_dots***, che rappresenta il punto di valore, aumenta la durata della nota per un valore che corrisponde alla metà della durata della nota stessa;
- il nodo ***tuplet_ratio***, che corrisponde alla terzina, decrementa il valore della singola nota.

Per questo motivo la durata dell'evento nota viene gestita in due modi differenti:

- Il punto di valore

Nel caso in cui è presente o meno il nodo augmentation_dots, la funzione che determina il valore della variabile DurEvent è lo stessa di quella utilizzata per i nodi Rest, con l' aggiunta di una variabile chaimata p_val che viene sommata a alla variabile dur.

```
DurEvent = (int)((dur + p_val) / Rettmin) * PXL;
```

Il valore iniziale di p_val è uguala a 0; in questo modo, nel caso in cui il punto di valore non fosse presente, il valore della nota rimane invariato. Il nuovo valore rappresenta infatti la durata del punto: attraverso un ciclo for, ripetuto per un numero di volte pari alla quantità di punti di valore che la nota presenta, viene assegnata a p_val la durata della nota stessa diviso due.

```
if (xC.ParentNode.ChildNodes.Item(1).Name.Equals("augmentation_dots"))
 {
   float point = dur;
   int p_num = Int32.Parse(xC.ParentNode.ChildNodes.Item(1).
   Attributes["number"].Value.ToString());
    for (int w = 0; w < p_num; w++)
     {
        point = point / 2;
        p_val += point;
     }
 }
```

Se ad esempio abbiamo l'attributo number di augmentation_dots uguale a 2 (cioè due punti di valore) e una nota di 1/4 (assegnata alla variabile point), il valore di p_val è:

dopo il primo ciclo for

point = (1/4) / 2 = 1/8 con p_val = 0 + 1/8 = 1/8

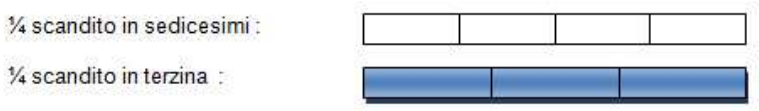
dopo il secondo (e ultimo) ciclo for

point = (1/8) / 2 = 1/16 con p_val = 1/8 + 1/16 = 3/16

La durata dell'evento è quindi ((1/4 + 3/16) / (1/16)) * 20 = 7 * 20 = 140 pixel (7 quadratini)

- La terzina

Il calcolo del valore in pixel di una nota che fa parte di una terzina è invece molto più articolato. Si tratta di un valore irregolare, che non segue cioè le regole standard delle durate. Ad esempio una terzina composta da tre note del valore di 1/8 ha una durata complessiva di 1/4 (che generalmente è composta, invece, da due ottave). Questo vuol dire che in una griglia in cui 1/4 è composto da 4 quadratini (da 1/16) dovrà essere colorato un quadratino e 1/3 del quadratino successivo.



Per attribuire ad una nota il valore di terzina viene adottato il seguente codice

```
 else if
(xC.ParentNode.ParentNode.ChildNodes.Item(1).Name.Equals("tuplet_ratio"))
  {
     DurEvent = (int)((dur - Rettmin / Rettmin) * (PXL)) + (PXL / 3);
     Durate.Add(DurEvent);
  }
```

Tenendo conto dell'esempio precedente e considerando il valore di un quadratino uguale a 21 pixel, il numero di pixel per una nota di 1/8 all'interno di una terzina è:

((1/8 – 1/16) / 1/16) * 21) + (21/3) = (1 * 21) + 7 = 28 (cioè 1 quadratino + 1 qudratino/3)

- **DEFINIZIONE DELLE COORDINATE X**

Il valore della coordinata X dei rettangoli rappresenta l'istante in cui l'evento si verifica nel tempo. Questo valore è specificato nei documenti IEEE 1599 dall'attributo ***vtu*** del nodo ***event***, nodo figlio di ***spine***.

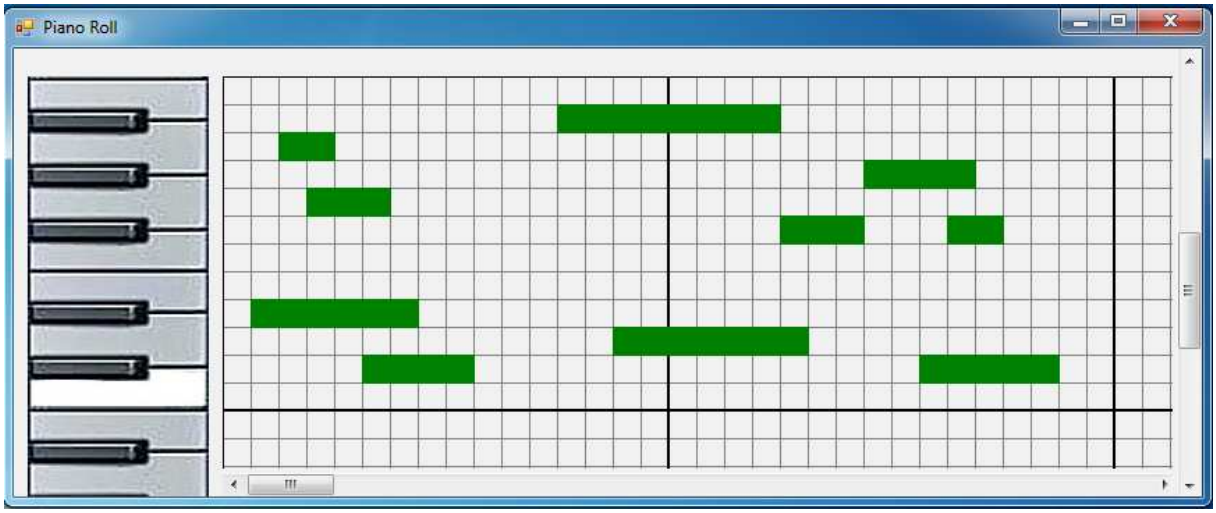


Fig.32 – Temporizzazione delle note in "Piano Roll"

Nel capitolo precedente è stato mostrato il principio utilizzato nei documenti IEEE 1599 per attribuire il valore in vtu ad ogni evento. Per chiarire meglio il concetto, di seguto viene ripresentato il criterio di assegnazione affiancato da un esempio della sua applicazione.

All'interno del nodo ***time_indication*** del ***los*** è presente l'attributo ***vtu_amount***, che indica il numero di vtu con il quale ogni battuta del brano viene suddivisa.
Il primo evento di una partitura (cioè il primo nodo event dello spine) ha un vtu pari a zero, che indica l'inizio del brano e l'assenza di eventi precedenti. Tutti gli eventi di parti diverse che coincidono temporalmente hanno anch'essi un vtu uguale a zero, ad indicare la contemporaneità con cui si presentano. Nel momento in cui si verifica un evento che invece non coincide con l'evento precedente (in questo caso il primo), il vtu che gli viene aseggnato corrisponde alla distanza (sempre in termini di vtu) che intercorre tra i due.

Poniamo il caso di avere un brano in 4/4, con un vtu_amount uguale a 8. Se nella prima battuta ci sono due note da 2/4 il loro valore in vtu è: 0 per la prima nota, perché non è preceduta da altri eventi, e 4 per la seconda (cioè 8 diviso 2 note). Se consideriamo invece la seconda battuta, e ipotizziamo che sia composta da quattro note da 1/4, il vtu della prima nota è 4 (perché la nota che precede è di 2/4) e quello di tutte le altre è 2 (cioè 8 diviso 4 note). Se infine, in corrispondenza dell'ultima nota della seconda battuta, subentra una seconda parte con una nota di 1/4, il valore vtu di quest'ultima sarà 0, perché coincide con l'ultimo evento considerato.

Tornando all'applicazione, vista le metodologia di selezione nodi utilizzata finora, non è possibile applicare il principio di assegnazione vtu sopra citato alle coordinate x: mentre nello spine ad ogni evento corrisponde ad una o più note (accordo), ogni elemento delle liste fin qui compilate corrisponde solo ad una singola nota o pausa. Per questo motivo, sono state utilizzate due diverse modalità di definizione: una per tutte le pause, che, essendo singoli eventi, possono andare a recuperare il valore vtu corrispondente nello spine, ed una per tutte le note, con le quali viene adottata una strategia più complessa.

I riferimenti in vtu estratti per le pause vengono assegnati ad una variabile time che definisce una lista di nome Time.

```
int time =
Int32.Parse(xR.ParentNode.ParentNode.ParentNode.ParentNode.ParentNode.
  ParentNode.SelectSingleNode("spine/event[@id='" + Event_ref + "']")
  .Attributes["timing"].Value.ToString());
Time.Add(time);
```

Per quanto riguarda le note, invece, viene generata una XmlNodelist di nodi chord. Per ogni nodo inserito in lista, viene creata una XmlNodelist di nodi notehead, che indica il numero esatto di note di cui è caratterizzato ogni accordo. Per il primo nodo di quest'ultima lista, cioè la prima nota dell'accordo, viene assegnato alla veriabile time il valore in vtu corrispondente nello spine, mentre a tutti gli altri nodi (cioè le altre note che completano l'accordo), il valore di time è posto uguale a zero. Ogni variabile time viene quindi aggiunta nella lista Time.

```
XmlNodeList xCount =
xmldoc.SelectNodes("/ieee1599/logic/los/part/measure/voice/
chord[@event_ref='" + Event_ref + "']");
foreach (XmlNode xc in xCount)
 {
     XmlNodeList Noteheads = xc.SelectNodes("notehead");
    //se il nodo chord ha solo una nota viene preso il vtu
     if (Noteheads.Count == 1)
     {
        int time = Int32.Parse(xc.ParentNode.ParentNode.ParentNode.
        SelectSingleNode("spine/event[@id='" +Event_ref + "']").
        Attributes["timing"].Value.ToString());
        Time.Add(time);
     }
    //se invece ha più di una nota
     else if (Noteheads.Count != 1)
      {
          //conta il numero di note (cioe n° di nodi in lista)
         for (int i = 0; i <= Noteheads.Count - 1; i++)
          {
             //alla prima assegna il valore di vtu nello spine
             if (i == 0)
              {
                int time = Int32.Parse(xc.ParentNode.ParentNode.
               SelectSingleNode("spine/event[@id='" + Event_ref + "']")
                .Attributes["timing"].Value.ToString());
                Time.Add(time);
              }
```

```
            //a tutte le altre assegna il valore 0
              else if (i != 0)
               {
                  int time = 0;
                  Time.Add(time);
               }
          }
      }
  }
```

Dopo aver completato tutti i cicli foreach di note e pause, il numero di valori inseriti nella lista Time dall'applicazione sarà uguale al numero di elementi contenuti nelle liste Pitch e Durate definite in precedenza, in quanto gli eventi con più note sono stati suddivisi per un numero pari alle note di cui erano caratterizzati. Inoltre, grazie a questo sistema, è stato possibile riproporre lo stesso criterio di assegnazione dei vtu utilizzato dai documenti IEEE 1599.

A questo punto, ogni elemento di Time viene trasformato in pixel utilizzando due variabili: ***Battuta***, che indica la dimensione di ogni singola battuta in termini di pixel, e ***vtu_amount***, che contiene il valore del vtu generale. La conversione avviene attraverso una funzione if/else, dove i valori in lista uguali a zero vengono mantenuti tali (generendo una variabile X=0) e tutti gli altri modificati. Nel momento in cui viene individuato in lista un vtu diverso da zero, l'applicazione assegna alla variabile X un valore uguale a : Battuta / (valore del vtu-generale/vtu del nodo corrente) che, di fatto, corrisponde alla durata dell'evento preso in considerazione. Ogni variabile X viene quindi inserita nella lista CordX.

```
    //conversione VTU -> Pixel
      int pix = valVTU[0];
      for (int t = 0; t <= Time.Count - 1; t++)
       {
          if (Time[t] == 0) //se vtu = 0
           {
              int X = 0;
              CordX.Add(X);
           }
          if (Time[t] != 0) //se vtu != 0
           {
               int A = pix / Time[t];
               int X = (int)Battuta / A;
               CordX.Add(X);
           }
      }
```

Per verificare la corrispondenza tra durata evento e variabile X, di seguito viene riportato un esempio di conversione del valore vtu di una nota di 2/4.

La definizione di durata in pixel, descritta nel paragrafo precedente, di una nota di 2/4 con durata minima di 1/16 è uguale a 160, cioè ((2/4) / (1/16)) * 20. Sostituendo in questa operazione il valore 2/4 con 4/4 si ottiene invece il valore in pixel della battuta che si sta considerando, cioè 320.
La variabile X che viene generata per la nota di 2/4 con vtu uguale a 4 è invece: 320 / (8/4) = 160.
Come si può notare, utilizzando questi algoritmi la corrispondenza durata – variabile X è garantita.

Questa prima conversione ci ha permesso di trasformare i valori vtu degli eventi nel corrispondente valore in pixel, ma non è sufficiente a fornirci i valori della coordinata x. Per avere una rappresentazione corretta degli eventi sul piano roll dal punto di vista temporale, è infatti necessario avere un valore della coordinata x che venga di volta in volta incrementato e che non torni a zero in caso di eventi coincidenti. La soluzione adottata per questo problema consiste nel sommare ogni valore corrente a quello precedente, in modo tale che un evento successivo ad un altro abbia sempre un valore maggiore o uguale al precedente. Il codice che eseque queste operazioni è riportato nella pagina seguente.

```
int Y = CordX[0];
RettX.Add(Y);

 //genera lista di coordinate x
 for (int c = 0; c <= CordX.Count - 1; c++)
  {
     int b = c + 1;
     if (b > CordX.Count)
     {
     }
     else if (b < CordX.Count)
     {
        int X = RettX[c] + CordX[b];
        RettX.Add(X);
     }
  }
```

Nella prime due righe viene assegnata ad una variabile Y il valore vtu del primo evento (collocato in posizione 0 nella lista CordX) e eggiunta nella lista RettX, la quale conterrà tutte le coordinate x. Successivamente, attraverso un ciclo for che scorre tutta la lista CordX, per ogni vtu viene definita:

- una variabile b con il valore della posizione corrente + 1 della lista CordX;
- una variabile X che somma il valore vtu situato nella posizione corrente di RettX con il valore vtu di CordX in posizione b;

Ad ogni ciclo viene quindi incrementata la lista RettX. Una volta completata la lista, il metodo che crea i rettangoli nota ha tutti gli elementi necessari per disegnare in modo corretto tutti gli eventi sulla griglia.

5.3 DEFINIZIONE DELL'INTERFACCIA

L'interfaccia del programma è caratterizzata da due tipologie di componenti: una centrale, rappresentata da una griglia sulla quale vengono tracciate le note in formato piano roll, ed altri elementi di contorno che specificano la sintassi della griglia stessa . Ogni componente dell'interfaccia è stata aggiunta al form (finestra principale dell'applicazione) con i toll messi a disposizione dal software Visual Studio Professional. Attraverso l' implementazione di metodi e funzioni C#, sono state

quindi definite le azioni che questi oggetti avrebbero dovuto compiere per raggiungere lo scopo prefissato dall'applicazione.

Per realizzare un applicazione in grado di rappresentare graficamente un brano in formato piano roll, le componenti principali di cui dovrà disporre necessariamente sono tre: l'immagine di una tastiera, una griglia associata ad essa e delle barre che indicano la presenza delle note.

L'***immagine della tastiera*** è stata implementata abilitando la proprietà *Image* dello strumento *picturebox,* grazie alla quale è stato possibile caricare sul form un'immagine che rappresenta tutti i tasti di un pianoforte.

Per quanto riguarda la ***griglia***, invece, si è optato per un disegno manuale. A differenza della tastiera, che ha sempre una lunghezza fissa di 88 tasti, la griglia ha una dimensione variabile dalla quantità di note presenti nel brano e, per questo motivo, non è possibile utilizzare sempre la stessa immagine. Questo disegno è stato fatto all'interno di una seconda picturebox, abilitandone l'evento *Paint* e scrivendo al suo interno il metodo DrawRectangle().

```
private void pictureBox2_Paint(object sender, PaintEventArgs e)
 {
    Graphics g = e.Graphics;
    Pen griglia = new Pen(Color.Gray, 1);

    for (int x = 0; x <= Roll ; x += PXL)
     {
        for (int y = 0; y <= 1680; y += PXL)
         {
             g.DrawRectangle(griglia, new Rectangle(x, y, PXL, PXL));
         }
           g.DrawRectangle(griglia, new Rectangle(x, 0, PXL, PXL));
     }
```

In questa porzione di codice è stato assegnato l'evento Paint ad una variabile di tipo Graphics, cioè g, in modo tale da poterla utilizzare nel momento i cui si richiama il metodo DrawRectangle; è stato quindi definito l'oggetto Pen, cioè la penna che disegna la griglia (colore Gray, spessore 1), e i singoli rettangoli, implementati attraverso due cicli for annidati che assegnano ad ognuno di questi una dimensione in pixel 20x20 (altezza e larghezza) e coordinate X/Y multiple di 20. Come già visto, anche il disegno delle ***note*** viene effettuato con questo metodo, con la differenza che il valore di ogni coordinata viene stabilito dai parametri forniti da ogni singola nota (cioè tempo, altezza e durata).

Per evitare che l'interfaccia risultasse illeggibile, sono state utilizzate per tastiera e griglia delle dimensioni abbastanza elevate. Questa scelta ha reso necessaria l'aggiunta di due ***barre di scorrimento***, una orizzontale ed una verticale, in modo tale da consentire all'utente di visualizzare ogni parte della griglia.
Il comportamento di ogni barra è stato definito all'interno dell'evento Scroll, che permette di modificare l'area da visualizzare ad ogni spostamento della barra stessa. Nel caso della barra verticale vengono prese in considerazione entrambe le picturebox, che si spostano di 20 pixel ad ogni evento scroll; la barra orizzontale invece, sposta solo la griglia, sempre per un valore di 20 pixel.

Il codice è il seguente:

```
 private void vScrollBar1_Scroll(object sender, ScrollEventArgs e)
 {
 int x = 10;
 int y = 20;

 pictureBox1.Location = new Point(x, y + vScrollBar1.Value*-20);
 pictureBox2.Location = new Point(hScrollBar1.Value*-20,vScrollBar1.Value*-
20);

 }

 private void hScrollBar1_Scroll(object sender, ScrollEventArgs e)
 {
 pictureBox2.Location = new Point(hScrollBar1.Value*-
20,vScrollBar1.Value*-20);
 }
```

Le due picturebox, per evitare che si sovrapponessero ad altri elementi del form (o tra di loro), sono state posizionate su un ***Panel***. In questo modo, nel momento in cui una picturebox oltrepassa il margine del pannello, la porzione di immagine che rimane fuori dal Panel viene nascosta dall'applicazione.

A questo punto, l'interfaccia del programma ha una sua prima forma:

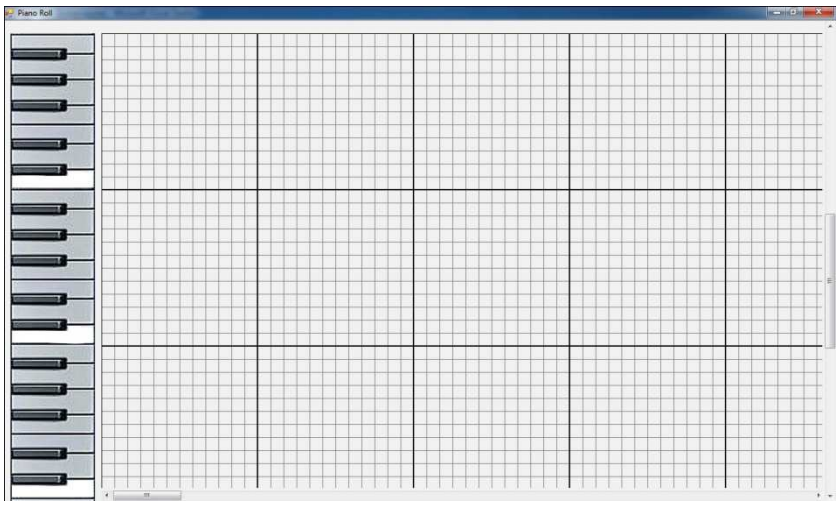


Fig.33 – Interfaccia grafica iniziale dell'applicazione "Piano Roll".

La dimensione del piano roll sull'asse X viene definita in base ai valori estratti da ogni documento IEEE1599, in particolare dal numero di battute di cui ognuno è caratterizzato. Di seguito vengono mostrate le variabili utilizzate per determinare la lunghezza del piano con un esempio di applicazione.

```
//valore che viene attribuito al singolo quadratino
 Rettmin = (float)Minima.num / (float)Minima.den;
 //valore che indica il n° di quadratini di ogni battuta
 Rettn = ((float)Massima.num / (float)Massima.den) / Rettmin;
```

```
//valore in pixel attribuito alla battuta
Battuta = Rettn * PXL;
//n° di battute del piano roll
 Roll = (int)Battuta * BatCount;
```

Minima e Massima rappresentano due strutture all'interno delle quali sono state memorizzate rispetivamente la durata minima riscontrabile e il tempo del brano, entrambe espresse in frazioni (num/den). Dividendo questi due valori si ottiene il numero di quadratini della battuta, ad esempio con Rettmin = 1/16 e Massima = 4/4 si ottiene un valore pari a 16 (variabile Rettn). Le variabili PXL e BatCount rappresentano invece il numero di pixel di ogni quadratino del piano roll (fissato a 20) e il numero totale di battute del brano. Proseguendo con l'esempio, il valore in pixel della battuta è 320 (16*20) e, con BatCount = 18, la dimensione del piano roll sull'asse X è 5760 pixel (320*18).

Le ultime componenti che caratterizzano l'interfaccia dell'applicazione sono:

- sei strumenti ***label***, ognuno dei quali fornisce indicazioni riguardanti il numero di battute, il tempo e il valore di quantizzazione del brano;
- uno strumento ***textbox*** che visualizza la directory del documento IEEE1599 su cui verrà effettuata l'operazione di estrazione dati;
- due strumenti ***button***, uno usato per caricare il file IEEE1599 di cui si vuole ottenere la rappresentazione in piano roll, l'altro contiene l'intero codice di query Xpath discusso finora, con l'aggiunta del metodo *picturebox2.Invalidate()* per ridisegnare la griglia dopo aver raccolto tutti i dati.

A questo punto l'applicazione è completa. Nell'immagine successiva viene mostrata l'applicazione che fornisce la rappresentazione in piano roll di un documento IEEE 1599.

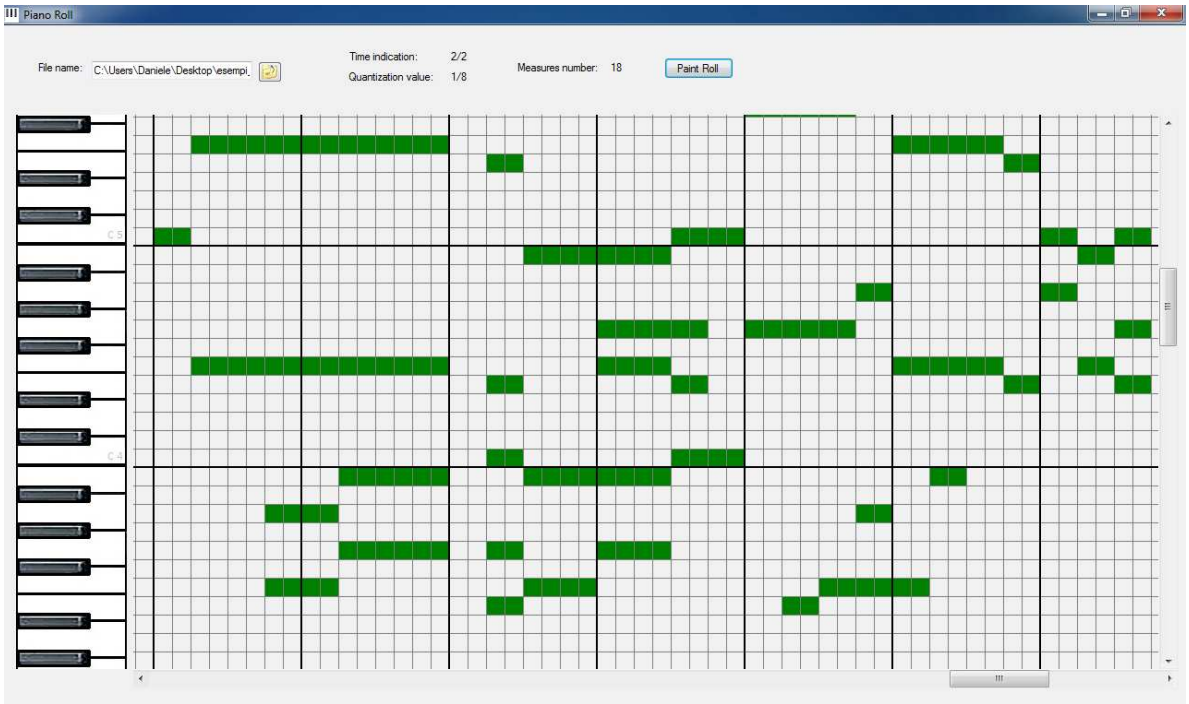


Fig.34 – Interfaccia grafica definitiva dell'applicazione "Piano Roll"

Conclusioni

Le applicazioni che supportano il formato IEEE 1599 finora sviluppate forniscono una rappresentazione grafica che considera gli eventi solo nella loro singolarità. Lo sviluppo dell'applicazione "Piano Roll" ha permesso la visualizzazione delle informazioni musicali contenute all'interno dei documenti IEEE1599 in una forma grafica più complessa e ben strutturata, che si avvicina molto allo spartito musicale.

Le difficoltà incontrate durante la fase di progettazione del presente lavoro hanno riguardato principalmente le modalità di traduzione dei valori delle note nei corrispondenti pixel; in questi casi è stato infatti necessario effettuare diversi passaggi prima di ottenere i valori desiderati.

Nel complesso, il formato IEEE 1599 si è dimostrato un linguaggio molto funzionale alla rappresentazione della musica, fornendo un valido elemento strutturale che può essere utilizzato sia in modo professionale che a livello intuitivo. La possibilità di contenere informazioni di ogni tipo, come la notazione, l'organizzazione grafica, l'audio e il video, danno la possibilità allo sviluppatore di creare nuove applicazioni di ogni genere.

Per quanto riguarda l'applicazione Piano Roll, i possibili sviluppi futuri potrebbero consistere nel fornire all'applicazione delle funzionalità aggiuntive, come ad esempio quelle messe a disposizione dai sequencer midi. Tra queste possiamo trovare:

- la possibilità da parte dell'utente di interagire con l'interfaccia grafica per compiere diverse azioni (modifica dei valori di altezza e durata delle note, inserimenti o cancellazioni di note, e così via).
- Collegare al piano roll un sequencer audio in grado di riprodurre musicalmente il brano seguendo le barre tracciate sulla griglia.

Pur avendo rispettato gli obbiettivi iniziali, l'elaborato può essere ulteriormente sviluppato e ampliato, aggiungendo ulteriori funzionalità fruibili dall'utente finale.

L'esperienza ricavata dallo sviluppo di questa applicazione è stata molto interessante, sia per quanto riguarda l'utilizzo del linguaggio di programmazione ad oggetti C# (e tutte le altre tecnologie coinvolte), sia per le metodiche di lavoro adottate nello sviluppo del programma. Trattandosi di una modalità di comunicazione musicale e multimediale di nuova generazione, ci sarà spazio per molteplici sviluppi futuri che potranno essere approfonditi, possibilmente, anche in ambito professionale.

Bibliografia

[1] Bradley L. Jones, “SAMS Teach Yourself - the C# Language in 21Days”, ed. SAMS (2004).

[2] James Foxall, Wendy Haro-Chun, “Teach Yourself C# in 24 Hours”, ed. Sams Publishing (2002).

[3] <http://www.pianola.org/index.cfm>, 2005, agg. 2008.

[4] <http://www.pianola.co.nz/pianorolls/about.html>, 2009, agg. 2010.

[5] <http://www.cubase.it/modules.php?name=News&file=article&sid=259>, 1999, agg. 2010.

[6] S.Robinson, C.Nagel, J.Glynn, M.Skinner, K.Watson, B.Evjen, "Professional C#", 3dt edition, "Manipulating XML", pp. 781-836, Wiley Publishing (2004).

[7] Luigi Rossi, “Teoria Musicale”, ed. Carrara (1977).

[8] Luca 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, IEEE CS, Lugano, Switzerland (2008).

[9] Luca A. Ludovico, “IEEE 1599: a Multi-layer Approach to Music Description”*, Journal of Multimedia (JMM)*, vol. 4/1, pp. 9-14, Academy Publisher (2009).