**Università degli Studi di Padova**

FACOLTÀ DI INGEGNERIA
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

# CaRo 2.0: sviluppo e valutazione di un software per il rendering interattivo ed espressivo di partiture musicali

Laureando:
**Massimiliano Barichello**
**Matricola 604153-IF**

Relatore:
**Prof. Antonio Rodà**

Correlatori:
**Maestro Davide Tiso**
**Prof. Sergio Canazza**

**Anno Accademico 2010-2011**

*Alla mia famiglia*

# Ringraziamenti

Questo lavoro di tesi risulta essere la conclusione di una carriera universitaria, iniziata senza la piena consapevolezza di ciò che mi avrebbe portato. Il contesto nel quale ho trascorso gli ultimi anni mi ha permesso di conoscere aspetti della vita che non avrei mai potuto vedere e apprezzare altrimenti. Per questo motivo sento il bisogno di dover ringraziare quanti hanno contribuito.

Un forte ringraziamento va ai miei genitori, che mi hanno sostenuto nelle scelte importanti della mia vita, e hanno saputo consigliarmi ed aiutarmi nei momenti difficili permettendomi di vivere serenamente in tutti questi anni. Un grande ringraziamento va anche a mia sorella Arianna, che si sta anche lei avventurando nella vita post-laurea, e potrà coltivare le sue ambizioni con la sua tenacia.

Un ringraziamento va anche agli zii e ai cugini che continuano ancora ad ascoltare le mie "tragiche avventure". Un particolare grazie anche alla nonna Celestina, sempre pronta ad accogliermi in qualsiasi momento.

Altrettanto importanti sono stati gli amici di Università: Federico, Davide, Luca, Lucio e Michele. Assieme a loro ho potuto superare indenne (o quasi) gli ostacoli e gli imprevisti che la vita universitaria ha saputo offrire, ed hanno saputo rendere divertente ogni giornata passata a Padova: mi vengono in mente certe risate, e certi discorsi, che solo voi che eravate presenti potete capire. Senza di loro questi anni sarebbero stati molto più difficili e pesanti.

A loro si aggiungono tutte le altre persone che hanno allietato i pomeriggi di studio e i pranzi in mensa Piovego, e che hanno quindi contribuito a distogliere l'attenzione dal cibo proposto (e dallo studio): Marta, Eliana, Maristella, ecc.

Non per ultimi ringrazio sicuramente gli amici, di compagnia, di lavoro, di bevute: Nicola, Enrico, Andrea, Diego, Cristina, Federica ...ce ne sono tanti altri, ma l'elenco annoierebbe sicuramente il lettore. Grazie a loro sono riuscito a *staccare* dalla vita impegnativa università+lavoro, e ho potuto ricaricare le batterie ogni volta che era necessario. Fortuna vostra è che ora non vi romperò più le scatole con la frase "*Go da studiar!*".

Un grazie anche ai prof Canazza e Rodà, per la stesura della tesi e la vittoria in Rencon.

Nel ringraziare anche coloro che non ho nominato, concludo con una citazione di Steve Jobs, recentemente scomparso: "*Your time is limited, so don't waste it living someone else's life*".

# Sommario

Questo lavoro di tesi ha per oggetto lo sviluppo dell'applicazione CaRo 2.0, un sistema computazionale automatico che permette, in modo interattivo, di realizzare un rendering espressivo di partiture musicali. Proprio per la funzione svolta da questo software, viene inizialmente illustrato lo stato dell'arte dei sistemi automatici per l'esecuzione espressiva di partiture musicali, e le difficoltà riguardanti la loro progettazione. Di particolare interesse è la progettazione di interfacce per il controllo dei numerosi parametri richiesti per la realizzazione di una performance espressiva. Tra le diverse tipologie di interfacce che si possono creare trovano spazio quelle interattive, cioè quelle che permettono ad un utente di controllare in real-time la performance musicale (andando a cambiare per esempio il tempo, le dinamiche, le legature ecc.), simulando una attività analoga alla conduzione di un'orchestra. Il sistema CaRo 2.0 è dotato di un'interfaccia interattiva che fa il mapping tra i movimenti del mouse nello spazio di controllo bidimensionale *Kinematic-Energy* e i parametri legati all'espressività musicale.
All'interno della tesi è presente una descrizione approfondita dell'intera parte implementativa di CaRo 2.0, con una descrizione dei miglioramenti rispetto la versione precedente (1.4). Tale capitolo rappresenta anche una guida nei confronti di nuovi utenti che vogliano apprendere il funzionamento dell'applicazione.
Il software ha dimostrato prestazioni allo stato dell'arte nel campo dei sistemi computazionali per l'esecuzione della musica, ottenendo il punteggio più alto nel Rencon Workshop 2011, il principale contest internazionale per sistemi di questo tipo.
Nel capitolo finale viene affrontato il tema della valutazione delle performance musicali automatiche, analizzando i fattori che possono influenzare il giudizio di una performance musicale realizzata da un calcolatore. In particolare viene posta e analizzata la questione della valutazione sotto l'aspetto qualitativo del brano prodotto, rispetto a un brano eseguito da un musicista umano, secondo quello che viene comunemente chiamato Test di Turing musicale.

# Indice

# Elenco delle figure

# Capitolo 1

# Sistemi automatici e interfacce interattive per il controllo di performance musicali espressive

## 1.1 Sistemi di performance automatici

### 1.1.1 Stato dell'arte

Quando un musicista suona un pezzo di musica, non lo fa in maniera meccanica, cioè con tempo e intensità sonora costanti come sono descritti nello spartito musicale stampato, ma velocizza alcune parti, ne rallenta delle altre, e scandisce certe note. L'espressività musicale può essere definita come la deviazione da uno standard musicale, quando uno spartito viene eseguito da un musicista. Si parla di deviazione in termini di attributi delle note, quali pitch, timbro (il timbro distingue strumenti musicali diversi), tempo (detto anche timing) e dinamiche (dette anche variazioni di intensità sonora). Il modo in cui questi parametri variano durante la performance non è precisato in maniera specifica nello spartito stampato, quindi il performer deve saper usarli in modo appropriato. Questo infatti è quello che distingue un buon musicista da un altro; ogni musicista interpreta a suo modo ogni pezzo, e nessuna interpretazione sarà mai uguale a quella di altri performer. A questo punto una domanda sorge in maniera spontanea:

può un computer prendere vantaggio da questa conoscenza,
e diventare abile a sostituire un famoso performer?

Come verrà visto nelle sezioni successive, Delgrado et al. [21] hanno riportato numerosi tentativi e modelli computazionali per tentare di raggiungere questo obiettivo.

### 1.1.2 La performance musicale e i suoi parametri

La maggior parte delle persone giudicherebbe un'esecuzione letterale, di uno spartito musicale, significativamente meno interessante rispetto a una performance di quel pezzo eseguita da un musicista moderatamente esperto. Questo perché quello che ascoltiamo non è un'interpretazione letterale o meccanica dello spartito. Naturalmente il veicolo principale per comunicare le intenzioni espressive di un compositore è proprio lo spartito; tuttavia le informazioni scritte nel foglio non possono dare una descrizione completa di queste intenzioni, infatti lo spartito trasporta informazioni riguardo la struttura ritmica e melodica di un pezzo, ma ancora non descrive precisamente le caratteristiche del tempo e del timbro del suono. Stesso discorso si potrebbe fare per la velocità e il tono usato durante la lettura di un testo scritto.
Quindi la musica che ascoltiamo ha due importanti sorgenti: lo spartito e la performance, le quali necessitano l'una dell'altra. Widmer e Goebl [84] definiscono la performance musicale espressiva come

*the deliberate shaping of the music by the performer,*
*in the moment of playing, by means of continuous variations of*
*parameters such as timing, loudness or articulation*

cioè la modellazione intenzionale della musica da parte del performer, nel momento dell'esecuzione, per mezzo di continue variazioni di parametri quali il tempo, l'intensità sonora e l'articulation.

- I cambiamenti di tempo sono una deformazione non lineare della successione regolare di beats [1] che definiscono il tempo nello spartito.

- I cambiamenti nell'intensità sonora (cioè le dinamiche) sono modifiche dell'intensità delle note rispetto alle altre.

- Con articulation si indica la tecnica di aggiungere modifiche alle singole note o gruppi di note tramite dei simboli di articolazione. Essa consiste nel variare la differenza tra note continue rendendo, ad esempio, la prima nota più corta o sovrapponendola alla seconda.

La performance musicale è un'attività umana che richiede attitudini emozionali, cognitive e artistiche, e coinvolge aspetti di tipo fisico, acustico, fisiologico, psicologico, sociale e artistico. Diversi fattori determinano l'interpretazione di un pezzo musicale. Uno dei più ovvi è la condizione fisica del performer; infatti lo stato d'animo, di salute e di fatica

[1] Beat o battiti. Un beat è legato al denominatore della time signature, indicando a quale nota viene associato 1 beat. Beats Per Minute (BPM) è anche detto Quarter note Per Minuto (QPM), cioè numero di note da quarto al minuto.

giocano un ruolo cruciale nel processo d'uso di uno strumento. Alcuni studi infatti [35] [68], hanno mostrato che si avevano maggiori diversità di interpretazione di un pezzo eseguito dalla stesso performer, quando si trovava in condizioni fisiche diverse. Altri fattori che affliggono l'interpretazione sono il luogo in cui si trova il performer, perché è il luogo a stabilire il suono che può essere prodotto, e il tipo di strumento che viene utilizzato.
Ci sono diversi motivi che possono portare ad un esecuzione che contenga deviazioni rispetto ai valori nominali riportati in uno spartito.

- In primo luogo, l'espressività è usata come strumento per comunicare emozioni. Rigg [68] ha trovato alcune relazioni tra emozioni e struttura musicale; infatti alcune delle regolarità scoperte sono ad esempio che la musica solenne tende ad essere lenta, con pitch bassi, e senza irregolarità, mentre la musica allegra è veloce e con pitch più alti. Canazza et al. [12] hanno studiato come i parametri fisici di brani musicali (come il tono, l'articulation o il tempo globale), sono affetti dalle modifiche delle intenzioni espressive del performer. Nei loro esperimenti, è stato chiesto al performer di esprimere, rispetto la sua interpretazione personale dello spartito musicale, concetti sensoriali come `bright`, `light` o `dark` (brillante, leggero o scuro). Poi un'analisi sonologica delle registrazioni ha reso possibile relazionare certi valori a questi concetti sensoriali. Ad esempio un'interpretazione light è stata trovata essere più veloce in tempo, con una durata delle note più breve e attacchi delle note più soft. Sono state ottenute delle performance automatiche settando certi valori più alti o più bassi e, in test di ascolto formali, gli ascoltatori sono stati in grado di riconoscere e identificare le intenzioni musicali.

- In secondo luogo, l'espressività chiarisce la struttura musicale, cioè la struttura metrica (ritmo e andamento generale del brano), il phrasing (indica un gruppo di note consecutive alle quali viene applicata una modifica espressiva, ad esempio un'accelerazione nel tempo) e la struttura armonica (toni e accordi). Anche la struttura musicale ha la sua influenza nell'espressività della performance; infatti è stato scoperto che l'inizio e la fine di una frase (insieme consecutivo di note) tende ad essere più lento rispetto al resto delle note.

### 1.1.3 La ricerca di deviazioni espressive nel brano musicale

I progressi nelle capacità computazionali e nella sintesi digitale dei suoni (cioè la generazione digitale del suono: c'è la sintesi diretta, basata sulla modifica diretta di parametri, e la sintesi per analisi di suoni preesistenti) ha permesso il controllo in real-time dei suoni

sintetizzati. Il controllo espressivo è diventato così un'area di grande interesse nel campo della Sound and Music Computing [2].
Storicamente la ricerca nelle performance musicali si è focalizzata nel trovare dei principi generali, che evidenzino i tipi di deviazioni espressive rispetto allo spartito come simbolo di interpretazione espressiva. Possono essere distinte tre diverse strategie di ricerca:

1. metodo di *analisis-by-synthesis* (analisi per sintesi), cioè usando interviste con musicisti esperti per cercare di tradurre la loro esperienza in regole espressive;

2. metodo di *analisis-by-mesaurement* (analisi per misurazione), cioè un'analisi acustica e statistica delle performance da parte di un musicista reale;

3. tecniche di *inductive machine learning* (macchine di apprendimento induttivo), applicate a grandi database di performance. Per apprendimento induttivo si intende che il sistema parte dai fatti e dalle osservazioni, ottenendo così conoscenza che sia valida anche per casi non ancora osservati.

Studi hanno mostrato che ci sono regolarità significative che possono essere scoperte utilizzando queste strategie, e che modelli computazionali per le performance espressive hanno dimostrato di essere capaci di riprodurre veri risultati musicali. Questi risultati ispirano tutt'ora a nuove ricerche su modelli computazionali più completi, e scenari applicativi ambiziosi. Uno dei problemi in questo settore consiste nella rappresentazione del modo in cui certi artisti suonano (cioè lo stile individuale del musicista), analizzando solo alcune delle loro interpretazioni. Queste informazioni dovrebbero consentirci di identificare un artista ascoltando soltanto la loro interpretazione. Però tali studi sono difficili in quanto lo stesso musicista può eseguire lo stesso pezzo in molti modi diversi. Recentemente sono nati numerosi nuovi metodi per il riconoscimento delle performance e dello stile. Finora, le ricerche si sono principalmente concentrate nella descrizione dettagliata delle variazioni nelle performance in relazione alla struttura musicale. Tuttavia c'è stato un recente spostamento verso descrittori musicali di alto livello, per la caratterizzazione e il controllo delle performance musicali, specialmente rispetto alle caratteristiche emozionali.
Il controllo interattivo di espressività musicale è tradizionalmente un compito del direttore d'orchestra. Diversi tentativi sono stati fatti per controllare il tempo e le dinamiche di uno spartito eseguito dal computer con qualche tipo di dispositivo di input gestuale (es. mouse). Per esempio Friberg [28] ha descritto un metodo per il controllo interattivo, e in real-time, di un sistema di regole che contengono modelli per il phrasing, piccoli livelli di timing, articulation e intonazione. Con questo sistema, possono essere raggiunti alti livelli di controllo espressivo. Recentemente sono stati effettuati degli sforzi per cercare di visualizzare gli aspetti espressivi di performance musicali (esempio nello spazio tempo-loudness [37]).

[2] http://smcnetwork.org

### 1.1.4 Problemi relativi all'esecuzione automatica delle performance

L'uso di modelli di performance musicale automatici hanno fatto nascere numerosi problemi, che finora sono stati solo parzialmente affrontati. Le intenzioni espressive di un artista sono imprevedibili, e l'aspettazione e le reazioni degli ascoltatori sono ancora trascurati nei modelli correnti, nonostante la sorpresa e l'imprevedibilità sono aspetti cruciali, soprattutto nelle performance live. I modelli dovrebbero tener conto di alcune variabili quali il contesto della performance, l'intenzione dell'artista, l'esperienza personale, e l'aspettativa degli ascoltatori.
Nel passato questo problema è stato affrontato usando modelli che utilizzano tecniche di *machine learning* e di induzione logica, ma molti di questi sforzi hanno solo cercato di catturare principi espressivi comuni tra performance e musicisti. Tuttavia, l'ultimo obiettivo di questo tipo di ricerche non è la replicazione automatica dello stile o la creazione di un musicista artificiale, ma l'uso del computer per insegnare di più a noi umani riguardo l'attività artistica, spesso sfuggente, delle performance musicali espressive. Avere un computer che sia in grado di estrarre informazioni per catturare aspetti dello stile individuale del performer, è solo un primo passo.

### 1.1.5 Caratteristiche e modelli di sistemi di performance automatici

La caratteristica principale di un sistema automatico è quella di convertire uno spartito musicale in una performance musicale espressiva, includendo tipicamente deviazioni di tempo, suono e timbro rispetto una realizzazione meccanica dello spartito. Come già accennato, di solito sono due le strategie che vengono maggiormente utilizzate per la costruzione di un sistema automatico di questo tipo: analisis-by-synthesis (analisi per sintesi), e analisis-by-mesaurement (analisi per misurazione).
Il primo metodo implica che l'intuito, il sapere non scritto, e l'esperienza di un musicista esperto vengano tradotti in delle regole per la performance. Una limitazione di questo metodo però, consiste nel fatto che queste regole riflettono principalmente le idee musicali di un specifico musicista. Però è anche da sottolineare che un musicista professionista dovrebbe possedere anche una certa generalità, e in alcuni casi le regole prodotte con questo metodo hanno un carattere generale.
Il secondo metodo invece genera delle regole che derivano da misure di performance reali, generalmente salvate in CD. I dati vengono spesso processati statisticamente, quindi le regole riflettono le deviazioni tipiche, e non le deviazioni individuali di una certa performance; nonostante le deviazioni individuali siano comunque musicalmente molto rilevanti.

Molti autori hanno proposto modelli per la performance automatica della musica:

- Todd [76] ha presentato un modello basato sul metodo dell'analisi per misurazione.

- Sistemi basati sulle regole sono stati proposti da Zanon e De Poli [87], Friberg [26], e Friberg et al. [30].

- Sono stati sviluppati anche sistemi basati su tecniche di intelligenza artificiale. Widmer [83] ha proposto un sistema basato su una macchina di apprendimento che estrae regole dalle performance. Ishikawa et al. [44] hanno sviluppato un sistema per le performance di musica classica, dove vengono estratte le regole da performance preregistrate usando un algoritmo di analisi a regressione multipla. Arcos et al. [3] hanno sviluppato un sistema per la sintesi di performance musicali per gli strumenti più semplici. Delgado et al. [21] hanno sviluppato un approccio multiagent per la composizione e generazione della musica.

### 1.1.6 Difficoltà nello sviluppo futuro dei sistemi automatici

Dal momento che la sintesi letterale delle note da uno spartito è blanda e poco attraente, c'è un'opportunità per i sistemi di apprendimento che possono automaticamente produrre variazioni espressive anche molto interessanti. Una performance musicale è una tra le tante attività che le persone possono fare senza conoscere esattamente come possono farla. Questo è uno dei problemi che si possono affacciare, perché non c'è nessun modello che ci dice accuratamente come eseguire una performance. Quando invece ci riferiamo al dominio artistico-musicale, è molto difficile trovare un modello corretto le cui predizioni corrispondano sempre a quello che fa l'umano.

Molti aspetti sono coinvolti in una performance espressiva, ed è impossibile usarli tutti. Inoltre ci sono alcuni parametri che sono comunemente considerati poco rilevanti, quando in realtà lo sono. Le tecniche odierne affrontano solo una piccola porzione dell'intero problema. Una sfida futura consiste nel direzionare il problema sulle variabili non ancora utilizzate, e in quelle ancora da scoprire.
Ottenere dei dati molto precisi per ogni parametro è un problema molto impegnativo, che non può essere fatto in modo automatico. Inoltre annotare tutte queste informazioni è un processo molto costoso in termini di tempo. Uno dei primi studi ha affrontato questo problema riducendo la lunghezza del brano musicale ad alcuni secondi. Approcci più recenti cercano di evitare questa riduzione, usando delle tecniche statistiche di apprendimento e focalizzandosi in rappresentazione più astratte delle note e del loro valore. Nonostante alcuni successi nella progettazione di modelli di performance automatici, i modelli correnti sono estremamente limitati e molto semplici nel complesso fenomeno dell'espressione musicale. Naturalmente è impossibile arrivare a dei modelli che predicano

in maniera completa la complessità del comportamento umano. Tuttavia, lavorare verso questo obiettivo può aumentare la conoscenza e l'apprezzamento della complessità dei comportamenti artistici. Capire le performance musicali richiede una combinazione di approcci e discipline.
Per le neuroscienze cognitive, scoprire i meccanismi che governano la comprensione della performance musicale, è un problema di primaria importanza. Diverse aree del cervello sono coinvolte nel riconoscimento delle diverse caratteristiche delle performance. Conoscere questo può essere un aiuto importante alla modellazione formale ai processi che, ad esempio, distinguono le performance fatte da un umano e le performance fatte da una macchina.

Un altro problema è l'individuazione di ogni lavoro. Sebbene ci sia una grande quantità di dati disponibili, ogni canzone è diversa dal resto. Quindi non è adeguato applicare il modo di eseguire un brano, al modo di eseguire un altro brano. È necessario uno studio approfondito per comprendere l'autore, il contesto e la musica, in quanto è bene ricordare che la performance artistica è tutt'altro che prevedibile.

Come visto, le ricerche nell'ambito delle performance musicali spaziano da studi, con lo scopo di capire le performance musicali espressive, al tentativo di modellare gli aspetti delle performance in modo formale, quantitativo e predittivo. Questa ricerca può fornire strumenti espressivi che tradizionalmente sono nascosti nelle abilità dei musicisti e nella loro intuizione. Infatti quando saranno formulati esplicitamente, questi strumenti daranno all'utente la possibilità di eseguire file musicali con diverse sfumature espressive. Nonostante siamo scettici nella rimpiazzo completo di una performance umana con una performance ottenuta attraverso una macchina, siamo sicuri che questa tecnologia sarà disponibile in un futuro non troppo distante. Infatti, realtà già esistenti come la competizione di Rencon vanno in questa direzione. Siamo fermamente convinti che è arrivato il tempo per l'informatica di lavorare nel dominio della musica, in quanto queste ricerche avranno un ottimo impatto nell'arte e nella scienza.

## 1.2 Interfacce per il controllo real-time delle performance musicali

La musica è stata storicamente un punto d'incontro tra la tecnologie e l'espressione artistica. La progettazione degli strumenti musicali potrebbe essere stata la prima area della tecnologia dove il design attento e sistematico delle interfacce gioca un ruolo essenziale e centrale. Mentre la musica è sempre stata una forza trainante per l'innovazione della tecnologia, è anche vero che le nuove tecnologie hanno aperto la strada a nuove forme di espressione musicale e di sperimentazione. Per dare un esempio familiare, il pianoforte moderno, e di conseguenza il repertorio classico del pianoforte (come i concerti di Beethoven), non sarebbe possibile senza gli ottimi miglioramenti nella metallurgia nel 18-mo secolo.

Con la rapida evoluzione dei media digitali, dei materiali e di altre tecnologie, si stanno aprendo delle opportunità senza precedenti per gli inventori e i designer di interfacce musicali. Le possibilità offerte da queste nuove tecnologie hanno dato una spinta a nuove forme musicali. Per esempio nell'ultimo secolo (specialmente negli anni 50 e 60) si è visto l'ascesa della sintesi del suono elettronico, che ha dato vita a nuove forme musicali, sia popolari che classiche. Ci si può aspettare che il continuo progresso nelle tecnologie informatiche, stimolerà i compositori e i musicisti a sperimentare con nuovi mezzi di composizione e nuovi strumenti di performance.

Lo sviluppo di nuovi sensori di interfaccia, di riconoscimento visivo, di realtà virtuale aumentata, e di strumenti per il feedback tattile, stanno aprendo nuove strade per nuove "avventure" musicali interattive. Il campo dei controller alternativi della musica sono ad uno stadio simile a come lo era la sintesi elettronica negli anni 50. I modelli di riferimento (paradigmi) di base sono ancora in fase di studio, e c'è un'esplosione di nuove interfacce con poco pensiero sistematico su dove il campo è diretto. Questo perché i controller alternativi sono essenzialmente dei mezzi di mappatura tra il comportamento umano e l'espressione musicale, e sono argomenti trattati dai progettisti di interfacce che potrebbero essere molto utili per comprendere e chiarire lo stato dell'arte di questo campo.

### 1.2.1 Stato dell'arte

La direzione di un'orchestra è uno dei mezzi più semplici ed efficaci di espressione. *Che cosa si intende per "controllo di livello superiore"?* I metodi per controllare una performance musicale possono dividersi in tre differenti categorie:

1. Tempo/dynamics. Un semplice caso è il controllo dei valori istantanei di tempo e di dinamica di una performance.

2. Performance models. Usando modelli per le performance per le strutture musicali, come ad esempio il *KTH rule system* (vedi sezione 1.2.4), è possibile controllare i dettagli della performance come le modalità di esecuzione del phrasing, articulation, accenti e altri aspetti della performance musicale.

3. Semantic descriptions. Queste descrizioni semantiche possono essere un'espressione emotiva come aggressivo, sognante, malinconico, o tipiche istruzioni per la performance (riferite spesso al movimento) come andante o allegretto.

Nel passato sono stati sviluppati e costruiti diversi sistemi di direzione di orchestra che usano la categoria 1, cioè che controllano il tempo e le dinamiche.
Uno dei primi sistemi è il sistema **Radio Baton** (progettato da Mathews [54]), il quale certe volte è ancora usato per condurre uno spartito, o utilizzato come un controllore generale. Questo tipo di controller consiste di due bacchette (due radio trasmittenti) e di una piastra rettangolare (l'antenna ricevente). Vengono misurate le posizioni in 3D di ogni bacchetta sopra il piatto. Tipicamente una bacchetta viene usata per battere il tempo, mentre l'altra per controllare le dinamiche. Usando un software "*Conductor*" (conduttore), viene suonato uno spartito simbolico (un file MIDI convertito) attraverso un sintetizzatore MIDI. Il sistema è molto preciso nel senso che la posizione di ogni beat è esattamente data dal gesto verso il basso dato dalla bacchetta. Questo permette un controllo molto accurato del tempo, ma richiede anche pratica, anche per un direttore d'orchestra esperto.

**Conducting Simulator**, sviluppato da Usa et al. [78], è un sistema che riconosce i movimenti di conduzione, utilizzando un modello di Markov nascosto (*HMM - Hidden Markov Model* [3]) per i dati ottenuti con sensori di accelerazione.

Un sistema di controllo più recente controlla sia l'audio che il video, ed si chiama **Personal Orchestra**, sviluppato da Borchers et al. [6]; il quale è stato ulteriormente sviluppato in **You're the Conductor** [49]. Questi sistemi sono condotti usando una bacchetta wireless con luce ad infrarossi per stimare la posizione della bacchetta in due dimensioni. La *Personal Orchestra* è un'installazione nell'*House of Music* a Vienna (Austria), dove gli utenti possono dirigere registrazioni reali della *Vienna Philharmonic Orchestra*. Il tempo dell'audio, del video e delle dinamiche del segnale audio possono essere control-

[3] Un Modello di Markov nascosto (*Hidden Markov Model - HMM*) è una catena di Markov i cui stati non sono osservabili direttamente. Più precisamente: la catena ha un certo numero di stati; gli stati evolvono secondo una catena di Markov; ogni stato genera un evento con una certa distribuzione di probabilità che dipende solo dallo stato; l'evento è osservabile ma lo stato no. I modelli nascosti di Markov sono conosciuti particolarmente per le loro applicazioni nel riconoscimento dello schema temporale dei discorsi parlati, della scrittura a mano, nel riconoscimento di textures e la bioinformatica.

late producendo un'esperienza molto realistica. A causa delle restrizione nel modello di manipolazione del tempo, il tempo è controllato solo in passi discreti.
L'installazione di *You're the conductor* è anch'essa esibita in un museo, ma è rivolta ai bambini invece che agli adulti. Quindi è stata progettata attentamente per essere intuitiva e di facile uso. È stato sviluppato un nuovo algoritmo di estensione del tempo, che permette qualsiasi cambio temporale delle registrazioni originali. Dall'esperienza con i bambini, è stato trovato che l'interfaccia più efficiente è una semplice mappatura della velocità dei gesti per il tempo, e della grandezza del gesto per il volume.

Sono stati costruiti anche altri sistemi di direzione. Per esempio il **Conductor's jacket** (*jacket* = giacca) di Marrin Nakra [53], il quale rileva diversi parametri del corpo umano, come la tensione dei muscoli e la respirazione, per poi tradurli in espressioni musicali.

La **Virtual Orchestra** è una simulazione grafica 3D di un'orchestra sviluppata da Ilmonen [43], controllata da una bacchetta che fa da interfaccia.

Recentemente, sono stati sviluppati sistemi di conduzione per l'intrattenimento. **Wii Music** [4] è un gioco che simula la conduzione, nel quale si usa Wii Remote come controller (vedi Figura 1.1).

Figura 1.1: Il controller Wii Remote

### I temi affrontati dalla conferenza NIME

La conferenza *NIME* (*New Interfaces for Musical Expression*) è una conferenza internazionale annuale, che raccoglie 200-500 partecipanti da tutto il mondo, per condividere il loro sapere e il loro lavoro sul design di nuove interfacce musicali. La conferenza ha una grande varietà di partecipanti: ricercatori (musicologia, informatica, interaction design, ecc.),

[4] www.wiimusic.com/

artisti (musicisti, compositori, ballerini, ecc.), e sviluppatori (lavoratori autonomi e industriali). Il comune denominatore è l'interesse nella tecnologia innovativa e nella musica, e i contributi alla conferenza coprono qualsiasi cosa, dalla ricerca di base sulla conoscenza umana, a strumenti tecnologici sperimentali, fino alle performance multimediali.
Nella conferenza vengono affrontati e discussi numerosi problemi riguardanti la progettazione e la valutazione di controller musicali alternativi, con lo scopo di stimolare la ricerca di base in merito all'impatto della tecnologia delle interfacce sulla cultura musicale.

- Il ritmo rapido di cambiamento delle nuove tecnologie utilizzate per costruire nuovi tipi di controlli, è a doppio taglio: c'è una crescita entusiasmante di nuovi controller, però questi controller rischiano di diventare tecnologicamente obsoleti molto velocemente. *La definizione di norme favorirà o ostacolerà l'evoluzione delle interfacce musicali?* Lo standard MIDI, ad esempio, è un caso emblematico perché è diventato uno standard diffuso nella musica elettronica, ma esiste comunque della polemica sul fatto che la sua influenza sia nel complesso positiva o negativa.

- I controlli alternativi portano nuova libertà di espressione, in quanto la mappatura tra azione e la generazione del suono può essere arbitrariamente cambiata. Tuttavia, poche mappature sono relativamente intuitive e naturali, e molte non fanno uso della nostra intuizione fisica e sono quindi difficili da imparare a usare. Risulta quindi importante discutere su quali caratteristiche di mappatura siano appropriate per la progettazione di interfacce musicali.

Inoltre viene dedicato ampio spazio alla questione pratica di come progettare una buona interfaccia musicale.

- Per individuare i criteri per la valutazione delle interfacce musicali, si discuterà di usabilità e comprensibilità, espressività, sensibilità e raffinatezza, estetica, sul fatto di sentire bene o male, e altri criteri. L'obiettivo è quello di chiarire quali sono le linee guida necessarie per lo sviluppo di interfacce.

- Identificare i principali sviluppi della tecnologia per le interfacce che offrono le opportunità più interessanti per l'espressione musicale. Per esempio: sensori tattili, sensori di posizione, orientamento e movimento, sensori di pressione e di tensione, guanti e tute, feedback tattile, computer vision, e riconoscimento di pattern. *Come viene mappato l'input del sensore in un suono musicale? Ci sono certi modi di comportamento umano, e di movimento, più adatti rispetto ad altri per l'espressione musicale?*

- Per discutere del ruolo delle scienze cognitive e psicologiche nella progettazione di interfacce musicali. *Quali sono i fattori che determinano se un'interfaccia è adatta all'espressione creativa di idee complesse e di pattern emozionali?*

- Per condividere esperienze collettive. Ogni partecipante dovrebbe aggiungere la sua esperienza nel progettare controller alternativi per la musica, con i relativi vantaggi e svantaggi nelle performance musicali. Tramite questi resoconti, si cercherà poi di suggerire linee guida per la progettazione.

### 1.2.2 Simulare la direzione di un'orchestra

Un problema comune dei sistemi di direzione visti sopra, è la carenza del punto di vista dell'orchestra: un conduttore/direttore reale guarda i membri di un'orchestra. L'orchestra è un gruppo di musicisti che hanno dei modelli di performance in base alla loro propria musicalità. L'orchestra non sempre ubbidisce completamente alle direzioni del direttore. Le interazioni tra le direzioni del direttore e il modello di performance dell'orchestra portano a una performance musicale reale. I sistemi esistenti non possono fornire un player con una sensazione di dirigere un'orchestra, proprio a causa di questa carenza.
**VirtualPhilharmony (V.P.)** è un sistema di conduzione per trasmettere la sensazione di dirigere un'orchestra, simulando la direzione e l'interazione con l'orchestra attraverso delle euristiche, che riguardando il rilevamento dei battiti forniti dal direttore. Ci sono due problemi nel simulare l'interazione:

- come progettare uno scheduler predittivo;

- come creare un modello di performance dinamica che contenga espressività.

Per risolvere questi problemi, verranno analizzate delle registrazioni reali e verrà utilizzato l'aiuto di direttori esperti per regolare i parametri ed effettuare valutazioni del V.P. La parte del V.P. che simula l'interazione, viene chiamata *Concertmaster Function* (funzione violino).

#### Panoramica di VirtualPhilharmony (V.P.)

La Figura 1.2 mostra una panoramica del VirtualPhilharmony.
Esso estrae i punti di minimo locale e i punti di massimo locale usando un sensore gestuale, il quale raccoglie i movimenti di conduzione del player, e rileva i punti di beat, i volumi e le pause (o fermata) da questi dati. La funzione Concertmaster predice il tempo del prossimo beat, rivede dinamicamente il modello della performance espressiva ottenuto dalle registrazioni reali, e schedula il modello usando il tempo del player e il modello del tempo. V.P. controlla le velocità di ogni nota schedulata in base ai volumi che vengono rilevati dal sensore. Infine, V.P. rilascia le performance dagli altoparlanti e mostra lo spartito.

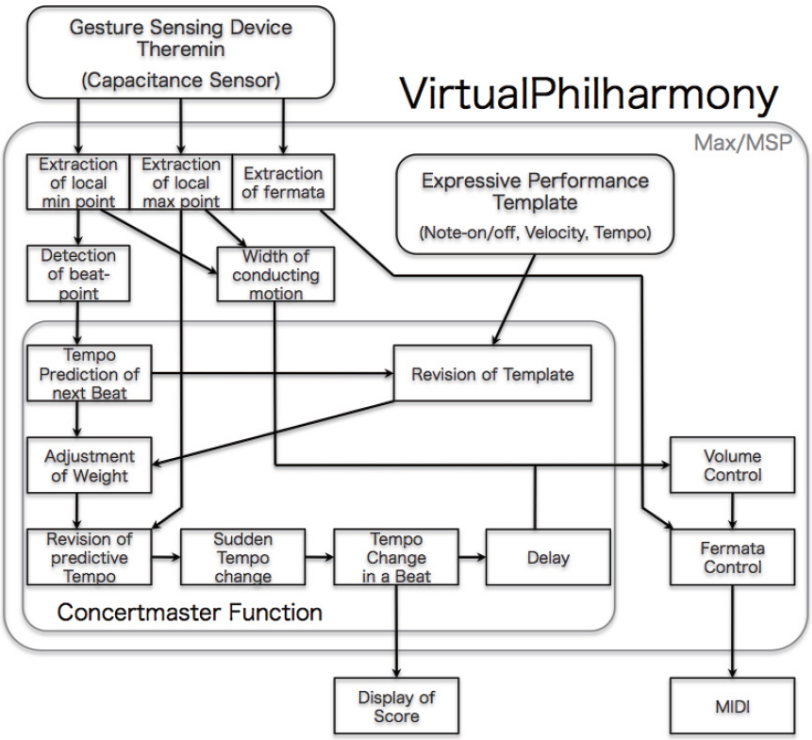



Figura 1.2: Panoramica del sistema di conduzione VirtualPhilharmony

Viene usato un *Etherwave Theremin* di *Moog* [5] come sensore di misura della distanza (vedi Figura 1.3). Il tempo di risoluzione del Theremin è 5ms.

Figura 1.3: Un Etherwave Theremin

[5]Nel 1954 un giovane Bob Moog (assieme al padre) iniziò a costruire i Theremin, uno dei più vecchi strumenti elettronici, e l'unico conosciuto che si può suonare senza toccarlo. Il Theremin è un singolo strumento di oscillazione, che usa due aste di metallo che fanno da antenne per controllare il pitch e l'ampiezza. L'antenna sinistra (un cerchio orizzontale) riduce l'ampiezza non appena la mano sinistra si muove vicino ad essa, mentre l'antenna destra (un cerchio verticale) incrementa il pitch non appena la mano destra si avvicina a essa. I segnali elettrici provenienti dal Theremin vengono poi amplificati e mandati agli altoparlanti. `http://www.moogmusic.com/products/Etherwave-Theremins`

La funzione *Concertmaster*, che è basata su euristiche di conduzione di un'orchestra, è formata dalle seguenti sette funzioni:

- predizione del tempo del prossimo beat;
- revisione del modello (template) di performance espressiva;
- aggiustamento dei pesi;
- revisione dei beat predetti usando un punto locale di massimo nel movimento di conduzione del player;
- supporto di cambi di tempo improvvisi;
- supporto di cambi di tempo in un beat;
- aggiustamento del ritardo tra i movimenti di conduzione e la performance musicale.

Come per il primo punto, vengono calcolati anche i parametri predittivi più adatti per ogni pezzo e ogni beat, analizzando registrazioni reali. Come per il secondo punto, V.P. rivede il modello con il tempo del direttore (in questo caso detto anche player). Per esempio, i rapporti di durata di tre beat in una misura nel valzer di Viennese ($Duration_W$), nel modello sono rivisti in accordo con il tempo del direttore tramite la seguente equazione di regressione:

$$Duration_W = a_W * x_W + b_W$$

Invece il rapporto di durata delle note con il dot (dotted note), nel modello sono rivisti ($Duration_D$) con la seguente equazione di regressione:

$$Duration_D = a_D * x_D + b_D$$

I coefficienti (a e b) di queste equazioni sono ottenute analizzando le registrazioni reali, mentre $x_W$ e $x_W$ sono il tempo del direttore (o player).
I direttori esperti fanno notare che è importante simulare l'interazione tra direttore e orchestra, perché si aggiustano i parametri nel V.P. In aggiunta, V.P. è stato premiato con il *Rendering Award* a *Rencon2010*, e con il *Silver Award* nel *Creative Showcase* in *ACE 2010* [6].

### Rendering delle performance con V.P.

Per effettuare il rendering delle performance con V.P., ci sono i seguenti passi da seguire:

[6] http://www.ace2010.org/

1. preparazione di un modello per le performance espressive a mano;

2. dirigere con V.P.

I modelli espressivi di performance del V.P. sono ottenuti manualmente dalle registrazioni reali, usando un sequencer MIDI. Per SMC-Rencon, la performance espressiva è stata ottenuta con il processo illustrato nello *Step 1* di Figura 1.4. In un primo momento le espressioni sono aggiunte a file del tipo *Standard MIDI File* (SMF) usando uno strumento basato su regole, nel quale vengono "assegnati" il "marchio" espressivo e i rapporti del volume tra mano destra e sinistra. Poi i file SMF vengono elaborati manualmente, e dopo vengono convertiti in un modello V.P. Infine, viene diretto il modello originale usando V.P. come illustrato nello *Step 2* di Figura 1.4. Intonazioni naturali di cambi di volume e di tempi, sono aggiunti al modello mentre si sta dirigendo.

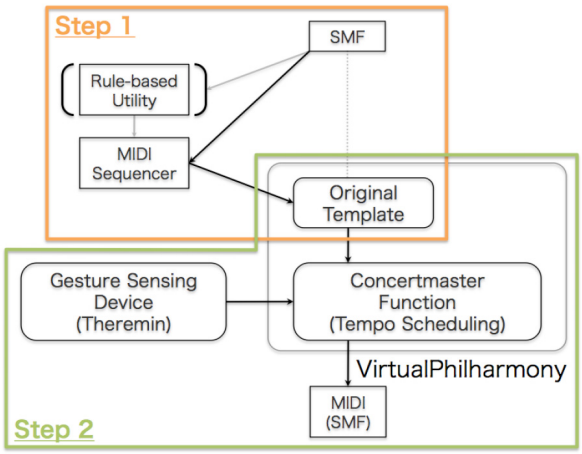



Figura 1.4: Panoramica del rendering di una performance con V.P.

### Esempio: "*A Little Consolation*"

Takashi Baba [5] fu il primo a descrive i risultati dell'analisi del seguente pezzo musicale: "*A Little Consolation*" composto da T. Murao. Il pezzo consiste di cinque parti: A-B-A'-B'-A". Basandosi sull'analisi, è stato creato un modello originale usando il sequencer MIDI: nelle parti A e B il tempo è stato leggermente rallentato, nella parte A' è stato leggermente velocizzato, nella parte B' è stato bruscamente rallentato, e nella parte A" il tempo è troppo lento rispetto al tempo specificato nello spartito. Il pezzo è stato poi diretto usando questo modello.

### 1.2.3 Schema generale per il controllo in Real-Time

Uno schema generale di un sistema basato sul computer per il controllo in real-time delle performance musicali, può essere idealizzato così:

**controller + mappatura**

Il controller è basato sull'analisi di input audio o gestuali (esempio i gesti del musicista). L'analisi fornisce i parametri gestuali (esempio velocità e grandezza del movimento) che possono essere mappati in parametri acustici (esempio tempo e sound level [7] ) responsabili delle deviazioni espressive nella performance musicale.

### 1.2.4 Un modello basato sul sistema di regole KTH

Le performance espressive musicali implicano il controllo di un insieme di parametri acustici. Una volta che questi parametri sono identificati, è importante fare dei modelli che permettano la loro manipolazione in modo musicalmente ed esteticamente significativo [22] [84].

#### *KTH Rule System*

Come descritto da Friberg e Bresin [33], un approccio a questo problema è quello fornito dal sistema di regole KTH per le performance musicali (*KTH performance rule system* [8] ). Questo sistema è il risultato di un continuo progetto di ricerca a lungo termine che riguarda le performance musicali, iniziato da Johan Sundberg [74] [75] [26] [32]. L'idea del sistema di regole è quella di modellare le variazioni introdotte dal musicista quando suona lo spartito. Il sistema contiene correntemente circa 30 regole, che modellano molti aspetti delle performance come tipi diversi di phrasing, accenti, schemi di tempo e intonazione.
Nelle tabelle qui sotto si possono visualizzare molte delle regole nel DM, e per ognuna di esse si possono vedere le variabili delle performance che vengono influenzate:

```
sl = sound level
dr = interonset duration (IOI)
dro = offset to onset duration
```

[7] La pressione sonora è la variazione di pressione rispetto alla condizione di quiete causata da una perturbazione (onda sonora). Il livello di pressione sonora (*SPL - Sound Pressure Level*) o livello sonoro Lp (*sound level*) è una misura logaritmica della pressione sonora efficace di un'onda meccanica (sonora) rispetto ad una sorgente sonora di riferimento. Se il mezzo di propagazione è l'aria (o altro mezzo gassoso), il livello di pressione sonora (*SPL*) è quasi sempre espresso in decibel rispetto alla pressione di riferimento di $20\mu Pa$ (micro-pascal = $10^{-6}$ pascal), di solito considerata la soglia di udibilità per l'uomo (equivale all'incirca alla pressione sonora prodotta da una zanzara che vola a tre metri di distanza)

[8] *KTH* è l'abbreviativo di *Royal Institute of Technology*, cioè un'università di Stoccolma (Svezia) `http://www.kth.se/en`

```
va = vibrato amplitude
dc = cent deviation from equal temperament in cents
```

**MARKING PITCH CONTEXT**

| Rule Name | Performance Variable | Short Description |
|---|---|---|
| High-loud | sl | The higher the pitch, the louder |
| Melodic-charge | sl dr va | Emphasis on notes remote from current chord |
| Harmonic-charge | sl dr | Emphasis on chords remote from current key |
| Chromatic-charge | dr sl | Emphasis on notes closer in pitch; primarily used for atonal music |
| Faster-uphill | dr | Decrease duration for notes in uphill motion |
| Leap-tone-duration | dr | Shorten first note of an up-leap and lengthen first note of a down-leap |
| Leap-articulation-dro | dro | Micropauses in leaps |
| Repetition-articulation-dro | dro | Micropauses in tone repetitions |

**MARKING DURATION AND METER CONTEXT**

| Rule Name | Performance Variable | Short Description |
|---|---|---|
| Duration-contrast | dr sl | The longer the note, the longer and louder; and the shorter the note, the shorter and softer |
| Duration-contrast-art | dro | The shorter the note, the longer the micropause |
| Score-legato-art | dro | Notes marked legato in scores are played with duration overlapping with interonset duration of next note; resulting onset to offset duration is dr+dro |
| Score-staccato-art | dro | Notes marked staccato in scores are played with micropause; resulting onset to offset duration is dr-dro |
| Double-duration | dr | Decrease duration contrast for two notes with duration relation 2:1 |
| Social-duration-care | dr | Increase duration for extremely short notes first note of a down-leap |
| Inegales | dr | Long-short patterns of consecutive eighth notes; also called swing eighth notes |
| Ensemble-swing | dr | Model different timing and swing ratios in an ensemble proportional to tempo |
| Offbeat-sl | sl | Increase sound level at offbeats |

**INTONATION**

| Rule Name | Performance Variable | Short Description |
|---|---|---|
| High-sharp | dc | The higher the pitch, the sharper |
| Mixed-intonation | dc | Ensemble intonation combining both melodic and harmonic intonation |
| Harmonic-intonation | dc | Beat-free intonation of chords relative to root |
| Melodic-intonation | dc | Close to Pythagorean tuning, e.g. with sharp leading tones |

**PHRASING**

| Rule Name | Performance Variable | Short Description |
|---|---|---|
| Punctuation | dro | Automatically locates small tone groups and marks them with lengthening of last note and a following micropause |
| Phrase-articulation | dro dr | Micropauses after phrase and subphrase boundaries, and lengthening of last note in phrases |
| Phrase-arch | dr sl | Each phrase performed with arch-like tempo curve: starting slow, faster in middle, and ritardando towards end; sound level is coupled so that slow tempo corresponds to low sound level |
| Final-ritard | dr | Ritardando at end of piece, modelled from stopping runners |

**SYNCHRONISATION**

| Rule Name | Performance Variable | Short Description |
|---|---|---|
| Melodic-sync | dr | Generates new track consisting of all tone onsets in all tracks; at simultaneous onsets, note with maximum melodic charge is selected; all rules applied on this sync track, and resulting durations are transferred back to original tracks |
| Bar-sync | dr | Synchronise tracks on each bar line |

Quindi, l'input del sistema è uno spartito simbolico, e l'output è il rendering espressivo nel quale vengono modellati i parametri della performance. Ogni regola introduce delle variazioni in una o più variabili di performance:

*IOI* [9], articulation, tempo, sound level, vibration rate, vibrato [10]

Molte regole operano sullo spartito "grezzo" usando i valori delle note come input. Tuttavia alcune regole per il phrasing, come per la carica armonica e melodica (harmonic e melodic charge), necessitano di un'analisi del fraseggio e un'analisi armonica fornita dallo spartito. Questo significa che il sistema di regole non contiene in genere modelli di analisi (questo è un problema complicato e separato di ricerca). Un'eccezione è la regola della punteggiatura, che comprende un'analisi melodica di raggruppamento [29].
Una descrizione musicale di alto livello (ad esempio con le espressioni emozionali), può essere modellata usando una selezione di queste regole e di parametri delle regole [8].
Le regole sono state progettate usando due metodi:

- il metodo *analysis-by-synthesis*;

- il metodo *analysis-by-measurements*.

Nel primo metodo, degli esperti musicali (Lars Frydén nel caso delle regole di performance KTH), dicono agli scienziati come funziona un particolare principio della performance. Gli scienziati lo implementano (esempio implementando una funzione con il linguaggio Lisp), e poi dei musicisti esperti testano le nuove regole ascoltando gli effetti prodotti su uno spartito musicale. Eventualmente l'esperto chiede allo scienziato di calibrare il funzionamento delle varie regole. Questo processo è iterato finchè l'esperto è soddisfatto del risultato. Un esempio di regola ottenuta applicando il metodo di analysis-by-synthesis è la regola di *Duration-Contrast*, nella quale le note corte sono considerate più corte, e le note lunghe sono considerate più lunghe [26].
Il secondo metodo invece consiste nell'estrarre nuove regole analizzando database di performance. Per esempio consideriamo che vengono usati due database per progettare le regole dell'articulation. Un database consiste di uno stesso pezzo di musica (Andante movement of Mozart's sonata in G major, K 545) suonato da 5 pianisti con 9 diverse intenzioni espressive. Il secondo database invece è fatto da 13 sonate di Mozart eseguite da un pianista professionista. La performance di entrambi i database sono state fatte su un pianoforte a coda monitorato da un computer: un Yamaha Disklavier per il primo database, e un Bosendorfer SE per il secondo database [7] [9].

Il programma Director Musices (**pDM**) è l'implementazione principale del sistema di regole. È stato progettato principalmente per scopi di ricerca, e non è adatto per controlli in real-time. Tuttavia, il controllo in real-time del sistema di regole ha un certo numero

[9] L'Inter-Onset Interval (*IOI*) è il tempo che intercorre tra gli inizio (o i punti di attacco) di note successive.

[10] Il vibrato è un effetto musicale che consiste nella variazione periodica dell'altezza di una nota riprodotta (più precisamente si tratta di una modulazione di frequenza)

di potenziali applicazioni. Potrebbe essere usato per un controllo in real-time dell'espressività nella musica, per esempio uno stile di conduzione (come il VirtualPhilharmony o altri sistemi che verranno suggeriti dopo). Considerando che i sistemi di conduzione precedenti controllavano soprattutto il livello complessivo del suono e del ritmo [54] [6], questo sistema fornirebbe un controllo a livello globale di tutte le caratteristiche espressive, cambiando i dettagli delle performance musicali.
Lo scopo di questo lavoro è stato quello di sviluppare gli strumenti di base per il controllo in real-time dell'espressività delle performance musicali.

### Director Musices (DM) e il metodo utilizzato

Il **Director Musices** è l'implementazione contenente la maggior parte di regole [30]. L'uso di Lisp [11] come linguaggio di programmazione, ha dimostrato di esibire una serie di vantaggi nell'ambito della ricerca. In particolare, è piuttosto rapido ed è in grado di produrre risultati affidabili, grazie ad una serie di funzioni specialistiche e macro che costituiscono l'ambiente di programmazione. Utilizzando il Director Musices, una persona tipicamente carica un file dello spartito, seleziona un insieme di regole (la "*rule palette*", cioè la tavolozza delle regole), applica le regole, e poi ascolta il risultato. Le variazioni risultanti sulla performance possono anche essere visualizzate su vari grafi. Un'ulteriore descrizione di tutte le regole, e di ulteriori riferimenti, è fornita dal manuale del DM [27]. Per il DM non era previsto nessun controllo in real-time dell'applicazione delle regole, per i seguenti motivi

- perché le regole non sono state progettate per essere applicate in real-time; inoltre, le regole ad esempio degli archi, richiedono il contesto dell'intero pezzo per implementare l'intera performance.
- perché l'ambiente Lisp non è una piattaforma adatta per il controllo in real-time. Le regole sono state implementate e testate per molti anni; la prospettiva di re-implementarle in un nuovo linguaggio di programmazione non è molto alettante. Inoltre potrebbe richiedere una discreta quantità di tempo, perché ogni regola deve essere verificata usando molti esempi musicali. Inoltre questo potrebbe portare a due differenti versioni da supportare.

Quindi è stato deciso di applicare un compromesso: le regole sono pre-processate nel DM, producendo così un nuovo formato file (*pDM file*) che contiene tutte le regole delle deviazioni. Questo file pDM è poi eseguito usando un player real-time programmato in *Pure Data* (detto anche Pd [12] [66]).
Come mostrato in Figura 1.5, il metodo è il seguente:

[11] Lisp (List Processor) è una famiglia di linguaggi di programmazione con implementazioni sia compilate sia interpretate, spesso usato nei progetti di intelligenza artificiale. È un linguaggio di programmazione

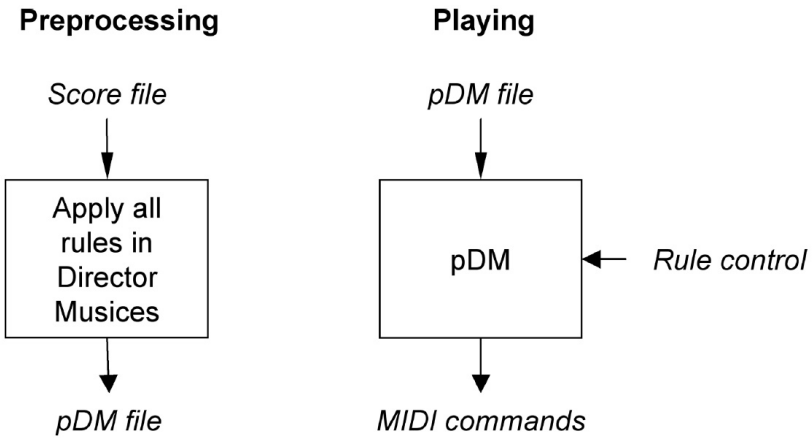



Figura 1.5: Il metodo utilizzato per il controllo real-time.

1. nella fase di pre-processing, tutte le regole vengono applicate una alla volta a uno spartito (dato in ingresso al DM);

2. per ogni regola, le deviazioni generate nel tempo, nel sound level, e nell'articulation, sono raccolte e salvate in un file di tipo pDM;

3. questo file viene poi caricato nel pDM in modo che, nella fase di performance, l'importo/quantità di ogni deviazione può essere controllato e applicato in real-time.

Questo approccio può avere anche dei vantaggi aggiuntivi. Il modulo in real-time è semplice da implementare, perché è essenzialmente un sequencer [13] leggermente esteso con soluzioni tecniche ben conosciute. Inoltre non necessita di potenze di calcolo eccessive, tanto che per l'esecuzione è adatto per dispositivi di piccole dimensioni come palmari o telefoni cellulari.
Il primo prototipo che seguì questa idea venne implementato da Canazza e altri [14], usando la piattaforma EyesWeb [14] [10]. Esso usa lo stesso metodo descritto precedentemente, con un pre-procesing nel DM che raggruppa le deviazioni per ogni regola.

### Applicazione delle regole nel DM

Nel primo passo (**fase di pre-processsing**) tutte le regole vengono applicate ad un dato spartito. Questo è fatto nel DM usando un insieme fisso di regole (di solito quelle di uso

che si basa sul concetto di programma come funzione.

[12] Pure Data (Pd) è un linguaggio di programmazione visuale, creato negli anni 90 da Miller Puckette per la creazione di lavori di musica digitale interattivi e di lavori multimediali. Pd è open source.

[13] Il sequencer è un dispositivo (hardware o software) utilizzato nel campo musicale, che permette di creare e riprodurre delle sequenze di segnali di controllo per comandare uno strumento elettronico.

[14] EyesWeb è una piattaforma open source che supporta lo sviluppo di applicazioni distribuite interattive multimodali e real-time.

generale, vedi tabelle precedenti). La procedura è la seguente:

1. i parametri della performance vengono resettati al loro valore nominale dato dal file dello spartito originale;
2. viene applicata una regola;
3. le deviazioni risultanti sono normalizzate e raggruppate in un vettore.

Questi tre passi sono ripetuti per tutte le regole (delle tabelle precedenti). Dopo, viene scritto il **file pDM** risultante, nel quale vengono inseriti

- il valore della deviazione deltaT (deviazione del tempo);
- il valore della deviazionedeltaSL ( deviazione sonora - sound level - in dB);
- il valore della deviazionedeltaART (articulation in termini di durata in ms dell'offset-to-next-onset)
- i valori nominali dei parametri;
- le note MIDI;
- il channel MIDI.

### Il player pDM

Dal momento che le regole sono situate nel DM, e che poi sono state incorporate nel file di input, l'architettura del pDM è piuttosto semplice. Per ognuno dei parametri della performance (tempo T, sound level SL, e durata della nota DUR), i valori nominali (dati nel file pDM) vengono modificati secondo le seguenti formule :

$$T = T_{nom} * \left(1 + \sum_{i=1}^{14} k_i \Delta T_i\right)$$

$$SL = SL_{nom} + \sum_{i=1}^{11} k_i \Delta SL_i$$

$$DUR = DUR_{nom} + \sum_{i=1}^{5} k_i \Delta ART_i$$

dove

$T_{nom}, SL_{nom}, DUR_{nom}$ sono i valori nominali del tempo, sound level e durata note,
$i$ è il numero della regola,
$k_i$ il fattore di peso per la regola i,

$\Delta T_i, \Delta SL_i$, e $\Delta ART_i$ sono le deviazioni date dal file pDM.
Tutte le deviazione non influenzate dalle regole vengono messe a zero. I valori k sono i pesi applicati dinamicamente che vengono usati per il controllo in real-time. I tre parametri T, SL e DUR vengono calcolati appena prima che ogni nuova nota venga suonata. Quindi, c'è un ritardo (trascurabile) dal controllo al suono. Gli effetti di più regole che agiscono contemporaneamente, vengono applicati in maniera additiva alla stesa nota (quindi si potrebbe ottenere anche la stessa nota di partenza).

### Il controllo in Real-Time

Lo schema a destra della Figura 1.5 si può analizzare meglio guardando qui sotto la Figura 1.6. Come si può vedere durante la **fase di playing** (esecuzione), le regole possono essere controllate o direttamente, o attraverso una mappatura.

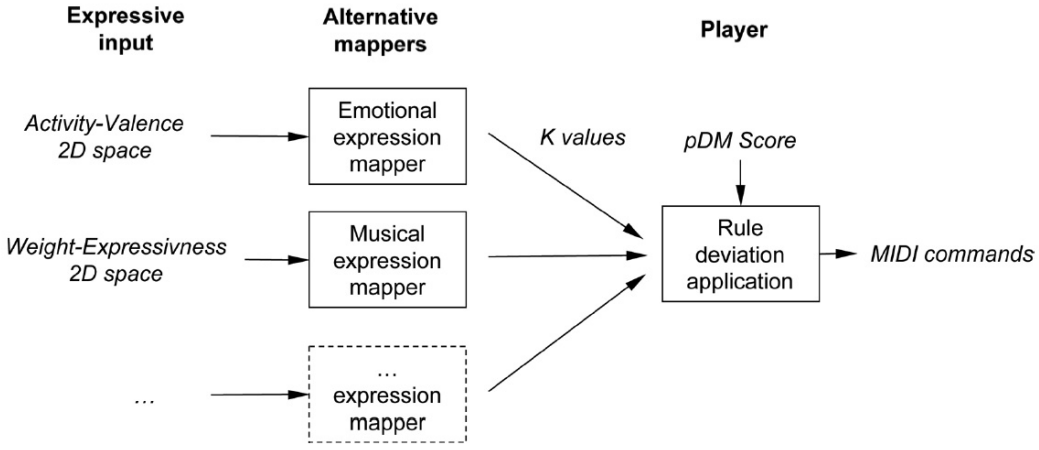



Figura 1.6: Una visione schematica dell'interazione con le regole

Il controllo più diretto è disponibile in termini di parametri k, i quali individuano il peso di ogni regola. Quando k=0 non ci sono effetti della regola, quando invece k=1 gli effetti della regola sono considerati normali. Tuttavia, i valori "normali" sono selezionati arbitrariamente dai ricercatori e dovrebbero essere usati solo come linee guida per la selezione dei parametri. Facendo una selezione di regole e di valori k, possono essere simulati diversi stili di performance e diverse variazioni di performance. Quindi, il sistema di regole dovrebbe essere considerato come un toolbox (cassetta degli attrezzi) per i musicisti, piuttosto che dare un'interpretazione fissa (vedi Figura 1.7)
I valori dei vari k sono modificabili dall'utente, potendo così ottenere delle tavole di valori utilizzabili per un certo pezzo musicale. Inoltre l'utente può ridimensionare gli effetti della regola in ognuno dei parametri influenzati (tempo, sound level e articulation). Per praticità questo controllo non viene incluso nelle formule viste precedentemente.

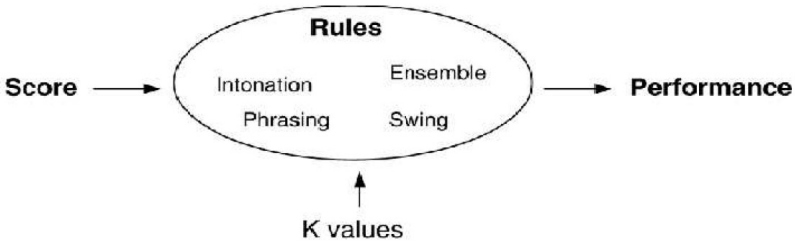



Figura 1.7: Schema di funzionamento del *KTH performance rule system*

Un'importante caratteristica del sistema di regole è che la maggior parte delle regole sono legate alle performance di diversi elementi strutturali della musica [32]. Per esempio, le regole del phrasing rafforzano la divisione in frasi, nonostante sia già apparente nello spartito. Questo indica un'interessante limitazione per la libertà del controllo espressivo: non è possibile violare la struttura musicale interna. Un esempio potrebbe essere quello di fare dei ritardandi e degli accelerandi all'interno della frase. Tuttavia, questo toolbox per produrre elementi strutturali nella musica può anche essere usato per modellare espressioni musicali ad un alto livello semantico.

**Descrittori di alto-livello.** Le performance espressive musicali emotive possono facilmente essere modellate usando diverse selezioni di regole KTH e dei loro parametri, come dimostrato da Bresin e Friberg [8]. Studi in psicologia della musica hanno mostrato che è possibile comunicare intenzioni emotive diverse, manipolando i parametri acustici che caratterizzano uno specifico strumento musicale [46]. Per esempio in una performance pianistica è possibile controllare la durata e il sound level per ogni nota. Negli strumenti a corda e a soffio è possibile anche controllare i tempi di attacco, il vibrato e l'energia spettrale. La tabelle qui sotto mostrano una possibile organizzazione delle regole e dei loro valori k, secondo Juslin, per ottenere delle performance con diverse espressioni con il DM: rabbia (*anger*), felicità (*happiness*), e tristezza (*sadness*).

**ANGER**

| **Expressive Cue** | **Juslin** | **Macro-Rule in DM** |
|---|---|---|
| Tempo | Fast | Tone IOI is shortened by 20% |
| Sound level | High<br>Abrupt tone attacks | Sound level is increased by 8 dB<br>Phrase arch rule applied on phrase level and on sub-phrase level |
| Articulation | Staccato | Duration contrast articulation rule |
| Time deviations | Sharp duration contrasts<br>Small tempo variability | Duration contrast rule<br>Punctuation rule |

**HAPPINESS**

| Expressive Cue | Juslin | Macro-Rule in DM |
|---|---|---|
| Tempo | Fast | Tone IOI is shortened by 15% |
| Sound level | High | Sound level is increased by 3 dB |
| Articulation | Staccato<br>Large articulation variability | Duration contrast articulation rule<br>Score articulation rules |
| Time deviations | Sharp duration contrasts<br>Small timing variations | Duration contrast rule<br>Punctuation rule |

**SADNESS**

| Expressive Cue | Juslin | Macro-Rule in DM |
|---|---|---|
| Tempo | Slow | Tone IOI is lengthened by 30% |
| Sound level | Low | Sound level is decreased by 6 dB |
| Articulation | Legato<br>Small articulation variability | Duration contrast articulation rule<br>Score legato articulation rule |
| Time deviations | Soft duration contrasts<br>Large timing variations | Duration contrast rule<br>Phrase arch rule applied on phrase level and sub-phrase level |
| Final ritardando | | Obtained from the Phrase rule with the next parameter |

### 1.2.5 Altri spazi di controllo 2D basati sugli insiemi di regole

Come si è visto, per il controllo in real-time di una performance è necessaria una mappatura da una descrizione ad alto livello dell'espressione/espressività, ai parametri della regola [41]. Oltre a descrizioni emotive come *happy* (felice), *sad* (triste), *angry* (arrabbiato) e *tender* (gentile, tenero), sono stati implementate anche altre descrizioni: *hard*, *light*, *heavy*, *soft* (rispettivamente duro, leggero, pesante, e morbido). La mappatura però è, necessariamente, una riduzione dello spazio di controllo; quindi, la scelta dello spazio di controllo deve essere un compromesso tra controllo (molte dimensioni di input) e usabilità (nel senso di pochi parametri, ma efficaci).
Ora vengono presentati alcuni esempi di spazi di controllo che usano solo due dimensioni. Usando due dimensioni diventa semplice costruire un'interfaccia, per esempio sullo schermo del computer, nel quale ogni angolo dell'interfaccia è specificato in termini di insieme di regole pesate e corrispondenti ad una certa descrizione. Questa interfaccia 2D può essere facilmente controllata direttamente con il mouse. Come si vede da Figura 1.6, le prime due mappe usano le dimensioni già conosciute (usate precedentemente), per categorizzare le emozioni o per descrivere l'espressività nella musica. In questo modo, hanno un significato intuitivo e semplice da capire per l'utente.

### Lo spazio Activity-Valence

Il primo esempio riguarda la mappa di espressioni emotive di Figura 1.6. Come input la mappa usa lo spazio a due dimensioni Activity-Valence, comunemente consigliato per descrivere le espressioni emotive nell'approccio dimensionale delle scienze dell'emozione [73].

- *Activity* (dimensione orizzontale dello spazio) è legata all'alta o bassa energia;
- *Valence* (dimensione verticale dello spazio) è legata alle emozioni positive o negative.

I quadranti dello spazio possono essere caratterizzati come

felice-*happy* (alta activity, valence positiva),

tenero/gentile-*tender* (bassa activity, valence positiva),

arrabbiato-*angry* (alta activity, valence negativa),

triste-*sad* (bassa activity, valence negativa).

Activity (attività) e Valence (valenza) appaiono spesso come le due dimensioni più di rilievo in esperimenti, per esempio, che applicano l'analisi fattoriale ai giudizi di somiglianza di emozioni nelle espressioni facciali. Questo spazio è la base per il modello di Russel [70], nel quale un grande numero di emozioni appaiono in cerchi all'interno dello spazio. È stato suggerito come particolarmente adatto per le applicazioni musicali, per la sua capacità di catturare continui cambiamenti che si verificano durante un brano musicale [73].

Lo spazio activity-valence è stato implementato definendo un insieme di valori k per ognuno dei quattro angoli, caratterizzati dalle emozioni menzionate prima. Viene usato un numero limitato di sette regole, combinate con il controllo globale del tempo e del sound level. Le regole e i valori dei loro k, sono scelti in modo che comunichino in maniera chiara ogni espressione emotiva [8] [46]. I valori risultanti dei k sono elencati nella Figura 1.8. Questi valori dovrebbero essere considerati solo come delle linee guida per una mappatura adatta.

Quando ci si muove nello spazio, le posizioni intermedie tra i quattro quadranti sono calcolate tramite un'**interpolazione** dei k valori per ogni regola. Ogni valore k è calcolato attraverso un'interpolazione lineare in due passi, causando un piano leggermente inclinato nello spazio a due dimensioni. Se $x$ indica la coordinata dell'activity e $y$ indica la coordinata della valence (variano da -1 a +1), la formula utilizzata per ogni valore di k è

$$k(x,y) = \frac{1}{4}\Big[\big(k_h - k_t - k_a + k_s\big)x + k_h + k_t - k_a - k_s\Big]y + \frac{1}{4}\Big[\big(k_h - k_t + k_a - k_s\big)x + k_h + k_t + k_a - k_s\Big]$$

| *Rule* | *Happy* | *Tender* | *Sad* | *Angry* |
|---|---|---|---|---|
| Phrase Arch 5 | 1 | 1.5 | 3 | –1 |
| Phrase Arch 6 | 1 | 1.5 | 3 | –1 |
| Final Ritard | 0.5 | 0.5 | 0.5 | 0 |
| Duration Contrast | 1.5 | 0 | –2 | 2.5 |
| Punctuation | 1.8 | 1.2 | 1 | 1.4 |
| Repetition Articulation | 2 | 1 | 0.8 | 1.5 |
| Overall Articulation | 2.5 | 0.7 | 0 | 1 |
| Tempo Scaling | 1.1 | 0.8 | 0.6 | 1.2 |
| Sound Level Scaling | 3 | –4 | –7 | 7 |

The tempo scaling parameter defines a tempo factor. The sound level scaling is given in dB SPL.

Figura 1.8: Parametri suggeriti per le regole in termini di valori k, per ognuno dei quattro angoli dello spazio Activity-Valence

Dove $k_h, k_t, k_a, k_s$ si riferiscono ai valori di k definiti nelle posizioni dei quattro angoli (happy, tender, angry, sad), e sono rispettivamente

$$(x, y) = (1, 1), (-1, 1), (1, -1), (-1, -1)$$

Un esempio dei valori k ottenuti dall'interpolazione per la regola "*Phrase Arch*" si possono vedere in Figura 1.9. Lo spazio activity-valence ha il vantaggio di essere semplice da capire

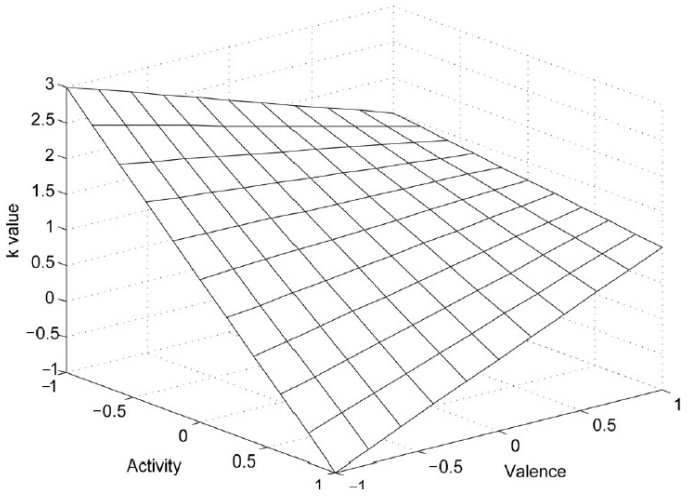



Figura 1.9: Un esempio dei valori k, ottenuti per interpolazione, per la regola *Phrase Arch*

e da usare, anche per utenti musicalmente poco esperti. Questo fa sì che questo spazio sia un buon candidato per applicazioni, come ad esempio per giochi e terapia musicale. Lo svantaggio è principalmente la mancanza di un controllo preciso dei parametri musicali.

### Lo spazio Kinematics-Energy

Lo spazio cinematica-energia [15] è stato suggerito da Canazza et al. [14] per descrivere l'espressività musicale. Questo spazio è emerso in numerose sperimentazioni relative alle performance espressive, caratterizzate da diverse descrizioni/aggettivi/intenzioni musicali come *hard* (duro), *light* (leggero), *heavy* (pesante) e *soft* (morbido). Questo ha comportato sia esperimenti di produzione, in cui ai musicisti era chiesto di suonare uno stesso pezzo con differenti intenzioni musicali, sia esperimenti di ascolto in cui agli ascoltatori è stato chiesto di votare le performance suonate secondo le intenzioni simili.
In termini di parametri musicali, venne trovato che

- *Kinematics* (dimensione orizzontale dello spazio) è associata al tempo;
- *Energy* (dimensione verticale dello spazio) è associata principalmente con le dinamiche [16] e con le articulation [17] (alta energia, staccato).

Utilizzando lo stesso schema di interpolazione visto prima, questo spazio era piuttosto semplice da realizzare. La dimensione dell'energia era mappata sul ridimensionamento del sound level, così come tutte le regole che riguardano l'articulation. La dimensione della cinematica invece, era semplicemente mappata sul controllo globale del tempo [12] [13]. Inoltre, sono stati aggiunti il phrasing [18] e i contrasti di durata [19]. Le regole del phrasing sono state mappate sulla dimensione della kinematics (tempo lento, phrasing largo), mentre le regole dei contrasti di durata sono mappate sulla dimensione della energy (alta energia, contrasto di durata elevato).
Nella Figura 1.10 invece, si possono vedere i valori di k per i quattro angoli dello spazio 2D. Come per lo spazio activity-valence di prima, i risultati ottenuti dalle ricerche sono qualitativi. Perciò i valori attuali sono stati scelti dagli autori testando con alcuni esempi musicali, e dovrebbero quindi servire solo come linee guida.

### Altri spazi di controllo

Usando lo stesso schema d'interpolazione usato nelle mappature activity-valence e kinematics-energy, si può assegnare qualsiasi insieme di parametri di regole ai quattro angoli dello

[15] La cinematica è quel ramo della fisica che si occupa di descrivere quantitativamente il moto dei corpi, senza porsi il problema di prevedere il moto futuro a partire da grandezze note. La cinetica (kinetics) invece è lo studio delle relazioni tra il movimento di corpi e le sue cause (forze e/o momenti).

[16] Alcuni tipi di dinamiche: pp, p, mp, mf, f, ff

[17] Con articulation si indica la tecnica di aggiungere modiche alle singole note o gruppi di note tramite dei simboli di articolazione. Essa consiste nel variare la differenza tra note continue rendendo, ad esempio, la prima nota più corta o sovrapponendola alla seconda.

[18] Il phrasing indica un gruppo di note consecutive alle quali viene applicata una modifica espressiva, ad esempio un'accelerazione nel tempo

[19] Il contrasto tra due valori di nota lunga e breve (ad esempio half note e eighth note), sono qualche volta migliorati dai musicisti, cioè suonano le note corte più corte, e le note lunghe più lunghe rispetto a quello che è scritto normalmente nello spartito.

| *Rule* | *Low Energy, Fast Tempo* | *High Energy, Fast Tempo* | *Low Energy, Slow Tempo* | *High Energy, Slow Tempo* |
|---|---|---|---|---|
| Phrase Arch 5 | 0.5 | 1 | 2 | 1 |
| Phrase Arch 6 | 0.5 | 1 | 2 | 1 |
| Final Ritard | 0 | 0 | 0.5 | 0 |
| Duration Contrast | –1 | 3 | –1 | 3 |
| Punctuation | 1 | 2 | 1 | 2 |
| Repetition Articulation | 1 | 2 | 1 | 2 |
| Overall Articulation | 0 | 2 | 0 | 2 |
| Tempo Scaling | 1.3 | 1.3 | 0.7 | 0.7 |
| Sound Level Scaling | –7 | 7 | –7 | 7 |

The tempo scaling parameter defines a tempo factor. The sound level scaling is given in dB SPL.

Figura 1.10: Parametri suggeriti per le regole in termini di valori k, per ognuno dei quattro angoli dello spazio Kinematics-Energy

spazio 2D. Per fare un esempio, uno spazio potrebbe essere caratterizzato come uno **spazio gesture-energy**, con

- *Gesture*, la componente gestuale, è connessa alle regole di phrasing;
- *Energy*, la componente di energia, è connessa al livello sonoro e al tempo (grande quantità di gesti), oppure potrebbe anche essere connessa all'articulation (piccola quantità di gesti).

I quattro angoli sono etichettati
gentile-*gentle* (tanti gesti, bassa energia),
espressivo-*expressive* (tanti gesti, alta energia),
leggero-*light* (pochi gesti, bassa energia),
forte-*strong* (pochi gesti, alta energia).
I valori di k sono dati dalla Figura 1.11. Questo spazio non ha ragione in termini di investigazioni scientifiche (come quelli visti prima), ma serve come esempio per far capire come potrebbero essere definite nuove interfacce (anche se leggermente diverse), usando i termini usati comunemente in musica.

Naturalmente, tutte le interfacce che usano solo due parametri, saranno limitate in qualche modo rispetto quello che usano più di ue parametri. Certamente l'espressione musicale è un campo ricco e complesso, che coinvolge numeroso possibilità potenziali di controllo. Uno spazio di controllo più preciso sarebbe quello di dividere il sistema di regole nelle sue componenti principali. Questo potrebbe implicare uno spazio a 6 parametri, che include:

- il tempo complessivo;
- il sound level complessivo;
- il phrasing (regole di Phrase Arch, Phrase Ritardando, High Loud, o Final Ritard);

| *Rule* | *Gentle* | *Expressive* | *Light* | *Strong* |
|---|---|---|---|---|
| Phrase Arch 5 | 2 | 2.5 | 0.5 | –0.5 |
| Phrase Arch 6 | 2 | 2.5 | 0.5 | –0.5 |
| Final Ritard | 0 | 0.5 | 0.5 | 0.5 |
| Duration Contrast | –1 | 2 | 1 | 3 |
| Punctuation | 1 | 2 | 2 | 1.5 |
| Repetition Articulation | 1 | 2 | 2 | 2 |
| Overall Articulation | 0 | 0 | 3 | 1 |
| Tempo Scaling | 0.8 | 1.1 | 1 | 1.2 |
| Sound Level Scaling | –7 | 3 | –7 | 7 |

The tempo scaling parameter defines a tempo factor. The sound level scaling is given in dB SPL.

Figura 1.11: Parametri suggeriti per le regole in termini di valori k, per ognuno dei quattro angoli dello spazio Geesture-Energy

- articulation (regole di Punctuation, Repetition Articulation, Overall Articulation, e di Score Legato/Staccato Articulation);

- microtiming (regole di Duration Contrast, Inegales, e di Double Duration);

- tonal tension (le regole di Harmonic Charge e Melodic Charge).

Tuttavia, questo potrebbe essere abbastanza difficile da controllare in real-time, a causa della sua molteplicità di parametri e alla possibilità che i controlli a volte si sovrappongono. Per esempio l'accentuazione di una nota, fornita dal "*Melodic Charge*", può essere ottenuta anche aumentando il livello sonoro complessivo solo per quella nota.

### 1.2.6 Interfacce gestuali ed alcune applicazioni

Lo spazio a due dimensioni descritto sopra è adatto ad essere controllato dal movimento di un cursore sullo schermo del computer. Tuttavia, la musica è intimamente legata ai gesti umani in molti modi [47].

- Primo, la musica suonata con strumenti tradizionali è direttamente influenzata dai gesti usati per il controllo del suono [19].

- Secondo, l'espressività musicale è spesso evidente anche nel linguaggio del corpo del musicista [80].

- Terzo, i direttori (di orchestra) usano i gesti per comunicare le loro intenzioni espressive ai musicisti.

- Quarto, la musica ha l'abilità di evocare sensazioni percettive di movimento nell'ascoltatore [31].

Questi risultati suggeriscono in modo forte che sarebbe meglio e più naturale usare dei gesti umani più complessi, piuttosto che posizionare semplicemente il mouse del computer per controllare l'espressività umana. Un altro vantaggio potrebbe essere che l'attivazione del corpo (e forse anche ballare), rende molto più semplice il coinvolgimento nel processo musicale.
L'idea di usare i gesti per controllare il suono non è nuova, e un numero di interfacce interessanti sono state sviluppate nel passato [79]. Molte di queste interfacce sono orientate verso la creazione di musica, cioè innescare/modificare sequenze di suoni; tuttavia, alcuni di queste interfacce (come la pDM) sono finalizzate alla conduzione di una parte composta.

### Interfacce gestuali con il *KTH Rule System*

Tuttavia, con lo strumento pDM, è possibile ottenere un livello di controllo più raffinato della performance musicale. Le precedenti interfacce per la conduzione, come ad esempio il Radio Baton di Mathews [54], hanno principalmente permesso il controllo del sound level e del tempo. Ora possiamo conservare l'esperienza del musicista e controllare le caratteristiche complessive, come phrasing e articulation, ma in un modo in cui il musicista lo avrebbe interpretato in base al contesto musicale. Questo è molto vicino a una situazione reale che riguarda il ruolo del direttore (di orchestra) e i musicisti.
Recentemente, ci sono stati progressi notevoli nell'estrarre dati di gesti espressivi da una videocamera. Sono stati identificati e modellati diversi gesti utili, per poter descrivere le caratteristiche espressive generali nel movimento umano e nella danza [11]. Questo include la quantità di movimento globale, l'indice di contrazione (arti vicini al corpo, o estesi), e la direzione globale del movimento (su o giù).

Seguendo queste idee, è stata implementata un'interfaccia gestuale. Lo scopo è quello di cercare il modo più semplice possibile di configurazione, in termini di tecniche di riconoscimento e di mappatura. È stata usata una videocamera tipica (QuickCam Pro 4000 - Logitech), e il segnale video è stato analizzato da una patch scritta in EyesWeb [10]. Il primo passo nell'analisi video è quello di prendere le differenze tra frame video consecutivi. L'effetto che si ha è quello che qualsiasi parte che cambia nel video sarà visibile, mentre tutte le parti statiche saranno nere. Questo significa che il sistema riconoscerà solo i movimenti degli oggetti, eliminando le necessità di qualsiasi elaborazione per togliere lo sfondo. Perciò la videocamera può essere usata in qualsiasi stanza senza la necessità di usare luci speciali o schermi di sfondo. Dalla differenza dei segnali video, vengono calcolate due cose:

- la quantità di movimento globale (QoM - Quantity of Motion) in termini del numero di pixel medio visibili nel tempo;

- la posizione verticale, calcolata come il centro di gravità dei pixel visibili.

Queste due quantità sono direttamente mappate nello spazio activiy-valence, dove QoM è connessa con l'activity (alta QoM, alta activity), e la posizione verticale con la valence (posizione alta, valenza positiva). Esperimenti informali confermano che questa semplice interfaccia è davvero molto più divertente da usare, e da vedere, rispetto al mouse. La domanda "*la musica stessa diventa più espressiva*?", è una interessante questione per le ricerche future.

### Il sistema *The Groove Machine*

Come già detto, per le emozioni di base (felicità, tristezza e rabbia), vi è un rapporto piuttosto semplice tra la descrizione emozionale e i valori (cioè i parametri misurati, come il tempo, il sound level e l'articulation). Il primo prototipo che include una prima versione del fuzzy analyser era un sistema che permetteva a un ballerino di controllare la musica cambiando lo stile di danza. Fu chiamato ***The Groove Machine***, ed era stata presentata in una performance a Kulturhuset (Stoccolma, Svezia, 2002). Vengono usati tre tipi di movimento: QoM (quantità di movimento globale - Quantity of Motion), la massima velocità dei gesti nel piano orizzontale, e il tempo tra i gesti nel piano orizzontale. Le emozioni analizzate, come in altre applicazioni, sono rabbia (anger), felicità (happiness) e tristezza (sadness). Il mixing dei tre corrispondenti cicli audio sono direttamente controllati dall'output del fuzzy analyser.

### Il sistema *ExpressiBall*

L'***ExpressiBall***, sviluppato da Roberto Bresin, è un modo di visualizzare la performance musicale nella forma di una palla sulla schermata del computer [32]. Un microfono è connesso al computer, e l'output del fuzzy analyser e dei valori di base vengono usati per controllare l'aspetto della palla. La posizione della palla è controllata dal tempo, dal sound level, e dalla combinazioni delle velocità di attacco e dall'energia spettrale. La forma della palla invece è controllata dall'articulation (arrotondato-legato, e poligonale-staccato), mentre il colore della palla è controllato dall'analisi emozionale (rosso-rabbia, blu-triste, giallo-felice). In Figura 1.12 si può vedere un esempio della palla.
In **Greta Music** la metafora della palla è rimpiazzata da espressioni facciali di Greta [20] *Embodied Conversational Agent* (ECA - agente di conversazione "incarnato") [52]. Qui i descrittori di alto livello (cioè le etichette emozionali), sono mappati in espressioni emotive

[20] http://www.speech.kth.se/music/projects/gretamusic/

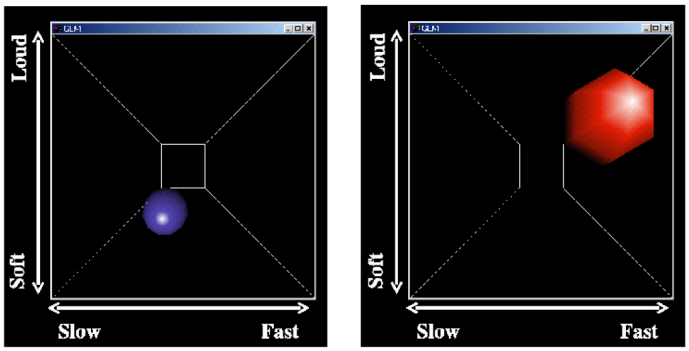



Figura 1.12: Due esempi del sistema *ExpressiBall*, il quale da un feedback visuale della performance musicale

di ECA. I valori dei parametri acustici estratti sono mappati in movimenti di Greta: il tempo è mappato nella velocità dei movimenti di Greta, il sound level nell'estensione spaziale dei movimenti della sua testa (vedi Figura 1.13).

Figura 1.13: Feedback visuale nel sistema Greta Music

### Il gioco *Ghost in the Cave*

Un'altra applicazione che fa uso del fuzzy analyzer è il gioco di gruppo ***Ghost in the Cave*** [69]. L'applicazione usa come input principale di controllo o il corpo o la voce: uno dei processi del gioco è quello di esprimere diverse emozioni usando o il copro o la voce. Entrambe le modalità vengono analizzate usando il fuzzy analiser descritto sopra. Il gioco è giocato in due squadre, ognuna con un giocatore principale (vedi Figura 1.14). Il compito di ogni squadra è quello di controllare l'avatar [21] di un pesce in un ambiente sottomarino, e di andare verso tre differenti grotte. Nelle grotte appare un fantasma che mostra tre diverse emozioni espressive. A questo punto i giocatori principali devono

[21] Nei giochi virtuali l'avatar è un'immagine scelta per rappresentare la propria utenza.

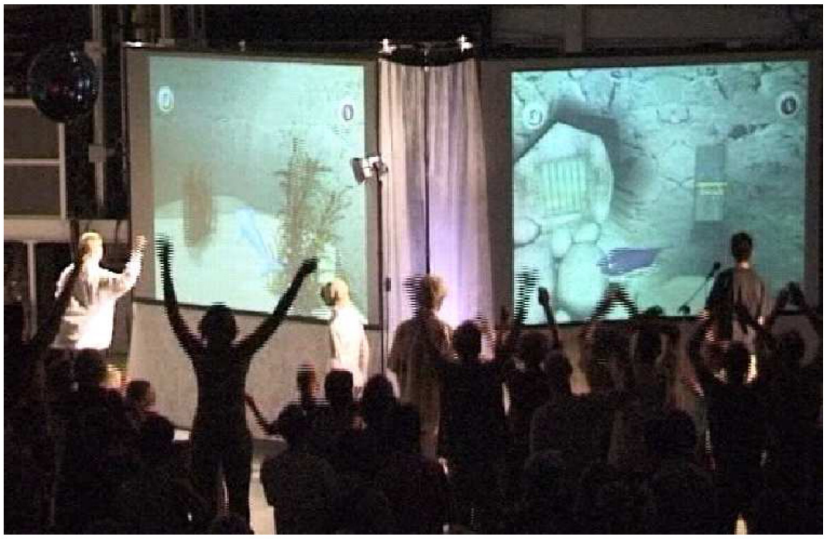


Figura 1.14: Figura relativa alla prima realizzazione del gioco *Ghost in the Cave*

esprimere la stessa emozione, causando di conseguenza il cambiamento del pesce. I punti sono dati per una navigazione più veloce, e per una più veloce espressione delle emozioni in ogni compito. L'intera squadra controlla la velocità del pesce e la musica in base alla loro attività di movimento. Il movimento del corpo e la voce dei giocatori principali sono misurati rispettivamente con una videocamera e un microfono, connessi su due computer nei quali vengono eseguiti due diversi fuzzy analizer. Il movimento della squadra è stimato da una piccola videocamera (una webcam), la quale misura la Quantity of Motion (QoM). Questa QoM è categorizzata in tre livelli (alto, medio, e basso) usando un insieme di funzioni fuzzy. La musica consiste in delle sequenze audio pre-composte, tutte con lo stesso tempo e la stessa chiave, e corrispondenti ai tre livelli di movimento. Le sequenze vengono modificate direttamente dal controllo dell'insieme di funzioni fuzzy. Una squadra controlla la batteria, e l'altra squadra controlla l'accompagnamento.
Il gioco è stato realizzato e sistemato cinque volte dalla sua prima realizzazione alla conferenza *Stockholm Music Acoustic Conference* (nel 2003), includendo anche la *Stockholm Art* e il festival scientifico (Konserthuset, Stoccolma 2004, Oslo University 2004).

### 1.2.7 Il sistema di controllo Home Conductor

Come è stato visto, l'applicazione software pDM permette il controllo in real-time dell'espressività musicale per mezzo di un sistema di regole (*KTH performance rule system*). L'idea generale è che tutto il sapere delle regole è conservato nell'applicazione originale per le regole DM, la quale produce un file pDM. Poi, il software pDM, applica e ridimensiona le diverse variazioni delle regole di performance generate per uno dato spartito durante il

tempo di riproduzione.
Lo sviluppo e il testing del pDM ha innescato molte nuove idee. Precedentemente le regole erano impostate su un'unica tavolozza di regole statica, la quale non cambiava durante il brano, e tali regole statiche funzionavano solo per pezzi musicali corti. Tuttavia, con il controllo real-time, si è aperta la possibilità non solo di fare l' "ultima" performance, ma anche di "condurre" il pezzo attraverso continui cambiamenti dei parametri delle regole. Questo sottolinea la necessità di diverse mappature, semplificando l'interfaccia a pochi parametri ma intuitivi. Sono state suggerite diverse mappature, tra cui lo spazio activity-valence e lo spazio energy-kinematics.

Il direttore di orchestra esprime tipicamente tramite i gesti gli aspetti complessivi della performance, mentre il musicista interpreta questi gesti e inserisce i dettagli musicali. Tuttavia, i sistemi di conduzione precedenti hanno spesso ristretto il controllo al solo tempo e alle dinamiche. Questo significa che i dettagli erano statici e fuori controllo. Un esempio potrebbe essere quello del controllo delle articulation. Le articulation sono importanti per l'impostazione della qualità dei gesti e dei movimenti delle performance, ma non possono essere applicati su base media. L'insieme delle articulation (esempio lo staccato) è impostato su una base note per nota, che dipende dalla linea e dall'insieme della melodia, come riportato da Bresin e Battel [7], e da Bresin e Widmer [9]. Questo rende difficile il controllo diretto per un direttore. Usando il sistema di regole KTH (*KTH rule system*) con il pDM, questi dettagli delle performance possono essere controllati ad un livello più alto, senza la necessità di adattare ogni singola nota. Il sistema di regole è abbastanza complesso, e con un gran numero di parametri. Quindi, quando si effettua un tale sistema di conduzione, il problema principale è la mappatura dai parametri gestuali ai parametri musicali. Recentemente sono stati sviluppati strumenti e modelli per fare un'analisi gestuale in termini di descrizioni semantiche dell'espressività [65]. Quindi, connettendo questo analizzatore di gesti al pDM, si ha un sistema completo per il controllo delle caratteristiche espressive complessive dello spartito. Una panoramica generale di questo sistema è data in Figura 1.15.
Il riconoscimento di espressioni emotive nella musica è stato mostrato di essere un processo semplice per molti ascoltatori, includendo bambini di circa 6 anni anche senza esperienza musicale [64]. Quindi, usando dei semplici descrittori emozionali di alto livello (come happy, sad e angry), il sistema ha la potenzialità di essere intuitivo e facilmente comprensibile per la maggior parte degli utenti, inclusi i bambini. Dunque, invece di un sistema da usare per le performance sul palcoscenico, viene previsto un sistema che sia in grado di essere usato dagli ascoltatori a casa propria. Gli obiettivi principali di progettazione sono quelli di

- un sistema semplice e divertente da usare, sia per gli esperti sia per i novelli;

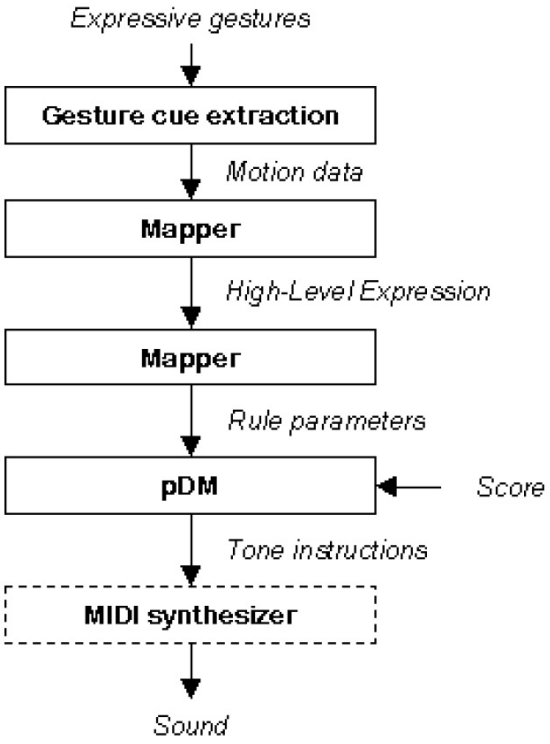



Figura 1.15: Panoramica di un home conducting system.

- realizzare una serie di strumenti standard che richiede un potenza modesta da parte del computer.

Quindi usando degli strumenti come pDM, si può immaginare ad un nuovo modo completamente nuovo di eseguire musica, nel quale i confini tradizionali tra musicista, direttore d'orchestra e ascoltatore sono meno definiti. Può essere costruito un sistema di conduzione d'orchestra "casalingo" (**home conductor**), nel quale l'ascoltatore controlla la performance musicale attraverso i gesti. Più dettagliatamente:

- ***Gesture cue extraction***. Viene usata una piccola videocamera, attraverso la quale vengono misurati per semplicità: la quantità di movimento globale (QoM), le coordinate x-y dell'intero movimento, e la grandezza e la velocità dei gesti verticali e orizzontali (vedi sezione 1.2.6).

- ***Mapping gesture cues to rule parameters***. A seconda dell'applicazione desiderata e dall'abilità dell'utente, le strategie di mappatura possono essere divise in tre categorie:

  1. Level 1 (listener level). L'espressione musicale è controllata in termini di emozioni base (felice-happy, triste-sad, arrabbiato-angry). Questo crea un feedback

musicale ed un interfaccia intuitiva, semplice e comprensibile, senza la necessità di nessuna conoscenza musicale particolare.

2. Level 2 (simple conductor level). Sono controllate le caratteristiche musicali complessive di base, usando per esempio lo spazio energy-kinematics, in quanto è stato precedentemente considerato adeguato per descrivere l'espressione musicale [14].

3. Level 3 (advanced conductor level). Le caratteristiche musicali espressive globali del livello 2, e le espressioni emozionali del livello 1, vengono combinate tra di loro con un controllo esplicito di ogni beat (simile al sistema Radio-Baton).

Usando diversi livelli d'interazione, il sistema si rende adatto per entrambe le voci: bambini, e utenti esperti. Contrariamente agli strumenti tradizionali, questo sistema può "suonare bene" anche per un principiante (quando usa un livello di interazione basso). Si può anche sfidare l'utente a far pratica, nell'ottica di padroneggiare livelli sempre più alti (come accade similmente nei videogiochi).

Un'interessante estensione sarebbe quella di sviluppare un player espressivo per le persone con limitate capacità di movimento. Una precedente esperienza indica che la tecnologia musicale può fornire vari modi per queste persone di accedere alla terapia musicale (musicoterapia [22]) e alla performance musicale [41]. In questo caso, le interfacce di riconoscimento dei gesti devono essere progettate specificatamente in accordo con le capacità di movimento degli utenti.

[22]La musicoterapia è una modalità di approccio alla persona che utilizza la musica o il suono come strumento di comunicazione non-verbale, per intervenire a livello educativo, riabilitativo o terapeutico, in una varietà di condizioni patologiche e parafisiologiche.

# Capitolo 2

# Implementazione di CaRo 2.0

All'inizio del lavoro di tesi il programma CaRo è stato presentato alla versione 1.4. L'applicazione era stata scritta molti anni prima utilizzando come editor C++Builder della Borland, e funzionava solo su macchina con sistema operativo Windows XP. Quindi prima di poter implementare qualsiasi miglioramento è stato necessario effettuare un porting dell'applicazione a sistemi operativi più odierni (Windows 7 o Vista), in modo che il programma potesse essere fruibile anche in futuro per successivi miglioramenti. In questo capitolo, si farà un accenno alla parte teorica che ha portato alla realizzazione del modello utilizzato da CaRo per aggiungere espressività alla musica [36], e poi verrà analizzata in modo molto approfondito la parte implementativa del programma, che ha portato alla release finale CaRo 2.0.

## 2.1 Il modello dell'applicazione CaRo

Un'interpretazione musicale è il risultato della combinazione di intenzioni espressive e competenze tecniche del musicista. Per l'analisi di un'esecuzione musicale è necessario introdurre due sorgenti di "espressività": la prima è relativa agli aspetti tecnici della partitura musicale come il fraseggio e la struttura armonica del brano, la seconda si riferisce alle intenzioni espressive che comunicano stati d'animo e sentimenti. Per enfatizzare alcuni elementi della struttura musicale (ad esempio frasi ed accenti) il musicista modifica la sua performance aggiungendo elementi espressivi come crescendo, decrescendo, sforzando, rallentando, ecc.; in caso contrario l'esecuzione non sarebbe una performance musicale.

Si definisce performance neutra [15], un'esecuzione umana senza una precisa intenzione espressiva, riprodotta in modo scolastico e senza nessun scopo artistico. Il modello è costruito agendo solamente sui gradi di libertà disponibili senza intaccare la relazione fra la struttura musicale e i pattern espressivi. Già nella performance neutra, il musicista introduce un fraseggio (variazioni di tempo e intensità la struttura del brano), ma con lo

sviluppo di questo modello (il modello utilizzato da CaRo) si mira a controllare in modo automatico il contenuto espressivo di una performance neutra pre-registrata.

L'input del modello del sistema è composto da una descrizione della performance neutra e da un controllo che specifica l'intenzione espressiva desiderata dall'utente. Il modello opera a livello simbolico, elaborando le deviazioni di tutti i parametri musicali coinvolti nella trasformazione. In particolare, il rendering [1] è fatto attraverso un sintetizzatore MIDI, mentre un motore di elaborazione audio esegue le trasformazioni sull'audio pre-registrato calcolate dal modello. Il controllo sull'intero sistema viene effettuato attraverso un'interfaccia grafica in cui l'utente può variare, a piacimento, l'espressività di una performance musicale. Nella Figura 2.1 è rappresentato lo schema del modello utilizzato dal Caro.

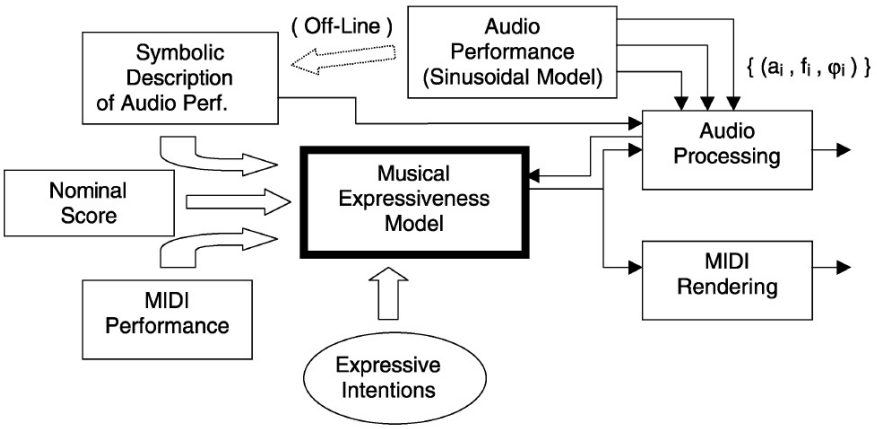



Figura 2.1: Schemda modello utilizzato da CaRo

Il modello descritto è stato applicato a un'ampia varietà di registrazioni monofoniche, dalla musica classica alla musica popolare. Musicisti professionisti hanno suonato diverse partiture musicali ispirati dai seguenti aggettivi: leggero (light), pesante (heavy), morbido (soft), duro (hard), brillante (bright) e cupo (dark). È stata aggiunta anche la performance neutra per poterla utilizzare come riferimento nell'analisi acustica delle varie interpretazioni. A questo punto, il musicista ha scelto la performance che nella sua opinione corrispondeva maggiormente agli aggettivi proposti, in modo da limitare l'influenza che l'ordine di esecuzione avrebbe potuto esercitare sull'esecutore. Tutte le registrazioni sono state effettuate al Centro Sonologia Computazionale (CSC) dell'Università di Padova nel formato monofonico digitale a 16 bit e 44.1 kHz. In totale sono stati considerati 12 pezzi suonati con differenti strumenti (violino, clarinetto, piano, flauto, voce e sassofono) e da vari artisti (circa 5 per ogni melodia). Sono state considerate solamente brevi melodie (dai

[1] Il rendering identifica il processo di "resa" ovvero di generazione di un brano audio (in questo caso) a partire da una descrizione matematica. In genere il termine si riferisce all'ambito della computer grafica.

10 ai 20 secondi) in modo da garantire che i parametri musicali ad alto livello (ad esempio, armonia e metronomo) rimangano costanti. È stata eseguita un'analisi acustica per poter determinare il valore dei parametri che vanno a distinguere i vari aggettivi tra di loro. A partire dai dati trovati, si riesce a ricavare la posizione di ogni aggettivo sullo spazio di controllo; si può quindi utilizzare il modello per modificare iterativamente l'espressività della performance neutra, muovendosi nello spazio di controllo 2-D. In questo modo all'utente è data la possibilità di disegnare qualsiasi traiettoria che rispecchi la variazione espressiva che vuole infondere alla performance neutra (vedi Figura 2.2).

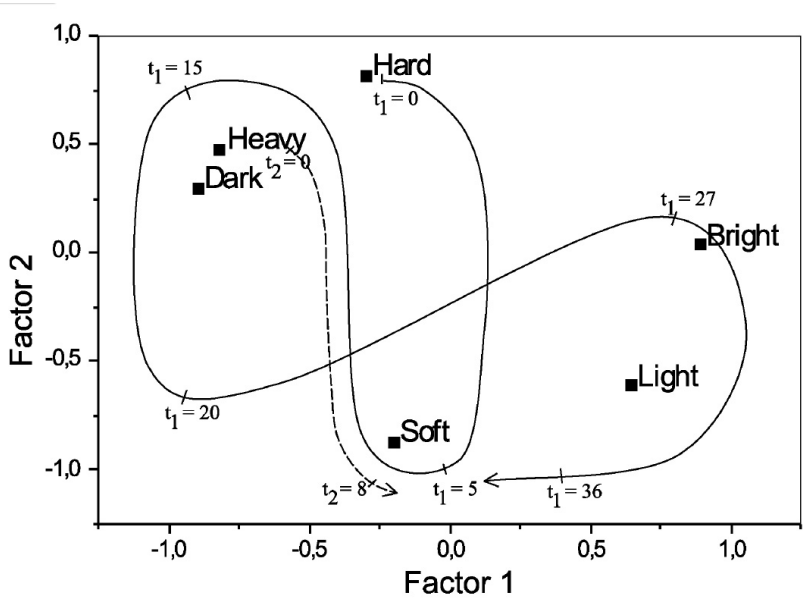



Figura 2.2: Le traiettorie nello spazio 2D corrispondono a diverse evoluzioni nel tempo delle intenzioni espressive della performance. La linea tratteggiata e la linea solida indicano due esempi di traiettorie usate in due performance diverse

Per ulteriori dettagli riguardo il modello e i parametri utilizzati per "infondere espressività" alle performance musicali, fare riferimento all'elaborato di Ganeo Davide [36].

## 2.2 Miglioramenti rispetto la versione precedente

Partendo dalla versione 1.4 di CaRo, la strada più adeguata da percorrere per un miglioramento dell'applicazione è apparsa subito quella di dotare il programma di nuove funzionalità, in modo da offrire all'utente nuovi strumenti.
In vista della conferenza del 6 luglio 2011 SMC Rencon (vedi sezione 3.5), è stato ritenuto molto importante il miglioramento del CaRo da un punto di vista della sua efficienza in termini di capacità di infondere intenzioni espressive ad un brano musicale, in modo che l'applicazione fosse in grado di competere con gli altri competitor partecipanti al

workshop. Inoltre la scelta di migliorare l'efficienza del programma, lo avrebbe reso più appetibile per possibili sviluppi futuri.

Nel lavoro di tesi il gruppo, formato dall'autore di questa tesi Massimiliano Barichello, dal collega Davide Ganeo, e assieme al relatore e ai correlatori Antonio Rodà, Sergio Canazza e Davide Tiso, ha aggiunto e implementato nuove funzionalità al programma. Inoltre è stato sistemato e commentato il codice, in modo che l'applicazione CaRo sia in futuro di più facile e veloce comprensione (in previsione di futuri e ulteriori sviluppi). A tale scopo questo capitolo può essere visto anche come un'utile guida per chi volesse capire come funziona il programma.

## 2.3 Librerie e organizzazione dei file

I problemi inizialmente riscontrati con la versione 1.4, riguardavano l'utilizzo delle librerie MidiShare. Il programma utilizzava vecchie versioni delle librerie:

`mshare.dll mshare32.lib mshare.h(modifica di MidiShare.h)`
`midifile32.dll midifile32.lib e midifile.h`

Quindi il primo passo è stato quello di fare in modo che il programma usasse le nuove librerie MidiShare versione 1.91 win32: [2]

`mshare32.dll mshare32.lib MidiShare.h`
`Player32.dll Player32.lib Player.h`

I file `Player32.lib` e `mshare32.lib` sono stati trasformati col programma `coff2omf`, trovato dentro le cartelle di installazione del C++Builder. Questo perché se un file LIB è della Microsoft (es. `mshare32.lib` e `Player32.lib`), la DLL corrispondente non è in grado di funzionare. La ragione risiede nei problemi di compatibilità tra i file LIB della Microsoft e del C++Builder. Microsoft supporta il Common Object Format File (COFF), mentre Borland C++Builder usa l'Object Model Format (OMF). Per fortuna `coff2omf` permette di creare un file LIB nel formato OMF. Comandi usati:

```
copy MyLib.lib MyBackupLib.lib
coff2omf MyLib.lib MyNewLib.lib
copy MyNewLib.lib MyLib.lib
```

Stesso discorso con mshare32.lib.
I due file LIB sono stati poi linkati al progetto tramite i comandi *Project -> Add to Project* del C++Builder, mentre tutti gli altri file sono stati inclusi nella cartella del progetto.

[2] http://midishare.sourceforge.net/

Per far questo ogni volta che venivano usate funzioni o costanti delle vecchie librerie, si è dovuto effettuare una sostituzione con funzioni delle librerie più aggiornate. La libreria `Player.h` consente la gestione completa di un sequencer MidiShare multi-tracks, permettendo quindi di creare e riprodurre delle sequenze di segnali di controllo (infatti in un sequencer non viene registrato nessun segnale audio, come nei registratori, ma solo segnali di controllo che vanno ad azionare gli strumenti che riproducono il suono). Un Player usa sequenze MidiShare, e fornisce funzioni per leggere e scrivere file MIDI e funzioni per convertirli nel formato delle sequenze MidiShare.

Nella cartella del progetto devono essere presenti anche le 3 applicazioni `msControl32`, `msDrivers`, `msEcho32`, e i file `midishare.ini` e `msMMSystem.ini`. Questi file si trovano nelle cartelle delle librerie e servono ad impostare le connessioni delle porte MidiShare per l'input/output di file MIDI. Il comportamento dei drivers è controllato usando i file INI, che sono simili a dei file di testo con le istruzioni scritte al loro interno. Ogni driver ha degli slot di input e di output, che corrispondono alle porte di input e output che vengono usate. Per impostare correttamente le istruzioni presenti all'interno dei file INI basta usare l'applicazione `msDrivers`, un Drivers Manager che permette di settare le connessioni tra porte MidiShare e gli slot del driver.

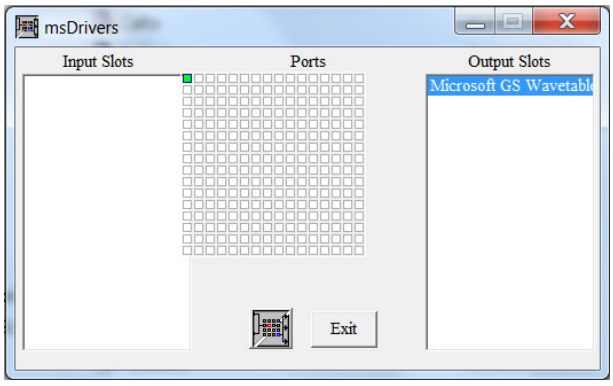



Figura 2.3: Applicazione msDrivers

Effettuando questi passi, è stato possibile compilare il programma CaRo senza errori utilizzando Borland C++Builder (versione 6). [3] La compilazione e l'esecuzione del programma è stata testata con Windows XP, Windows Vista e Windows 7 a 32 bit. Inoltre è stata provata l'esecuzione del programma anche in Windows XP su macchina virtuale (*Oracle* VM *VirtualBox*): l'applicazione non presentava nessun errore di compilazione o di esecuzione, ma durante l'esecuzione non emetteva nessun suono. Molto probabilmente il problema risiede nei driver, che non riescono ad interfacciarsi correttamente con le porte MidiShare.

[3] http://www.embarcadero.com/products/cbuilder

Il progetto costruito con il C++Builder che và a creare l'applicazione CaRo, è composto dai seguenti file:

`XML.h Parabola.h`
`Options.h Options.cpp`
`modello.h modello.cpp`
`Oggettone.cpp`

`XML.h` e `Parabola.h` sono dei header file che contengono rispettivamente i prototipi di funzioni e le variabili delle classi XML e Parabola.
`Options.cpp` contiene il codice relativo al form Opzioni dell'applicazione CaRo (vedi Figura 2.4). `Options.h` è il suo header file corrispondente.

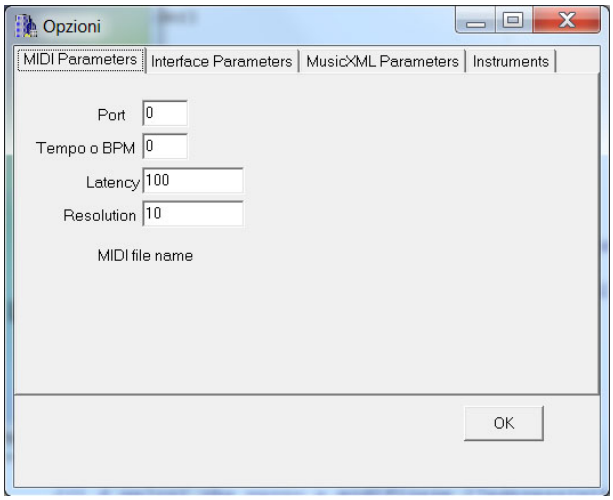



Figura 2.4: form Opzioni

`modello.cpp` è il file più consistente, e contiene la maggior parte del codice dell'applicazione. Al suo interno vengono implementate le funzioni delle classi `XML.h` e `Parabola.h`, e il codice relativo al form Spazio, cioè il form principale all'apertura dell'applicazione CaRo (vedi Figura 2.5). `modello.h` è il suo header file corrispondente.
Un programma creato col C++Builder consiste di un file contenente il codice sorgente del progetto (`Oggettone.cpp`), uno o più file con codice sorgente (`modello.cpp` e `Options.cpp`), uno o più header file (`modello.h`, `Options.h`, `XML.h` e `Parabola.h`), e altri file come le librerie. Il codice sorgente del progetto contiene il main del programma, che viene creato automaticamente dal C++Builder, e di solito effettua le chiamate ai metodi

`Initialize(), CreateForm(), Run()`

che permettono di inizializzare creare a avviare l'applicazione. Nel nostro caso `CreateForm()` viene chiamato sia per il form Spazio sia per il form Opzioni; inoltre all'inizio del file viene utilizzata l'istruzione `USEFORM` per includere tutti i file CPP (`modello` e `Options`), i

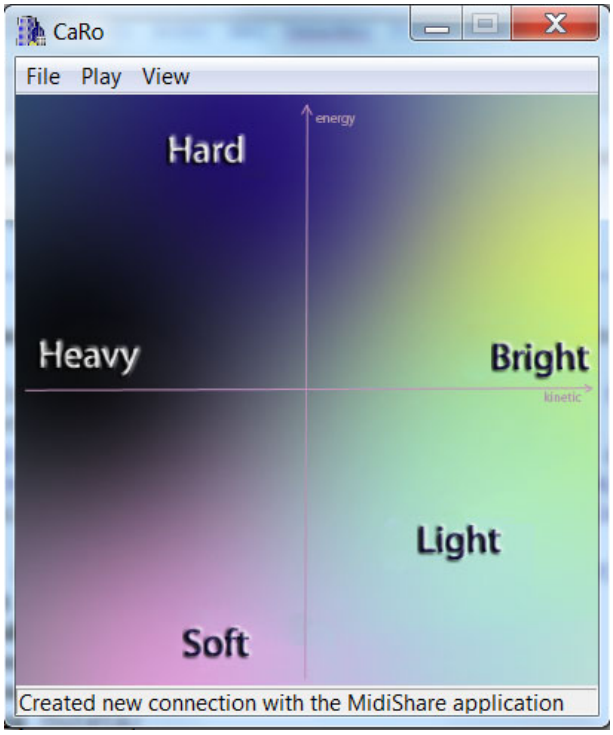



Figura 2.5: form Spazio

quali a sua volta includono gli header file menzionati sopra (`modello`, `Options`, `XML` e `Parabola`).

## 2.4 Panoramica delle funzionalità

All'avvio dell'applicazione, il programma si presenta con una finestra di nome CaRo (vedi Figura 2.5), che sarà l'interfaccia di interazione tra utente e programma. In alto si trova il menu, mentre nella parte centrale c'è un'immagine bitmap di 400x400 pixel che farà da interfaccia interattiva tra utente e applicazione. Tutte le operazione che l'utente può effettuare con CaRo 2.0, si possono suddividere in 3 principali azioni:

1. Esecuzione MECCANICA di un file midi (MIDI);

2. Esecuzione NEUTRA di un file midi (MIDI + MusicXML)

3. Esecuzione ESPRESSIVA di un file midi (esecuzione meccanica + interfaccia, oppure esecuzione neutra + interfaccia)

verrà analizzato in dettaglio il funzionamento e l'implementazione che sta dietro a ciascuna di queste azioni.

## 2.5 Apertura dell'applicazione CaRo 2.0

All'apertura del programma CaRo vengono fatte, dal punto di vista implementativo, alcune operazioni: viene eseguito automaticamente il costruttore della classe `TSpazio`, e viene chiamata la funzione `FormActivate` (è una funzione di `TSpazio` che viene chiamata automaticamente quando attiviamo il form Spazio: in questo caso quando apriamo il programma).

### 2.5.1 Form principale

**TSpazio** è la classe che eredita ogni funzione dalla classe `TForm`; infatti nel file `modello.h` possiamo notare la definizione di eredità della classe `TSpazio` (classe derivata) dalla classe `TForm` (classe base):

```
class TSpazio : public TForm {  ...  };
```

come funzione pubblica aggiuntiva alle funzioni di `TForm`, abbiamo solo il costruttore di `TSpazio`, mentre le funzioni e le costanti private aggiuntive verranno viste man mano che verranno spiegati i diversi pezzi del programma. Spazio è il nome dato al nostro form come oggetto della classe `TSpazio`.

Il costruttore di `TSpazio` inizializza alcune variabili interne, mentre la funzione **FormActivate** apre la connessione tra CaRo e l'applicazione MidiShare (il Player). Un nuovo Player vuoto viene aperto utilizzando la funzione `OpenPlayer`, la quale restituisce il numero di riferimento della corrispondente applicazione MidiShare (questo numero viene inserito nella variabile `ourRefNum`, e viene utilizzato per tutte le funzioni del Player). Le sequenze MidiShare sono caricate sul Player usando la funzione `SetAllTrackPlayer`. Dopo la sua creazione, il Player è collegato all'input/output MIDI. Codice della funzione `FormActivate`:

```
ourRefNum = -1;
//MQmodel è il nome dell'applicazione MidiShare corrispondente
if (ourRefNum = OpenPlayer("MQmodel"))
   StatusBar1->SimpleText = "Created new connection with the
                            MidiShare application";
else
   StatusBar1->SimpleText = "The CaRo application has not been
                            connected to MidiShare application";
//connetto il player (ourRefNum) all'output (lo 0 indica l'output).
//True o 1 indica lo stato on (0 indica lo stato off)
MidiConnect(ourRefNum, 0, true);
//MidiIsConnected restituisce lo stato della connessione tra le due
//applicazioni (è una comunicazione in real-time di eventi midi)
```

```
if (!MidiIsConnected(ourRefNum, 0))
   StatusBar1->SimpleText = "The CaRo application has not been
                             connected to MidiShare application";
```

Il Player verrà poi chiuso usando la funzione `ClosePlayer`, la quale libera la traccia interna al Player e chiude l'applicazione MidiShare.

```
ClosePlayer(ourRefNum);
```

Questo viene applicato all'interno della funzione **FormClose**, una funzione di `TSpazio` che viene chiamata automaticamente quando chiudiamo il form Spazio (cioè quando chiudiamo l'interfaccia di CaRo premendo sulla X in alto a destra), o all'interno della funzione **FileExit**, chiamata quando si preme il tasto Exit dal menù File.

### 2.5.2 Parametri del Player

Di default il Player ha i seguenti parametri:

```
Tempo 120bpm
time signature 4/4
tick_per_quarter 500
```

Significato parametri:

- **Beats Per Minute (BPM)**. Un beat è legato al denominatore della time signature. BPM è anche detto **quarter note per minuto (QPM)**, cioè numero di note da quarto al minuto.

- **Tick o Pulses Per Quarter note (PPQ)**, indica il numero di tick che ci sono per una nota da quarto (o semiminima, o quarter note). Il tick è l'unità di misura del tempo interna al Player.

- **Time signature** (o frazione metrica) è l'indicazione utilizzata dal compositore di musica per segnalare il metro scelto (numeratore) e la notazione adottata (denominatore). Essa è composta da due numeri (es. 4/4): il numeratore indica il numero di beat per ogni misura (la misura o battuta è l'insieme di valori compresi da due linee verticali, dette stanghette, poste sul pentagramma), il denominatore indica a quale nota viene associato 1 beat.

  Esempio: 4/4 indica 4 beats per ogni misura, dove 1 beat è la nota da quarto (quindi la minima, o half note, sono 2 beat).

Calcolo del numero di tick per minuto:

$$BPM * PPQ = \frac{quarternotes}{minute} * \frac{ticks}{quarternote} = \frac{ticks}{minute} = 120 * 500 = 60000 \ \frac{ticks}{minute}$$

Convertire tick in millisecondi:

$$1minuto = 60secondi$$

$$1secondo = 1000ms$$

$$60 * 1000 = 60000 \ \frac{ms}{minute}$$

$$\frac{60000\frac{ticks}{minute}}{60000\frac{ms}{minute}} = 1\frac{tick}{ms}$$

Quindi di default nel Player 1 tick corrisponde esattamente a 1 ms.

In generale il coefficiente di conversione che mi permette di passare da tick a ms è

$$tick2ms = \frac{60000}{PPQ * BPM}$$

### 2.5.3 Form Opzioni

Prima di affrontare i tre tipi di esecuzione del file MIDI, è necessario descrivere il form Opzioni, in quanto il suo contenuto verrà spesso utilizzato per la manipolazione del brano musicale. Questo form è un oggetto della classe `TFormOpzioni`, una classe che, come `TSpazio`, eredita ogni funzione dalla classe `TForm`. L'eredità dalla classe base e i prototipi delle funzioni aggiuntive, si possono trovare nell'header file `Options.h`. Il form Opzioni è visualizzabile dall'utente premendo il tasto *Options..* dal menu *View*, ed è composto da quattro schede:

- *MIDI Parameters*. La prima scheda (vedi Figura 2.6) contiene quattro caselle di testo che individuano la porta della connessione MidiShare, il valore del Tempo (o Beat Per Minute o Quarter note Per Minute), la latenza (parametro in ms utilizzato per spostare gli istanti temporali degli eventi), e la risoluzione (valore in ms di un timer che viene utilizzato durante l'esecuzione di un brano). All'apertura del programma Caro, i valori delle caselle di testo vengono settati rispettivamente a 0, 0, 100, 10. Il settaggio di questi valori viene fatto nel costruttore della classe `TFormOpzioni`, che viene implementato nel file `Options.cpp`. Al momento del caricamento del file MIDI invece il tempo viene di default impostato a 120bpm, mentre gli altri valori rimangono immutati. Se il MIDI viene caricato correttamente c'è la possibilità di visualizzare il percorso del file in fondo alla scheda.

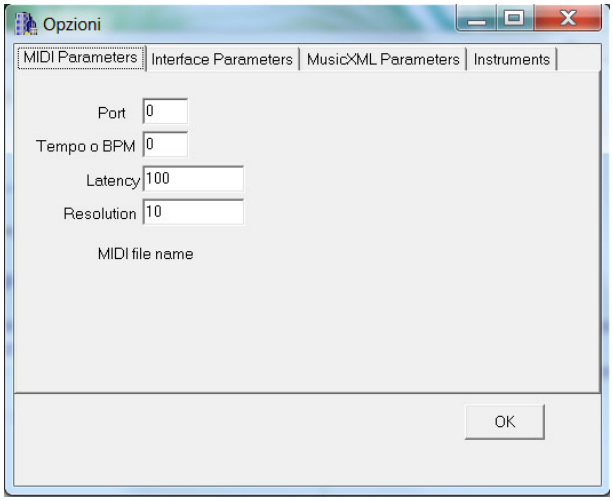



Figura 2.6: scheda "MIDI Parameters" del form Opzioni

- *Interface Parameters*. Nella seconda scheda (vedi Figura 2.7) troviamo tutti i valori che permettono a CaRo di rendere interattiva l'interfaccia per l'esecuzione espressiva del brano in real time. Questi parametri verranno ripresi più avanti nell'implementazione dell'esecuzione espressiva di un brano. Mentre per una spiegazione più approfondita è necessario fare riferimento alla tesi di Ganeo [36].

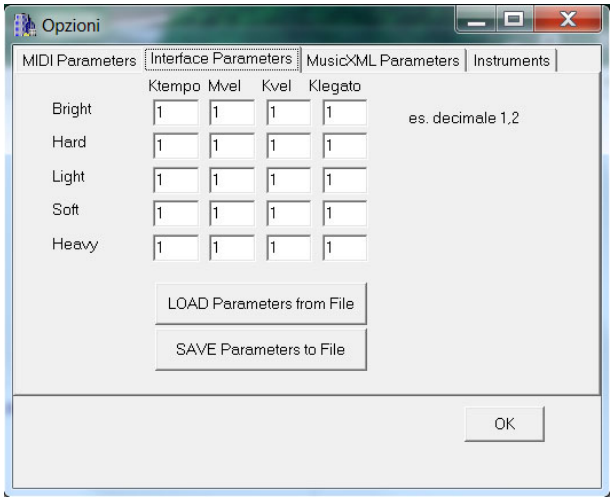



Figura 2.7: scheda "Interface Parameters" del form Opzioni

- *MusicXML Parameters*. Nella terza scheda (vedi Figura 2.8) troviamo tutte le costanti moltiplicative relative ad ogni tipo di informazione che viene acquisita tramite l'uso del file MusicXML; infatti ognuna di queste informazioni va ad agire in alcuni dei parametri del modello utilizzato da CaRo, che permettono la modifica espressiva del brano MIDI. Nella seconda e nella terza scheda del form Opzioni, ci sono due

pulsanti *Load* e *Save*, che permettono di caricare e salvare i parametri delle scheda su file di testo. Queste due schede sono risultate molto utili nella fase di testing del programma CaRo, in quanto permettono la modifica delle costanti moltiplicative durante l'esecuzione del programma, e l'immediato ascolto dei cambiamenti effettuati. I valori di default che contengono le caselle di testo della terza scheda sono riportati nella sezione 2.8.4, mentre in tutte le caselle della seconda scheda è inserito di default il numero 1 (vedi sempre sezione 2.8.4).

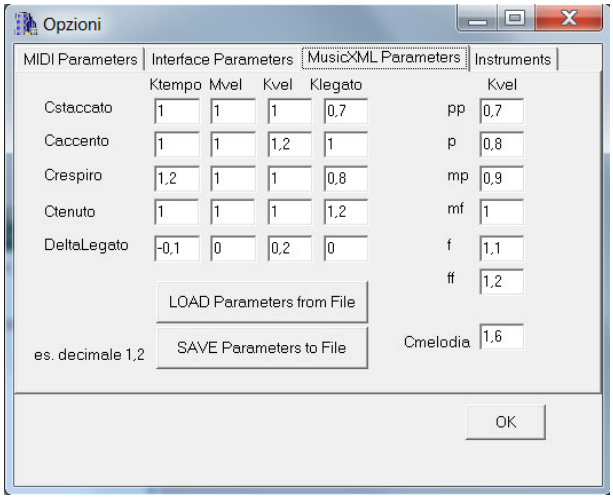



Figura 2.8: scheda "MusicXML Parameters" del form Opzioni

- *Instruments*. Nella quarta scheda (vedi Figura 2.9) è possibile selezionare lo strumento di esecuzione del brano (al momento CaRo non applica nessuna modifica a riguardo), e scegliere tramite una check button indicata con *play only on channel 1* se il Player deve eseguire tutte le note del file MIDI con il pianoforte. Il Player ha di default quest'impostazione attivata, quindi verrà aggiunto un evento, che modifica il timbro, alla sequenza MidiShare solo se tale opzione non è selezionata. All'avvio del programma CaRo l'opzione *play only on channel 1* non è selezionata.

Quando l'utente preme sui pulsanti *Load* o *Save* della seconda e terza scheda, vengono richiamate delle funzioni del form Opzioni. Tali funzioni sono implementate nel file `Opzions.cpp`, e sono in tutto quattro, una per ogni pulsante.

**LoadAdjectivesParametersClick** Questa funzione viene chiamata alla pressione del tasto *LOAD Parameters from File* presente nella seconda scheda del form Opzioni. La funzione apre una finestra di dialogo di Windows per la scelta del file di testo da aprire; se scelto correttamente il file conterrà tutte le informazioni da inserire nelle caselle di testo della seconda scheda. Per farlo basta eseguire una scansione riga per riga del file, estrarre il valore numerico, ed inserirlo nella casella di testo

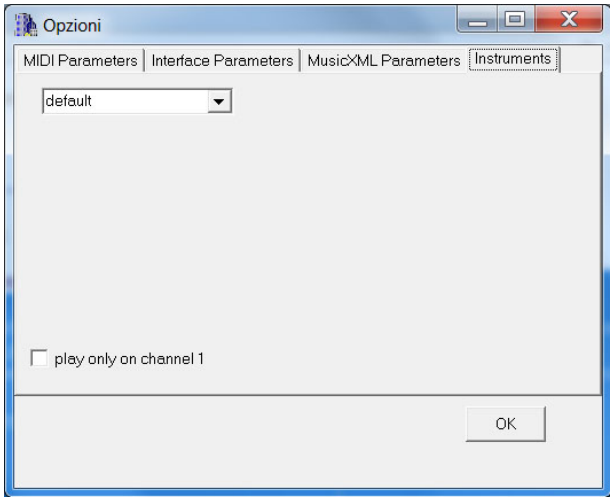



Figura 2.9: scheda "Instruments" del form Opzioni

corrispondente. Esempio del contenuto di un file di testo per la scheda *Interface Parameters* del form Opzioni:

```
0,85 Bright/Brillante - Ktempo
1 Bright/Brillante - Mvelocity
1,25 Bright/Brillante - Kvelocity
0,57 Bright/Brillante - Klegato
1 Hard/Duro - Ktempo
1 Hard/Duro - Mvelocity
1,4 Hard/Duro - Kvelocity
1 Hard/Duro - Klegato
0,9 Light/Leggero - Ktempo
1 Light/Leggero - Mvelocity
0,7 Light/Leggero - Kvelocity
0,6 Light/Leggero - Klegato
1,2 Soft/Morbido - Ktempo
1,4 Soft/Morbido - Mvelocity
0,6 Soft/Morbido - Kvelocity
2,12 Soft/Morbido - Klegato
1,3 Heavy/Pesante - Ktempo
0,5 Heavy/Pesante - Mvelocity
1,5 Heavy/Pesante - Kvelocity
1,4 Heavy/Pesante - Klegato
```

**SaveAdjectivesParametersClick** Questa funzione fa l'esatto contrario della funzione precedente: salva i dati contenuti nelle caselle di testo della seconda scheda del form

Opzioni, in un file di testo simile a quello dell'esempio appena visto. La funzione viene chiamata quando viene premuto il tasto *SAVE Parameters to File*. Per inserire i dati nel file di testo vengono utilizzate le librerie IOStream; infatti la funzione

```
ofstream file(completePath);
```

permette di aprire un file situato in `completePath` in scittura, e di utilizzare la semplice formulazione

```
<< pesKlegato->Text.c_str() << " Heavy/Pesante - Klegato";
```

per inserire le informazioni del form e stringhe all'interno del file.

**LoadMusicXMLParametersClick** Questa funzione viene chiamata appena si preme sul pulsante *LOAD Parameters from File* della terza scheda del form Opzioni. Esegue le stesse operazioni di `LoadAdjectivesParametersClick`, con la differenza che il contenuto del file di testo caricato deve corrispondere alle caselle di testo della terza scheda. Un esempio del contenuto è il seguente:

```
1 Cstaccato0
1 Cstaccato1
1 Cstaccato2
0,7 Cstaccato3
1 Caccento0
1 Caccento1
1,2 Caccento2
1 Caccento3
1,2 Crespiro0
1 Crespiro1
1 Crespiro2
0,8 Crespiro3
1 Ctenuto0
1 Ctenuto1
1 Ctenuto2
1,2 Ctenuto3
-0,1 DeltaLegato0
0 DeltaLegato1
0,2 DeltaLegato2
0 DeltaLegato3
0,7 Dynamics0 pp - pianissimo
```

```
0,8 Dynamics1  p - piano
0,9 Dynamics2 mp - mezzo-piano
1 Dynamics3 mf - mezzo-forte
1,1 Dynamics4  f - forte
1,2 Dynamics5 ff - fortissimo
1,6 Cmelodia
```

Più avanti verrà visto che questo non è un semplice esempio di file di testo, ma i valori riportati all'inizio di ogni riga sono i valori di default che vengono caricati nel form Opzioni all'apertura dell'applicazione CaRo (vedi fine capitolo 2.8.4).

**SaveMusicXMLParametersClick** Questa funzione viene chiamata quando viene premuto il pulsante *SAVE Parameters to File* della terza scheda del form Opzioni. L'implementazione della funzione è molto simile a `SaveAdjectivesParametersClick`, con la differenza che i valori salvati vengono presi dalle caselle di testo della scheda *MusicXML Parameters*.

## 2.6 Esecuzione meccanica di un file MIDI

Con esecuzione meccanica si intende l'esecuzione, da parte del player MidiShare, del brano midi assegnato, cioè la semplice esecuzione delle note che vanno a comporre lo spartito del file midi. Come si può vedere in Figura 2.10, per effettuare tale operazione basta avviare l'applicazione CaRo 2.0, andare sul Menu e premere

*File -> Open...*

A video si aprirà una finestra che permetterà di selezionare un file MIDI dal computer. Fatto questo noteremo nella status bar (nella parte bassa dell'interfaccia) la scritta *MIDI file was read*, ad indicare il fatto che il midi file è stato letto correttamente. Per eseguire il brano MIDI appena caricato (vedi Figura 2.11), basta andare sul menu e premere

*Play -> Play Mechanical*

Inizierà cosi automaticamente l'esecuzione del brano. Se si vuole terminare anticipatamente la riproduzione del MIDI, basta cliccare col mouse *Stop* dal menù *Play*. Ogni operazione eseguita con l'interfaccia del CaRo va a modificare la status bar; questo permette all'utente di capire l'azione appena eseguita, o l'elaborazione che il programma ha appena terminato.

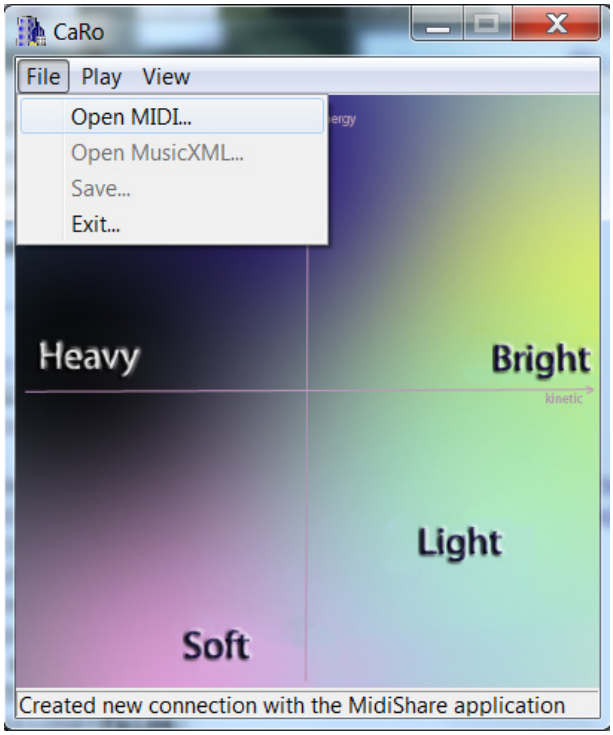



Figura 2.10: Apertura di un file MIDI

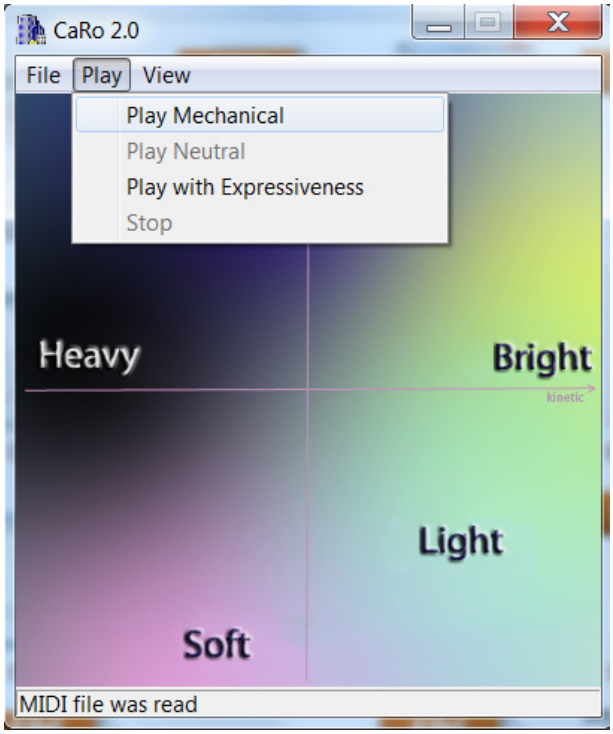



Figura 2.11: Avvio esecuzione meccanica di un file MIDI

### 2.6.1 Apertura del file MIDI

Vengono ora analizzate le parti di codice che permettono di effettuare l'esecuzione meccanica di un file midi.
Alla pressione del tasto *Open...* viene eseguita la funzione **FileOpen**, interna a `TSpazio`. Le librerie grafiche VCL danno la possibilità al C++Builder di interagire con le principali finestre di dialogo di Windows. Nel nostro caso `TOpenDialog` è la classe che permette di aprire un file tramite finestra di dialogo Windows. Quindi attraverso l'istruzione

```
FileOpenDialog->Execute()
```

si apre la finestra di windows nella quale l'utente sceglierà il file midi, e attraverso l'istruzione

```
NomeFile = FileOpenDialog->FileName
```

avremo il percorso della locazione di tale file nel pc. Fatto questo viene richiamata la funzione `LeggiMidiFile`, la quale legge il file MIDI, lo trasforma in una sequenza MidiShare e carica tale sequenza nel Player. Successivamente la funzione `FileOpen` abilita alcuni pulsanti del menù, in modo che possano essere utilizzati.

La funzione **LeggiMidiFile** è stata creata per trasformare il file MIDI (passato in input) in una sequenza MidiShare, che viene poi caricata nel Player e restituita alla fine della chiamata alla funzione. Questo perché nel Player la comunicazione MIDI è basta su messaggi detti eventi. Una sequenza MidiShare è una sequenza di eventi che indicano qualsiasi tipo di informazione usata dal Player (inizio nota, fine nota, ProgChange, ecc.); ogni evento ha un puntatore all'evento successivo (`link`), e questo ci permette di scandire tutta la sequenza. Il Player ci da la possibilità di utilizzare la funzione `MidiFileLoad` per leggere da un file MIDI.

```
MidiFileLoad( itsName, seq, &fdInfo)
```

I parametri in ingresso sono il percorso del file, il puntatore ad una sequenza MidiShare (inizialmente vuota, nella quale viene caricata la sequenza), e un puntatore ad un oggetto di tipo `MidiFileInfos`, il quale contiene alcune informazioni sul file MIDI (numero di tick per nota da quarto, numero di tracce, ecc). Tale funzione restituisce un error code, cioè un codice numerico che viene utilizzato per capire se sono avvenuti degli errori (0=nessun errore, 1=errore di apertura file, 2=errore di lettura file, ...).
La sequenza MidiShare viene poi caricata nel Player attraverso la funzione
`SetAllTrackPlayer`, la quale rimpiazza tutte le tracce presenti nel Player con una nuova sequenza MidiShare passata in ingresso. Se viene caricato un file MIDI multi-track, tutte le tracce vengono messe assieme in un'unica sequenza MidiShare, e vengono distinte tra di loro tramite un reference number (uno dei campi dell'evento MIDI). Una particolare

attenzione deve essere prestata quando utilizziamo `SetAllTrackPlayer`: dopo la sua esecuzione la sequenza MidiShare passata come parametro viene completamente svuotata, quindi è necessario effettuarne una copia prima di utilizzarla. In alternativa è possibile, successivamente, estrarre dal Player una alla volta le track in esso inserite (utilizzando `GetTrackPlayer`), e riunirle in una nuova sequenza MidiShare. Nell'implementazione del programma CaRo viene scelta questa seconda opzione: `GetTrackPlayer` estrae una traccia alla volta come se fosse una sequenza MidiShare, e poi tale traccia viene aggiunta alla sequenza MidiShare `seq` (inizialmente vuota) utilizzando la funzione `AggiungiSeq` creata appositamente. In questo modo viene caricata la sequenza MidiShare nel Player, e la funzione `LeggiMidiFile` può restituire un puntatore (`seq`) a tale sequenza.

Il prototipo della funzione **AggiungiSeq** è il seguente

```
MidiSeqPtr TSpazio::AggiungiSeq(MidiSeqPtr src, MidiSeqPtr dst)
```

Tale funzione incorpora la sequenza MidiShare `src` nella sequenza MidiShare `dst`, restituendo un puntatore alla sequenza risultante (cioè `dst`). Gli eventi all'interno della sequenza `dst` risultante rimangono ordinati a seconda del loro campo `date`, cioè secondo l'istante temporale di ogni evento. `AggiungiSeq` esamina uno ad uno gli eventi della sequenza `src`, e in base al tipo di evento effettua le seguenti operazioni:

- se è un evento di tipo nota (`typeNote`), viene fatta una duplicazione della struttura dell'evento, e tale copia viene aggiunta alla sequenza `dst` utilizzando la funzione `MidiAddSeq`. `MidiAddSeq` è una funzione interna a `MidiShare.h` che permettere di aggiungere un'evento all'interno della sequenza MidiShare mantenendo l'ordine degli eventi in base al loro `date` (istante temporale dell'evento).

- un evento di tipo 'inizio nota' (`typeKeyOn`) e un evento di tipo 'fine nota' (`typeKeyOff`) vengono raggruppati in un unico evento di tipo nota (`typeNote`), e aggiunto alla sequenza `dst`. Il tipo di evento 'fine nota' può essere identificato anche da un evento di tipo `typeKeyOn` con `velocity` nulla. Il campo `duration` del nuovo evento nota sarà calcolato come la differenza dei `date` dei due eventi on e off corrispondenti.

- un evento di tipo tempo (`typeTempo`) viene duplicato e aggiunto alla sequenza `dst`. Questo tipo di evento va a modificare la velocità di esecuzione del file midi dato in ingresso; più precisamente va a modificare il BPM, cioè il numero di note da quarto per minuto. Per questo motivo viene anche modificato il valore del tempo nel form Opzioni con tale valore di BPM. Nell'evento `typeTempo` però il valore del tempo è in microsecondi per nota da quarto, quindi è necessario trasformarlo in un valore che indica il numero di note da quarto per minuto:

```
1 microsecondo = 0,001 ms = 0,000001 s
1 minuto = 60 secondi = 60000000 microsecondi
FormOpzioni->tempo->Text=60000000.0/(float)MidiGetField(e,0);
```

- un evento di tipo program change (`typeProgChange`) va ad impostare un specifico timbro musicale (o program) ad un dato channel. Nel form Opzioni è presente una check button indicata con *play only on channel 1* che se selezionata, fa in modo che il Player esegua tutte le note del file MIDI col pianoforte. Essendo questa un'impostazione di default del Player, tale evento viene aggiunto alla sequenza `dst` solo se tale opzione non è selezionata, in modo tale che il Player possa modificare il timbro in base a tale evento.

### 2.6.2 Esecuzione meccanica

Una volta caricato il file MIDI nell'applicazione CaRo, il brano è pronto per essere eseguito in maniera meccanica. Alla pressione del tasto *Play Mechanical* dal menù, viene chiamata la corrispondente funzione **PlayMechanical** di `TSpazio`. Tale funzione non fa altro che avviare il Player attraverso la seguente riga di codice

```
StartPlayer(ourRefNum);
```

La funzione `StartPlayer` è interna al Player (passato in input), e permette l'esecuzione del brano MIDI precedentemente caricato partendo dall'inizio.

Se l'utente vuole terminare anticipatamente l'esecuzione del file Midi, deve premere il pulsante *Stop* dal menù, in modo che venga chiamata automaticamente la corrispondente funzione **MenuPlayStop** di `TSpazio`. Questa funzione richiama una funzione del Player

```
StopPlayer(ourRefNum);
```

la quale ferma il Player bloccando l'esecuzione del file MIDI corrente, e manda al Player tutti quegli eventi mancanti che sono legati ad eventi già interni al Player (ad esempio gli eventi di tipo `typeKeyOff`). Per concludere in `MenuPlayStop`, c'è una parte di codice che verrà utilizzata solo quando viene fatta un'esecuzione espressiva del file MIDI: si tratta di alcune righe di codice che chiedono all'utente se salvare il brano espressivo appena eseguito; in caso affermativo verrà chiamata una funzione apposita che permette di eseguire questa operazione (vedi la sezione 2.9).

## 2.7 Esecuzione neutra di un file MIDI

Come già accennato ad inizio capitolo, una performance neutra [15] è un'esecuzione umana senza una precisa intenzione espressiva, nella quale il musicista introduce già di

suo un fraseggio (variazioni di tempo e intensità nella struttura del brano). In CaRo 2.0, con esecuzione neutra si intende l'esecuzione da parte del Player del brano assegnato, prendendo informazioni sia dal file MIDI, e sia da un file MusicXML. L'interfaccia di Caro infatti permette, attraverso il menù, di caricare un file MIDI e facoltativamente di caricare anche un file MusicXML. Se quest'ultimo viene caricato avremo attiva anche l'opzione per un'esecuzione neutra del file MIDI, oltre all'esecuzione meccanica del brano musicale.

### 2.7.1 MusicXML e Finale

MusicXML è un sistema di codifica XML (eXtensible Markup Language) in grado di rappresentare il sistema di notazione musicale occidentale, cioè quello attualmente utilizzato. L'XML è un linguaggio di markup aperto che viene utilizzato per creare nuovi linguaggi, usati per descrivere documenti strutturati. MusicXML, in quanto codifica XML, è quindi un linguaggio libero di essere usato da tutti sotto licenza senza diritto di autore (royalty-free), e offre le potenzialità dell'XML quali la strutturazione dei dati, la modularità, e l'estensibilità. Lo scopo e l'uso principale di MusicXML è l'interscambio di spartiti musicali su internet. Mentre il formato MIDI (Musical Instrumental Digital Interface) nasce come supporto alle performance musicali, ed è un linguaggio informatico che da vita al protocollo di interazione tra strumenti musicali elettronici diversi, il MusicXML si propone come uno standard di codifica dello spartito musicale: una volta codificato, lo spartito può essere considerato come un database semi-strutturato e quindi interrogato e rielaborato.
L'obiettivo che ci ha portato all'uso di questo tipo di file è il seguente

> La miglioria da dare al programma CaRo originale (1.4) era quella di poter dare delle informazioni aggiuntive al Player, oltre al file MIDI dato in ingresso, allo scopo di poter dare maggiore espressività esecutiva al file audio. Il modo più immediato per far questo era quello di prendere lo spartito musicale corrispondente al file MIDI in ingresso, aggiungere delle informazioni alle note di tale spartito, salvare in qualche modo queste informazioni in un file, e poter usare al momento opportuno tali informazioni, cioè durante l'esecuzione del file MIDI da parte del Player.

Lo strumento utilizzato per aggiungere informazioni allo spartito musicale è il software **Finale**.
Finale è un programma di impaginazione musicale per spartiti, usato sia da dilettanti che da professionisti del campo musicale come editor di musica digitale. Il programma permette di visualizzare lo spartito di qualsiasi file con estensione MIDI, e di salvare il documento in più formati come MUS (formato proprietario) e MIDI. Inoltre da la possi-

bilità di importare ed esportare anche file di tipo MusicXML. Quest'ultima caratteristica e la facilità d'uso del programma, fanno si che sia stato scelto per eseguire quello di cui abbiamo bisogno. Quindi i passi preliminari che l'utente dovrà fare prima di usare CaRo, per ottenere un'esecuzione neutra del file MIDI, saranno i seguenti:

1. aprire con Finale il file MIDI che sarà dato in input al programma CaRo 2.0. Al momento dell'apertura si può decidere se mettere ogni traccia in un pentagramma diverso (*Tracks become staves*), o se riunirle in un'unica riga. Noi abbiamo scelto la seconda (vedi Figura 2.12), quindi i passaggi per mettere tutte le tracce nello stesso pentagramma sono: *Set Track-To-Staff list* -> *AutoSet to Tracks* -> selezionare la traccia -> scegliere *None* nella sezione *split*.

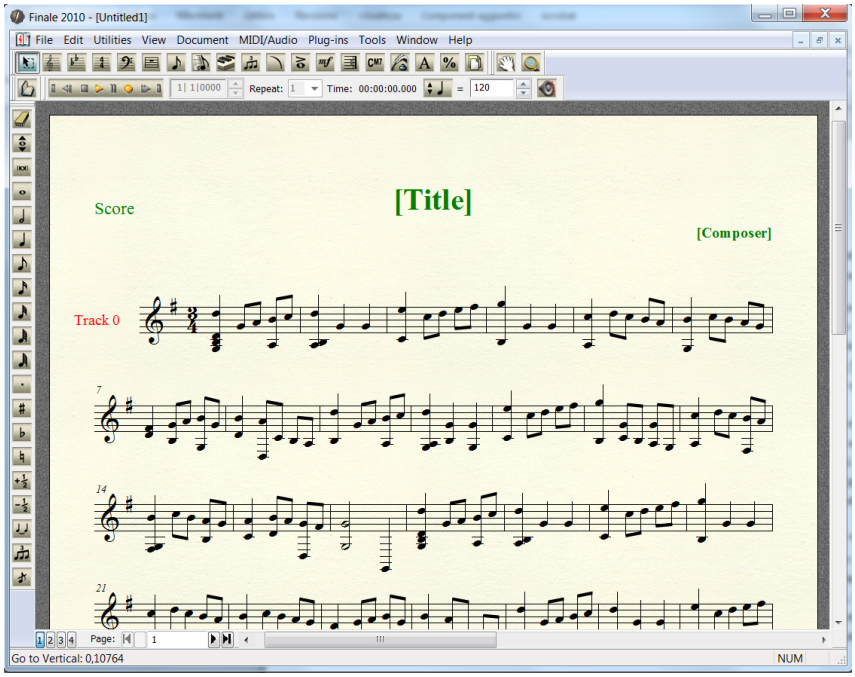



Figura 2.12: Apertura di un file MIDI con Finale

2. aggiungere allo spartito musicale tutte le informazioni espressive utilizzando gli strumenti offerti dal software Finale (vedi Figura 2.13). Il programma CaRo 2.0 non è in grado di riconoscere ed elaborare tutte le possibili informazioni che Finale permette di aggiungere. Le informazioni aggiuntive che CaRo riconosce sono le seguenti:

| dinamiche (pp, p, mp, mf, f, ff) |
|---|
| pedale |
| tie |
| staccato |
| tenuto |
| respiro |
| accento |
| slur (o legato) |

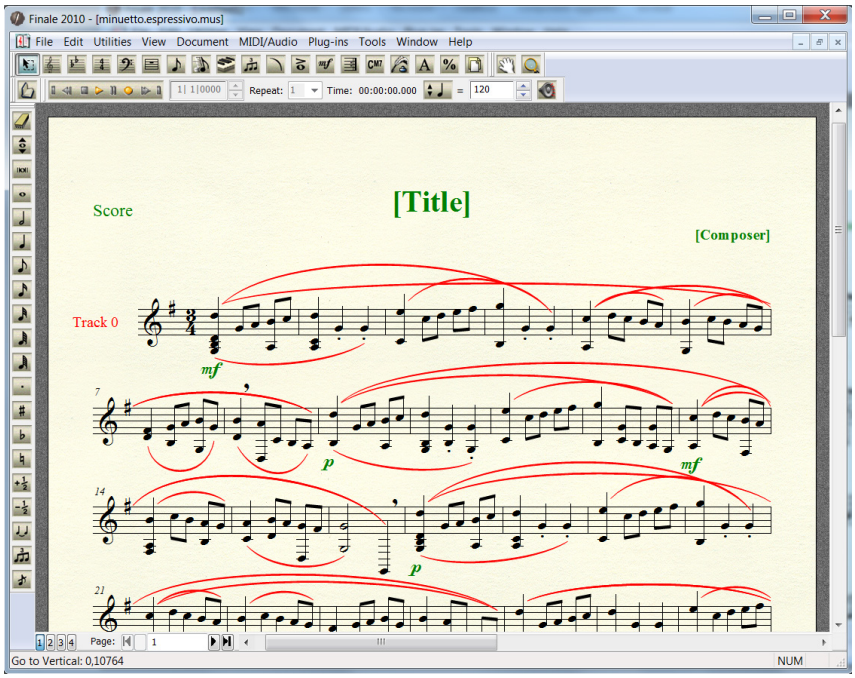



Figura 2.13: Aggiunta di informazioni espressive allo spartito musicale (articulation)

3. salvare con Finale lo spartito in formato MusicXML. Per farlo andare su *File -> MusicXML -> Export*. Se in futuro si volessero aggiungere ulteriori informazioni espressive allo spartito è consigliabile salvare anche in formato MUS; tale formato è proprietario a Finale, e permettere al programma di aprire lo spartito nelle medesime condizioni iniziali. Finale ha anche la possibilità di importare file MusicXML, ma il programma aggiunge automaticamente delle modifiche allo spartito che lo renderebbero diverso dall'originale. Per questo motivo è preferibile salvare in formato MUS.

Completati questi 3 passaggi l'utente può utilizzare CaRo avendo così a disposizione il file MIDI, e il corrispondente file MusicXML. Per ottenere un'esecuzione neutra, l'utente deve

caricare il file MIDI con CaRo, e caricare il file MusicXML corrispondente selezionando la voce *Open MusicXML* dal menu *File*. Per l'esecuzione del brano ci sono due possibilità disponibili nel menu:

- *Play Mechanical* permette di eseguire il brano MIDI senza informazioni espressive aggiuntive (è l'esecuzione meccanica spiegata precedentemente).
- *Play Neutral* permette di eseguire il brano MIDI con le informazioni aggiuntive estrapolate dal file MusicXML (è l'esecuzione neutra del brano).

Da notare che il caricamento del file MusicXML è un'operazione opzionale, mentre il caricamento del file MIDI è obbligatoria.

### 2.7.2 Classe XML

Dal punto di vista implementativo, l'aggiunta del file MusicXML comporta l'esigenza di una struttura dati che sia in grado di estrarre le informazioni inserite in questo tipo di file, di elaborarle, e di utilizzarle al momento opportuno. Per questo fine è stata creata una classe (chiamata `XML`), che offre una semplice struttura dati per organizzare e memorizzare tali informazioni. La scelta del tipo di dati da adottare per il salvataggio, e i metodi da utilizzare per l'estrapolazione e l'applicazione di tali informazioni, ha reso necessario una minima conoscenza delle informazioni musicali che possono essere aggiunte da Finale allo spartito musicale. Qui di seguito verranno illustrati tutti i componenti e le procedure che permettono a CaRo 2.0 di utilizzare i file MusicXML.

`XML.h` è un header file che contiene i prototipi delle funzioni e delle variabili che vengono utilizzate dalla classe `XML`. In CaRo, ad ogni nota dello spartito musicale viene associato un oggetto di tipo `XML`, solo se tale nota ha delle informazioni che sono state aggiunte con Finale. Per questo motivo `XML.h` ha la seguente struttura:

```
class XML{
   private:
      int ID;
      int respiro;
      int accento;
      int tenuto;
      int staccato;
      int legato[10][3];
      int numLegato;
      char nota;
      int dinamica;
      int pedale;
```

```
    public:
        XML();
        int getID();
        void setID(int i);
        int getRespiro();
        void setRespiro(int r);
        int getAccento();
        void setAccento(int a);
        int getTenuto();
        void setTenuto(int te);
        int getStaccato();
        void setStaccato(int d);
        int getLegatoNumber(int riga);
        int getLegatoStart(int riga);
        int getLegatoStop(int riga);
        void setLegatoNumber(int riga, int n);
        void setLegatoStart(int riga, int start);
        void setLegatoStop(int riga, int stop);
        int getNumLegato();
        void setNumLegato(int num);
        char getNota();
        void setNota(char n);
        int getDinamica();
        void setDinamica(int d);
        int getPedale();
        void setPedale (int p);
};
```

ID. L'ID è l'identificativo della nota, che la distingue dalle altre note. Ad ogni nota dello spartito viene associato un numero intero partendo da 0, se poi successivamente vengono aggiunte delle informazioni ad una nota con Finale, il suo corrispondente numero intero diventa il suo ID.

respiro. In inglese è detto anche Breath Mark, e il suo simbolo è una virgola posta appena dopo la nota interessata (vedi Figura 2.14). Negli strumenti a fiato o a voce, il respiro indica un momento in cui si può prendere respiro, mentre per gli strumenti non a fiato indica il momento in cui si può prendere una breve pausa. Nel nostro caso tale variabile può assumere due valori: 1 se la nota corrispondente ha un respiro posto appena dopo, 0 altrimenti.

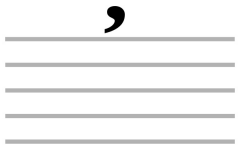


Figura 2.14: Respiro

accento. L'accento dinamico viene indicato graficamente con la parentesi angolata > (accento orizzontale o accento lungo), e viene posto sopra o sotto una nota (vedi Figura 2.15). La nota con l'accento deve essere eseguita con una intensità sonora più forte. La variabile accento vale 1 se la nota corrispondente ha un accento, 0 altrimenti ("La lezione di chitarra", n.d).

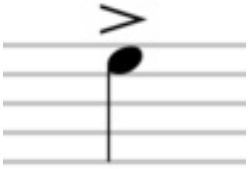


Figura 2.15: Accento orizzontale

dinamica. In musica le dinamiche sono dei simboli posti nello spartito che indicano con quale intensità devono essere eseguite le note successive (vedi Figura 2.16).

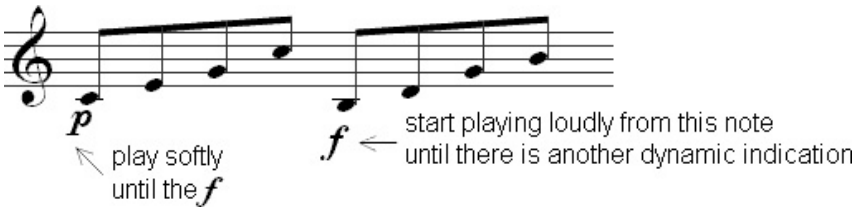



Figura 2.16: Simboli di dinamiche nello spartito

Nel pentagramma vanno messi nel punto dopo il quale vogliamo che le note vengono eseguite con quell'intensità specificata dalla dinamica; in questo caso verrà creato un oggetto XML per tutte le note successive a tale dinamica, in modo che possono contenere le informazioni per l'applicazione della dinamica. Nel programma vengono gestite solo alcune tra le tante dinamiche disponibili. La variabile può contenere i seguenti valori:

| valore | simbolo | significato |
|---|---|---|
| 0 | nessuna dinamica | |
| 1 | pp | pianissimo (very soft) |
| 2 | p | piano (soft) |
| 3 | mp | mezzo piano (moderately soft) |
| 4 | mf | mezzo forte (moderately loud) |
| 5 | f | forte (loud) |
| 6 | ff | fortissimo (very loud) |

`tenuto`. Nel pentagramma è indicato con un trattino posto sopra alla nota (vedi Figura 2.17). Con il tenuto la nota diventa più lunga e un po' più forte (come un leggero accento). Tale variabile vale 1 in presenza dell'accento nella nota corrispondente, 0 altrimenti ("MusicArrengers", n.d.).

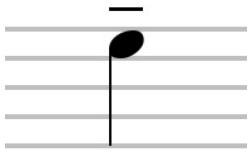


Figura 2.17: Tenuto

`staccato`. Detto anche Dot Staccato. Viene indicato con un puntino posto sopra o sotto ad una nota (vedi Figura 2.18). Tale simbolo indica che la nota è separata o sconnessa dalle note successive, quindi richiede un'esecuzione più breve della nota, normalmente la metà della sua durata originale ("MusicArrengers", n.d.).

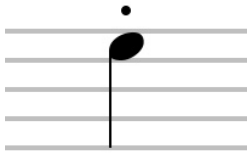


Figura 2.18: Staccato

`pedale`. La funzione principale del pedale è quella di lasciar risuonare una nota anche dopo aver abbandonato il tasto ("Tecnica pianistica: il pedale", n.d.). Nel programma l'informazione del pedale viene associata alla nota immediatamente successiva al simbolo

start-pedale o stop-pedale inserito nello spartito (vedi Figura 2.19). La variabile ha i seguenti possibili valori:

| valore | significato |
|---|---|
| 0 | nessun pedale |
| 1 | start pedale |
| 2 | stop pedale |
| 3 | start pedale e stop pedale nella stessa nota |


Figura 2.19: Simboli che indicano l'inizio e la fine del pedale

`legato[10][3]`. Detto anche Slur. Nel pentagramma è indicato con un arco tra due note (anche con pitch diverso), o tra un gruppo di note. Un'altra espressione musicale che viene indicata con lo stesso simbolo è il `Tie`, un arco che collega due note con lo stesso pitch. Il programma CaRo non memorizza in nessuna struttura le informazioni di eventuali tie inseriti nello spartito, ma gestisce comunque la presenza di tie al momento della lettura del file MusicXML. Tie e legato indicano che l'esecuzione deve essere smoothly, cioè le note di una frase melodica devono essere eseguite accostate l'una all'altra senza interruzioni, in modo che il suono dell'una inizi non appena cessa quello della nota precedente ("MusicArrengers", n.d.). La variabile `legato` memorizza solo le informazioni riguardante gli slur inseriti nello spartito musicale. Tale variabile è una matrice 10x3 che viene abbinata ad una nota nella quale inizia o finisce un legato (se il legato raggruppa più di due note, le note intermedie non avranno nessuna matrice associata). La scelta di una matrice come variabile risiede nel fatto che da una nota possono partire o finire più legato contemporaneamente. Ad ogni riga della matrice corrispondono le informazioni di uno solo dei legato presenti in quella nota. Nella prima colonna viene indicato il `number` (l'id) del legato in quella nota, nella seconda colonna viene messo 1 se il legato parte dalla nota (0 altrimenti), e nella terza colonna viene messo 1 se il legato finisce in quella nota (0 altrimenti).
Se ad esempio dalla nota partono 3 slur e terminano 2 slur (vedi Figura 2.20), allora la matrice è composta da 5 righe. I number di ogni riga sono estratti dal file MusicXML, e di solito sono dei numeri interi (quindi essendo nell'esempio all'inizio dello spartito i number andranno da 1 a 5).

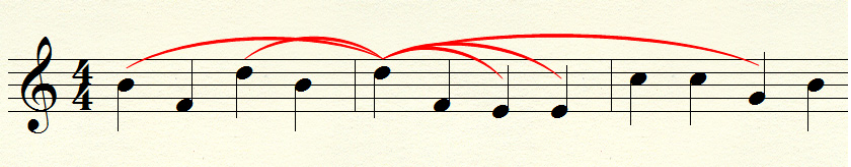


Figura 2.20: Esempio di note con dei legato

La matrice supporta al massimo 10 slur per ogni nota; in caso di necessità basta aumentarne le dimensioni.
`numLegato`. Tale variabile contiene il numero di legato presenti nella nota corrente. Questa informazione diventa utile nel momento in cui si vuole scandire la matrice dei legato (ad esempio per stamparne il contenuto).

`nota`. Contiene il valore del pitch della nota. Il pitch indica la posizione in verticale della nota nello spartito musicale, e nel file MusicXML viene indicato attraverso una lettera maiuscola (vedi Figura 2.21).

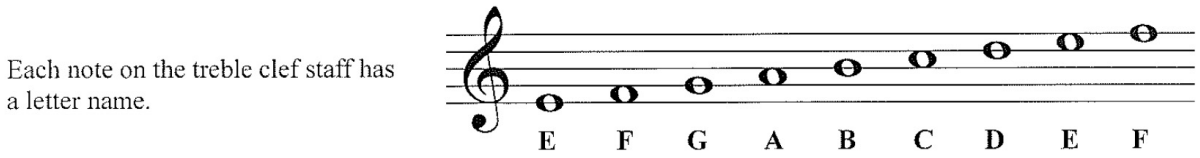



Figura 2.21: Pitch e relative lettere

Nella parte pubblica della classe `XML` oltre al costruttore ci sono un elenco di funzioni che permettono l'estrazione e l'inserimento dei valori delle variabili private sopra citate. Tali metodi sono implementati nel file `modello.cpp` che, come già detto in precedenza, include l'header file `XML.h`. La parte implementativa della classe è molto semplice: il costruttore pone a 0 tutte le variabili (compresi gli elementi della matrice), i metodi `get` ritornano il valore della variabile, mentre i metodi `set` impostano il valore della variabile corrispondente in base al valore passato come parametro.

### 2.7.3 Parser del file MusicXML

Verrà ora analizzato cosa succede quando l'utente effettua un'esecuzione neutra del file MIDI. Dopo aver aperto il MIDI con CaRo, l'utente carica il file MusicXML corrispondente premendo su *Open MusicXML* dal menu *File*. Alla pressione del tasto viene chiamata automaticamente la funzione **MenuOpenMusicXMLClick** interna a `TSpazio`. Anche qui viene usata una finestra di dialogo Windows per la scelta del file, e poi viene chiamata

la funzione **ParserMusicXML**, creata appositamente per interagire con il file MusicXML. Quest'ultima funzione ha in input il percorso del file MusicXML, scandisce tutte le righe del file, ed estrae le informazioni utili per poi inserirle in oggetti della classe `XML`. Per ogni nota del brano a cui sono associate delle informazioni aggiuntive, viene creato un oggetto `XML` ed inserito in un vettore dinamico (contenitore), in modo che tali oggetti possono essere utilizzati successivamente dal programma. Il vettore offre metodi come `push_back()` per inserire un elemento alla fine del vettore allocando memoria se necessario, `pop_back()` per eliminare l'ultimo elemento, e `size()` per conoscere il numero di elementi inseriti. Altre variabili utilizzate in `ParserMusicXML` sono:

- `numeroNota`, che viene utilizzato per numerare le note dello spartito (la prima nota corrisponde al numero 0, le pause non vengono contate come note);
- `f`, una variabile di tipo `ifstream` all'interno della quale viene inserito il contenuto del file MusicXML. `Ifstream` fornisce un'interfaccia per la lettura di dati da file di input;
- `riga`, una stringa che viene utilizzata per estrarre una riga alla volta dal contenuto della variabile `f`;
- `ogg`, una variabile di tipo `XML`. È l'oggetto nel quale di volta in volta vengono inserite le informazioni estratte dal file MusicXML, e che poi viene inserito nel vettore contenitore;
- `trovato`, una variabile booleana che indica di volta in volta se è stata trovata qualche informazione utile: se è stata trovata vuol dire che bisogna riempire il contenitore con un nuovo oggetto XML, altrimenti non serve far niente.

Per estrarre informazioni dal file MusicXML, è necessario sapere com'è strutturato. Quindi verrà spiegato anche come sono descritte le informazioni da estrarre all'interno del file. Ogni battuta dello spartito ha la seguente struttura, che si riferisce ad una battuta con le note di Figura 2.22:

```
<!--=========================================================-->
    <measure number="2" width="174">
      <note default-x="14">
        <pitch>
          <step>C</step>
          <octave>5</octave>
        </pitch>
        <duration>1</duration>
        <voice>1</voice>
        <type>quarter</type>
```

```
      <stem default-y="-50.5">down</stem>
    </note>
    <note default-x="54">
      <pitch>
        <step>A</step>
        <octave>4</octave>
      </pitch>
      <duration>1</duration>
      <voice>1</voice>
      <type>quarter</type>
      <stem default-y="10.5">up</stem>
    </note>
    <note default-x="94">
      <pitch>
        <step>F</step>
        <octave>4</octave>
      </pitch>
      <duration>1</duration>
      <voice>1</voice>
      <type>quarter</type>
      <stem default-y="0">up</stem>
    </note>
    <note default-x="133">
      <pitch>
        <step>G</step>
        <octave>4</octave>
      </pitch>
      <duration>1</duration>
      <voice>1</voice>
      <type>quarter</type>
      <stem default-y="5.5">up</stem>
    </note>
  </measure>
  <!--=========================================================-->
```

Come si può notare ogni riga è formata da tag che danno delle informazioni ben precise.

- `number` del tag `measure` indica il numero della battuta (nell'es. siamo nella battuta numero 2);
- `note` indica una nota in ogni battuta, mentre `default-x` la sua posizione nello spartito (rispetto l'inizio della battuta) se apro il file con Finale;

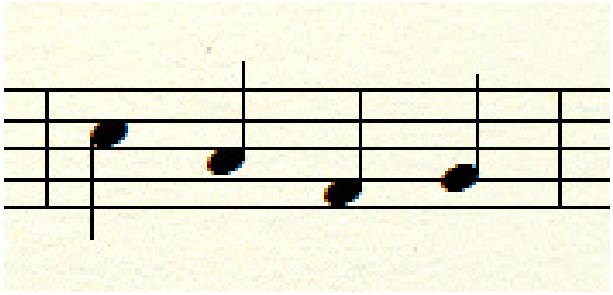


Figura 2.22: Note a cui si riferisce il pezzo di codice MusicXML

- `pitch` indica il tipo di nota a cui ci stiamo riferendo. Sostituendo questo elemento con `<rest/>` al posto della nota avremo una pausa della stessa durata;
- `duration` indica quante parti di semiminima (cioè la nota da quarto) stiamo considerando per la nota in corso. Ad esempio se vale 1 sia la `duration` sia la `division` (è indicata all'inizio del file MusicXML, ed è l'unità di misura che indica in quante parti si suddivide una nota da quarto), allora la nota corrente dura come una semiminima;
- `voice` indica il numero della voce. Se ad esempio è un brano con 2 voci parallele, avremo delle note per la voce 1, e delle note per la voce 2;
- `type` indica come è rappresentata la nota corrente. In questo caso è rappresentata con una nota da quarto (lo si poteva capire anche dalla `duration`);
- `stem` indica graficamente il verso della stanghetta della nota (up o down).

La funzione `ParserMusicXML` effettua un ciclo while su `f`, estraendo ad ogni iterazione una riga del file (come quelle dell'esempio appena visto) usando il comando

```
getline(f, riga);
```

Poi per ogni riga vengono controllati i tag, e vengono estratte le informazioni necessarie. Per verificare il tipo di tag presente nel record corrente, viene estratta una sottostringa della riga utilizzando il metodo `substr`, e poi confrontando il suo contenuto con il tag utilizzando il metodo `compare`. Queste funzioni sono proprie della classe `string`. Esempio per controllare se alla riga corrente c'è il tag `note` verranno eseguite le seguenti operazioni:

```
if (riga.size()>11) nota = riga.substr(6,6);
if (nota.compare("<note ")==0){ ... }
```

Nella prima riga vengono estratti 6 caratteri dal record a partire dalla posizione 6 (l'if serve per evitare errori di puntatori nella funzione `substr`). Nella seconda riga invece

viene confrontato il tag `note` con la sottostringa estratta: se coincidono allora `compare` ritorna 0, altrimenti un numero diverso da 0.

`Rilevazione delle note (<note )`. Quando siamo in presenza del tag `note` vuol dire che le righe successive conterranno informazioni riguardanti una nota musicale. In questo caso viene creato un nuovo oggetto di tipo `XML` (ponendo il puntatore di tale oggetto nella variabile `ogg`), nel quale viene inserito il `numeroNota` corrente come ID utilizzando il metodo pubblico corrispondente. Quando poi successivamente si arriverà ad una riga col tag `</note>`, verrà valutata la variabile `trovato`: se vale false non accade nulla, mentre se vale true significa che nella nota corrente sono state trovate delle informazioni aggiunte con Finale, e quindi l'oggetto `ogg` viene aggiunto in fondo al vettore contenitore.

`Pitch (<step>)`. Per ogni nota che viene aggiunta al vettore contenitore, viene estratta anche l'informazione del pitch. Come si vede facilmente dall'esempio visto in precedenza, il valore del pitch si trova in corrispondenza del tag `<step>`. Questo valore viene poi inserito nel campo `nota` dell'oggetto XML `ogg`.

`Pause (<rest/>)`.Nel file MusicXML le pause vengono trattate come note, la differenza risiede nella presenza del tag `<rest/>` al posto del tag `<step>`, che individua il pitch. In fase di progettazione abbiamo deciso di non considerare le pause come delle note, nonostante siano introdotte dal tag `note`. Per questo motivo quando siamo in presenza del tag `<rest/>` decrementiamo il contatore `numeroNota` (che individua l'ID delle note), in modo tale da annullare gli effetti dell'incremento fatto alla variabile durante la lettura del tag `note`.

`Respiro (<breath-mark)`. Alla presenza del tag del respiro, viene messa a 1 la corrispondente variabile dell'oggetto `XML` corrente. Essendo il respiro un'informazione aggiunta con Finale, la variabile `trovato` viene impostata a true.

`Accento (<accent)`. Stesse operazioni del respiro anche per l'accento: la variabile `accento` dell'oggetto viene impostata a 1, e `trovato` viene messo a true.

`Tenuto (<tenuto)`. Anche per il tenuto, viene messa a 1 la variabile `tenuto` dell'oggetto `XML`, e a true la variabile booleana `trovato`.

`Staccato (<staccato)`. Nel caso di staccato la variabile `staccato` viene impostata a 1, e `trovato` a true.

Dinamiche (<dynamics). Le dinamiche sono molto semplici da individuare; infatti vengono individuate dal tag <dynamics dopo il quale è presente la posizione del simbolo della dinamica nello spartito (a partire dal carattere 31 della riga), mentre alla riga successiva del file MusicXML è presente il tipo di dinamica (pp, p, mp, mf, f, ff). La differenza rispetto ai tipi visti in precedenza, sta nel fatto che un simbolo di dinamica ha effetti per tutte le note successive, a partire dalla posizione del simbolo nello spartito. Questo effetto va avanti finché non cambia la dinamica o fino al termine delle note. Per questo motivo è necessario un codice più elaborato, per far in modo che anche le note successive alla dinamica abbiano un proprio oggetto di tipo XML nel vettore contenitore. In questo modo è possibile applicare gli effetti della dinamica a tutte le note successive. In CaRo viene utilizzato un vettore dinamico `din`, che viene riempito mano a mano che si trova una dinamica nel file. Il vettore ha una struttura particolare. In esso vengono alternate 2 tipi di informazioni:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| X | D | X | D | .. | .. | X | D | -1 |

Dove `X` indica la posizione della dinamica nello spartito, mentre `D` indica il tipo di dinamica. L'ultimo simbolo sarà sempre `-1`, il quale indicherà la fine del vettore. Quindi ogni volta che troviamo il tag `<dynamics`, viene estratta la posizione del simbolo della dinamica e inserita nel vettore `din`. Poi si avanza nel file di una riga, si estrae il tipo di dinamica, e la si inserisce anch'essa nel vettore `din`. Grazie a questo il vettore `din` si riempirà, man mano che si scandisce il file, delle dinamiche presenti nello spartito. Per salvare queste informazioni in un oggetto di tipo `XML` è necessario aggiungere una porzione di codice nella parte che rileva il tag `note`. Se al momento dell'individuazione del tag di una nota il vettore `din` non è vuoto, allora vuol dire che è stata precedentemente inserita qualche dinamica. Per individuare la posizione della dinamica rispetto alla posizione della nota corrente, viene estratta la posizione della nota (presente nella riga del tag `note`) e poi confrontata:

- se la posizione della nota è dopo la posizione della dinamica, allora tale dinamica deve essere applicata alla nota corrente. Per farlo basta prendere le informazioni della dinamica corrente dal vettore `din`;

- se invece la posizione della nota viene prima della posizione della dinamica, allora alla nota viene applicata la dinamica precedente (esistente perché il vettore `din` non era vuoto).

Fatto questo viene impostata a true la variabile `trovato`, in modo tale da permettere l'aggiunta dell'oggetto `XML` nel vettore `contenitore`.

Pedale (<pedal). Questo tag individua che nello spartito è stato inserito un simbolo di pedale. Esistono due tipi di simboli: quello che individua la pressione del tasto pedale (type=start), e quello che individua il rilascio del tasto pedale (type=stop). Uno stop in una nota la esclude dagli effetti del pedale, mentre uno start la include. Per gestire il pedale è stata creata una variabile statoPedale, che indica di volta in volta se siamo in uno stato di pedale on o pedale off. Quindi, al momento dell'individuazione del tag e del tipo di pedale, è possibile impostare una variabile di appoggio con i seguenti valori:

- 1 se siamo in presenza del simbolo start pedale;
- 2 se siamo in presenza del simbolo stop pedale;
- 3 se siamo in presenza di un simbolo start pedale e un simbolo stop pedale nella stessa nota;
- 0 se non abbiamo nessun simbolo che riguarda il pedale.

Nel file MusicXML le righe che riguardano il tag <pedal sono posizionate prima della riga col tag note della nota corrispondente al pedale. Per questo motivo la variabile pedale dell'oggetto XML viene settata, con il valore della variabile di appoggio, nel momento in cui viene individuata la riga della nota.

Tie (<tied). Nel file MusicXML esiste il tag tied per identificare un tie tra due note con lo stesso pitch. In corrispondenza della prima nota è di tipo start, mentre in presenza della seconda nota è di tipo stop. Nel file midi invece, due note legate da un tie vengono viste come un'unica nota di durata pari alla somma delle durate delle due note. Quindi per evitare sincronizzazioni sbagliate tra il numero di note dello spartito, e gli eventi di tipo nota della sequenza MidiShare, viene solo decrementata di uno la variabile numeroNota in presenza di un tie-stop, e se serve viene creato un oggetto XML in presenza della prima nota. Se esiste qualche informazione aggiuntiva sulla seconda nota viene persa, in quanto al momento CaRo non prevede la gestione delle informazioni di entrambe le note. Un miglioramento utile al programma sarebbe quello di raccogliere le informazioni di tutte e due le note e riunirle in un'unica nota, creando se necessario un oggetto XML.

Legato (<slur). Ogni nota può avere più di un tag di tipo <slur in quanto da una nota possono iniziare/terminare più legato contemporaneamente. Nell'oggetto di tipo XML della nota verranno inserite le informazioni di ogni legato: se si tratta di un legato-start o di un legato-stop, un numero intero (contaSlur) che permetterà di distinguere i legato di una stessa nota tra di loro, e un numero intero (number) che identifica il legato nel file MusicXML. Il valore di contaSlur viene azzerato ogni volta che aggiungiamo un nuovo

oggetto di tipo `XML` nel vettore contenitore, mentre il valore di `number` dipende dalla sequenza di legato presenti all'interno dello spartito musicale.

La funzione `ParserMusicXML` termina con una parte di codice che permette la stampa di un messaggio a video con tutte le informazioni raccolte e contenute nel vettore, oppure la stampa di queste informazioni su un file di testo (`FileInfo.txt`), che se non presente viene creato nella cartella del programma CaRo.

### 2.7.4 Classe Parabola

Prima di passare alla spiegazione dell'esecuzione neutra del file MIDI, che utilizza le informazioni raccolte negli oggetti `XML` grazie alla funzione `ParserMusicXML`, è stato necessario creare una nuova classe: `Parabola`. Tale classe permette di creare nuovi tipi di oggetti, che verranno poi utilizzati per gestire le informazioni dei legato durante l'esecuzione neutra del file. La classe e i prototipi delle funzioni sono inserite nel file `Parabola.h`, il cui contenuto è riportato qui di seguito:

```
class Parabola{
   private:
        int number;
        double xVertice;     //coordinate del vertice parabola
        double yVertice;
        double xPuntoSx;     //coordinate del punto a sx del vertice
        double yPuntoSx;
        double xPuntoDx;     //coordinate del punto a dx del vertice
        double yPuntoDx;
        double a;            //3 coefficienti dell'eq. della parabola
        double b;
        double c;
   public:
        //due costruttori
             //inizializza tutte le variabili a 0
        Parabola();
             //calcola i coefficienti a-b-c
        Parabola(int num, double xSx, double xDx,double delta);

        //metodi
        void setNumber(int num);
        int getNumber();
        void setxSx(double xSx);
        double getxSx();
```

```
            void setxDx(double xDx);
            double getxDx();
                 //calcola i coefficienti a-b-c
            void calcolaParabola(double delta);
                 //data x, restituisce la coordinata y della parabola
            double calcolaY(double coordX);
};
```

La variabile privata `number` permette di distinguere le parabole (cioè i diversi legato) tra di loro. Questo perché una nota può essere soggetta a più legato contemporaneamente. Il valore di questa variabile viene estratto dal file MusicXML in quanto anch'esso utilizza un numero intero per distinguere gli slur tra di loro. Gli oggetti di questa classe conterranno le coordinate dei tre punti che andranno a formare la parabola (vertice, punto a sinistra del vertice, e punto a destra del vertice), e le tre costanti `a-b-c` che vanno a formare l'equazione esplicita della parabola.

$$y = ax^2 + bx + c$$

La classe ha due costruttori: uno che inizializza tutte le variabili a 0, e uno che dando in ingresso le coordinate x dei due punti e il delta, calcola ed imposta il valore delle tre costanti che vanno a formare l'equazione della parabola. I metodi pubblici permettono di leggere e settare le variabili private, di calcolare le tre costanti dell'equazione, e di restituire la coordinata y della parabola corrispondente alla coordinata x data in input alla funzione. La classe è implementata all'interno del file `modello.cpp`. L'implementazione è molto semplice, e l'unico punto che è bene sottolineare è come vengono calcolati i valori di `a-b-c`. Per trovare queste costanti basta utilizzare la formula della parabola passante per il vertice ed un punto:

$$y - yVertice = a(x - xVertice)^2$$

Scegliendo come punto quello a sinistra del vertice, e sostituendo le sue coordinate al posto di `x` e `y`, siamo in grado di calcolare il valore di `a`. Poi riscrivendo la formula qui sopra con il valore di `a`, e portandola in forma esplicita, è possibile ricavare anche i valori di `b` e `c`.

### 2.7.5 Esecuzione neutra

Una volta caricato il file MusicXML con Caro, abbiamo la possibilità di eseguire il file MIDI in maniera meccanica o in maniera neutra. Alla pressione del tasto *Play Mecha-*

*nical* vengono attivate le stesse funzioni che vengono attivate per l'esecuzione meccanica del file vista in precedenza. Premendo invece *Play Neutral* viene chiamata la funzione **PlayNeutral** appartenente a `TSpazio`.
La funzione imposta una variabile (IsNeutra) a true per indicare la pressione del tasto *Play Neutral* dal menù, e richiama la funzione `PlayExpressiva` di `TSpazio`. Questo perché le operazioni delle due funzioni sono molto simili, e duplicare gran parte del codice in un'altra funzione risulterebbe inutile. In `PlayExpressiva` viene svuotato il contenuto della sequenza MidiShare `SeqEspressiva`, perché verrà successivamente riempita con gli eventi del nuovo brano MIDI appena caricato. Poi vengono estratte tutte le informazioni presenti nel form Opzioni (ad esclusione della seconda scheda *Interface Parameters*), ed inserite nelle seguenti strutture:

- nelle variabili `Porta`, `Tempo`, `Latenza`, e `Risoluzione`, che sono private della classe `TSpazio`, vengono inseriti i corrispondenti valori della prima scheda;

- in `SoloPiano` viene inserito true se nella quarta scheda del form Opzioni è stata scelta l'opzione *play only on channel 1*, altrimenti viene inserito false;

- `parametriCstaccato` è un array a livello globale di grandezza 4, che contiene i valori dei parametri *Ktempo, Mvelocity, Kvelocity e Klegato* da applicare in presenza di un staccato nel file MusicXML. Tali valori sono dei parametri moltiplicativi, e si trovano nella terza scheda del form Opzioni;

- `parametriCaccento` è un array a livello globale che contiene i valori dei quattro parametri nel caso di un accento;

- `parametriCrespiro` è un array a livello globale che contiene i valori dei quattro parametri nel caso di un respiro;

- `parametriCtenuto` è un array a livello globale che contiene i valori dei quattro parametri nel caso di un tenuto;

- `parametriDeltaLegato` è un array a livello globale che contiene i valori dei quattro parametri in presenza di un legato. A differenza dei precedenti parametri, nel caso di un legato il parametro si riferisce ad una differenza di valori (delta), utilizzato per calcolare la costante moltiplicativa da applicare (vedi sezione 2.7.4);

- `parametriDynamics` è un array a livello globale di grandezza 6 che contiene i valori di *Kvelocity* per ognuna delle sei dinamiche (pp, p, mp, mf, f, ff). In base alla dinamica presente nel file MusicXML, viene utilizzato il parametro moltiplicativo corrispondente;

- `parametriCmelodia` è una variabile double che contiene un parametro moltiplicativo di valore maggiore di 1, che viene utilizzato per aumentare la *Kvelocity* delle note presenti nel channel 0. Questa parametro è stato creato per permettere di aumentare l'intensità delle note del canale 0, che di solito sono le note della melodia del brano, rispetto alle note dell'accompagnamento che di solito si trovano negli altri canali.

Le informazioni inserite nella seconda scheda del form Opzioni invece, vengono estratte ed inserite nelle variabili corrispondenti solo nel caso in cui l'utente abbia premuto il tasto *Play with Expressiveness* dal menù del CaRo (le conseguenze della pressione di questo tasto vengono spiegate meglio più avanti). Le variabili sono le seguenti:

- `modelloKtempo` è un array a livello globale di grandezza 5, che contiene i valori del parametro *Ktempo* per le 5 intenzioni espressive bright, hard, light, soft, e heavy. Tali valori sono presenti nella seconda scheda del form Opzioni;

- `modelloMvelocity` è un array a livello globale che contiene i valori del parametro *Mvelocity* per le cinque intenzioni espressive;

- `modelloKvelocity` è un array a livello globale che contiene i valori del parametro *Kvelocity* per le cinque intenzioni espressive;

- `modelloKlegato` è un array a livello globale che contiene i valori del parametro *Klegato* per le cinque intenzioni espressive;

Se, come in questo caso, è stato premuto *Play Neutral* queste variabili vengono tutte settate a 1 (la motivazione di questa operazione viene spiegata nella sezione 2.8.4).
Fatto questo viene calcolata la costante che permette di trasformare i tick (unità di misura del tempo nel Player) in millisecondi. Questo coefficiente di conversione viene chiamato `tick2ms`, e viene ricavato così:

$$tick2ms = 60000.0/PPQN/Tempo = \frac{60000}{PPQ * Tempo}$$

dove `Tempo` si riferisce al valore di BPM (o QPM) presente nel form Opzioni (vedi sezione 2.5.3 per una spiegazione della formula). Il resto della funzione `PlayExpressiva` è formata da due parti di codice: la prima parte viene eseguita solo se è la prima volta che viene premuto il tasto *Play with Expressiveness* per il brano corrente, la seconda parte invece viene eseguita sempre.

Nella prima parte viene calcolata la media e la varianza delle `velocity` delle note, e la durata massima di ogni nota del brano. Per calcolare la media `KVmedia`, vengono analizzati gli eventi della sequenza MidiShare `SeqNeutra`, che contiene tutti gli eventi del brano MIDI caricato da CaRo, considerando solo le `velocity` degli eventi di tipo

typeNote. Oltre al calcolo della media vengono applicate delle modifiche a tutti gli eventi della sequenza:

- per ogni evento viene modificato il campo date, trasformandolo da tick a millisecondi utilizzando il coefficiente di conversione tick2ms;
- solo per gli eventi di tipo typeNote viene convertito anche il campo durata, utilizzando sempre il coefficiente tick2ms.

Poi viene scandita di nuovo la sequenza MidiShare, e viene sfruttata la media precedentemente calcolata, per calcolare la varianza delle velocity. Inoltre viene creato un array Durata che contiene le durate massime per ogni evento di tipo typeNote del brano. Per ogni evento nota la durata massima è la differenze tra il suo campo date, e il campo date dell'evento successivo di tipo typeNote che abbia lo stesso pitch della nota corrente. Se non esiste una nota successiva con lo stesso pitch, il corrispondente valore di durata massima viene impostato a -1. Inoltre, durante la scansione degli eventi di SeqNeutra per il riempimento dell'array Durata, viene modificato il campo velocity degli eventi di tipo nota scalandolo in base al valore di Out.

Nella seconda parte della funzione PlayExpressiva viene scandita di nuovo la sequenza SeqNeutra partendo dall'inizio. Se l'opzione *play only on channel 1* del form Opzioni è selezionata, manda un evento di tipo typeProgChange al Player. In questo modo viene indicato al Player di eseguire tutte le note col solo timbro del pianoforte. Inoltre vengono inseriti nella sequenza MidiShare SeqEspressiva tutti gli eventi che si trovano prima del primo evento di tipo nota nella sequenza SeqNeutra. Nel caso di eventi di tipo typeProgChange, oltre ad essere inseriti nella sequenza, vengono anche mandati al Player solo se l'opzione *play only on channel 1* non è selezionata.
Prima di terminare, la funzione imposta un timer presente nell'oggetto PerformanceTimer (appartenente al progetto), e lo fa partire. PerformanceTimer è un oggetto della classe TTimer, ed è un'applicazione delle libreria VCL. Esso permette di impostare il valore di un timer, il quale inizia a decrementare non appena viene impostata a true la voce Enabled dell'oggetto. Allo scadere del timer, viene chiamata automaticamente una funzione, dopo la quale il timer riparte dal suo valore originale. In CaRo il timer vale 10ms (campo *Resolution* del form Opzioni), e la funzione che viene chiamata allo scadere del timer è OnPerformanceTimer.

**OnPerformanceTimer** è una funzione che viene chiamata automaticamente ogni 10ms, che ci permette di intercettare tutti gli eventi caricati nel Player e di manipolarli con le informazioni raccolte dal file MusicXML. La funzione infatti permette una sorta di sincronizzazione tra gli eventi di tipo typeNote, e le note con informazioni aggiuntive salvate nel vettore contenitore. Al momento parleremo solo della sincronizzazione; tutte

le altre azioni svolte dall'`OnPerformanceTimer` verranno analizzate più avanti quando si parlerà dell'esecuzione espressiva del brano MIDI.
La parte di codice dell'`OnPerformanceTimer` che viene processata nel caso di esecuzione neutra del file MIDI, viene eseguita solo se il vettore contenitore non è vuoto; cioè solo se esiste almeno una nota del brano che ha delle informazioni aggiunte con Finale. Il vettore viene scandito partendo dal primo oggetto `XML`, e si prosegue con gli oggetti successivi solo se è stato trovato l'evento di tipo nota corrispondente. Per una sincronizzazione corretta vengono confrontati l'`ID` dell'oggetto XML e la variabile NumeroEv, una variabile inizializzata a 0 in `PlayExpressiva`, e incrementata ogni volta che si incontra un evento di tipo nota che sta per essere mandato dal Player. Quando abbiamo una corrispondenza tra un evento `typeNote` e un oggetto `XML`, vengono controllate tutte le informazioni contenute nell'oggetto.

- Se l'oggetto indica che la nota corrispondente ha uno staccato nello spartito modificato con Finale, allora viene modificata la costante moltiplicativa che viene influenzata dallo staccato, con un peso pari alla costante indicata nella terza scheda del form Opzioni. In questo caso viene influenzata la variabile `klegato`:

  ```
  Klegato = Klegato*parametriCstaccato[3];
  ```

  `Klegato` andrà ad incidere sulla durata della nota;

- Nel caso di un accento viene modificata la `Kvelocity`, che và ad influenzare la velocità di pressione del tasto (campo `velocity` dell'evento di tipo `typeNote`)

  ```
  Kvelocity = Kvelocity*parametriCaccento[2];
  ```

- In presenza di un respiro vengono modificate le seguenti costanti moltiplicative

  ```
  Ktempo = Ktempo*parametriCrespiro[0];
  Klegato =Klegato*parametriCrespiro[3];
  ```

  che vanno ad agire entrambe sulla durata dell'evento nota, e il `Ktempo` anche sull'istante di inizio della nota successiva (tramite `OnsetEspressivo`), il quale viene calcolato in base alla differenza tra i date originali delle due note moltiplicata per il valore di `Ktempo`:

  ```
  OnsetEspressivo=OnsetEsprPrec+Ktempo*(OnsetNeutro-OnsetNeutroPrec);
  ```

- Nel caso di tenuto viene modificato il valore del `Klegato`, e quindi la durata della nota

```
Klegato = Klegato * parametriCtenuto[3];
```

- In presenza di una delle cinque `dinamiche`, viene modificata la `Kvelocity`. Infatti una dinamica del tipo ff necessità una pressione del tasto (`velocity`) molto più potente rispetto ad una dinamica del tipo pp.

- Nel caso di un `pedale` si deve controllare il tipo di pedale. In caso di pedaleON si deve creare un nuovo evento di tipo `typeCtrlChange` il quale, una volta mandato al Player, permette di applicare gli effetti del pedale del pianoforte alle note successive. Tale evento è composto da due campi: il campo 0 indica il tipo di controllo, nel nostro caso viene impostato a 64 per indicare il pedale (Damper Pedal), il campo 1 invece indica il valore di pressione del pedale (da 0 a 63 pedale off, da 64 a 127 pedale on). Inoltre l'evento viene aggiunto anche alla sequenza MidiShare `SeqEspressiva`, in modo che risulti presente sia nel Player e sia nella sequenza di tutti gli eventi del brano. Questo garantisce la presenza di tale evento anche nel caso di salvataggio del brano MIDI durante l'esecuzione di *Play with Expressiveness*. In caso di pedaleOFF invece, vengono eseguite le stesse cose, con la differenza che vengono impostati in maniera diverse i campi dell'evento `typeCtrlChange` che viene creato ("MIDI Messages", n.d.).

| | |
|---|---|
| Pedale on: | MidiSetField(evPedale,0,64);<br>MidiSetField(evPedale,1,127); |
| Pedale off: | MidiSetField(evPedale,0,64);<br>MidiSetField(evPedale,1,0); |

- In presenza di `legato`, le operazioni da fare sono molto più complesse. Infatti, dobbiamo considerare nel loro insieme tutte le note che stanno sotto ad uno slur, e costruire una parabola. La parabola è del tipo

$$y = ax^2 + bx + c \text{ con } a < 0$$

cioè una parabola rivolta verso il basso, e passante per due punti. Le coordinate x dei due punti vengono trovate andando a guardare il campo `date` delle due note in cui inizia e finisce il legato. La coordinata y invece, che è uguale per entrambi i punti, viene calcolata in base al parametro *DeltaLegato* presente nel form Opzioni:

$$\text{coordinata } y = 1 - DeltaLegato$$

Il vertice della parabola ha per coordinata y il valore 1, mentre per coordinata x il valore che sta esattamente alla metà delle coordinate x dei due punti. Grazie al vertice e ai due punti è possibile calcolare l'equazione della parabola, e utilizzarla per

trovare le costanti moltiplicative da applicare alle note del brano che sono coinvolte nello slur. Ci viene quindi in aiuto la classe `Parabola`, spiegata in precedenza nel sezione 2.7.4, che attraverso i suoi metodi permette l'inserimento delle coordinate dei due punti e del delta, per ottenere le costanti che andranno a formare l'equazione della parabola. Infatti la costante moltiplicativa da applicare ad una nota con campo `date` pari a *X1*, non è altro che la coordinata y trovata sostituendo il valore *X1* nell'equazione della parabola. Nella classe `Parabola` il metodo che implementa questa funzione è `calcolaY`. Nel nostro caso il legato và ad influire sulle costanti `Ktempo` e `Kvelocity`, quindi per ogni slur è necessario creare due parabole, ognuna per le due costanti moltiplicative. Attraverso i valori dei due parametri *DeltaLegato* del form Opzioni, possiamo trovare l'equazione della parabola relativa al `Ktempo` e l'equazione della parabola relativa alla `Kvelocity`. Dal punto di vista implementativo, le due parabole vengono create solo se l'oggetto `XML` indica che la nota corrispondente nel Player ha un legato-start. Dalla nota corrente viene estratto il `date`, che sarà la coordinata x del punto a sinistra del vertice della parabola, e dall'oggetto `XML` viene estratto il `number` del legato. Per poter costruire la parabola sono ora necessarie le informazioni della nota che si trova a fine dello slur. Per trovarla viene fatto un ciclo secondario che avanza nel vettore `contenitore`, e viene trovato l'oggetto contenente lo stesso `number` del legato e il parametro legato-stop. Allo stesso modo viene preso l'evento nota corrente, e vengono analizzati gli eventi successivi finché si trova l'evento `typeNote` che ha lo stesso ID dell'oggetto `XML` con il legato-stop appena trovato. In questo modo abbiamo la possibilità di estrarre il `date` dell'ultima nota sotto al legato (cioè la coordinata x del punto a destra del vertice della parabola), e di inserirlo nei due oggetti di tipo `Parabola`. Avendo i `date` delle due note agli estremi del legato, e il *delta* presente nel form Opzioni, è possibile calcolare per ogni parabola le tre costanti `a-b-c` che vanno a formare l'equazione, utilizzando la funzione `calcolaParabola` della classe. In questo modo i due oggetti di tipo `Parabola` sono ora utilizzabili per trovare le costanti moltiplicative da applicare agli eventi di tipo nota che sono influenzati dal legato.

Con il programma Finale è possibile aggiungere più di un legato, e CaRo gestisce correttamente anche il caso di più legato sovrapposti in una stessa nota. Infatti da una nota possono iniziare e terminare più di uno slur, e allo stesso tempo tale nota può essere influenzata da altri legato iniziati in note precedenti. In CaRo questa situazione viene gestita creando due vettori:

`listaParaboleTempo`

`listaParaboleVelocity`

Questi vettori dinamici conterranno in ogni istante i legato, e quindi le parabole che influenzano l'evento nota corrente. Quindi appena abbiamo un oggetto `XML` che indica un legato-start vengono creati i due oggetti di tipo `Parabola`, vengono riempiti con tutte le informazioni necessarie per la creazione dell'equazione delle parabola (come visto in precedenza), e vengono aggiunti ai due vettori. Quando poi si arriverà ad un oggetto `XML` contenente un legato-stop, vengono eliminati i due oggetti `Parabola` corrispondenti nei due vettori grazie alla corrispondenza tra il `number` dello slur nel file MusicXML e il `number` delle due parabole nei vettori.

Avendo a disposizione i due vettori per le parabola, ora risulta molto più semplice l'applicazione degli effetti dei legato. Durante il procedimento di sincronizzazione tra gli oggetti `XML` e gli eventi che vengono mandati al Player, per ogni evento di tipo nota vengono controllati i due vettori `listaParaboleTempo` e `listaParaboleVelocity`:

* se sono vuoti significa che la nota corrente non è soggetta a nessun legato;
* se non sono vuoti allora esistono uno o più legati che influenzano tale nota. Questi slur, che vanno a riempire i due vettori, possono sia iniziare o terminare nella nota corrente, ma possono anche essere dei legato che sono iniziati in note precedenti e che devono ancora terminare (vedi Figura 2.23).

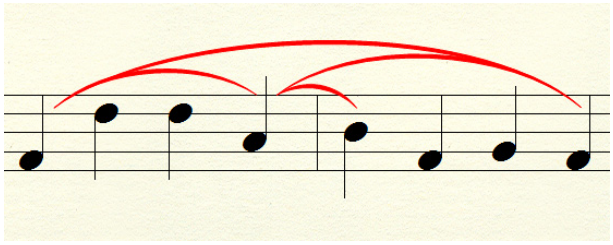


Figura 2.23: Note con più informazioni di legato

Per ogni legato che influenza la nota corrente vengono aggiornate le due costanti moltiplicative `Ktempo` e `Kvelocity` tramite la funzione `calcolaY`.

```
for(int i=0;i<listaParaboleTempo.size();i++){
   Ktempo = Ktempo *
            listaParaboleTempo[i].calcolaY(EvEspressivo->date);
   Kvelocity = Kvelocity *
               listaParaboleVelocity[i].calcolaY(EvEspressivo->date);
}
```

Quindi il legato và ad influire sulla `durata` e sull'istante di inizio delle note (OnsetEspressivo) grazie alla costante `Ktempo`, e sul campo `velocity` degli eventi di tipo nota grazie alla costante `Kvelocity`.

Con la sincronizzazione tra il vettore con gli oggetti di tipo `XML` e gli eventi MidiShare, abbiamo così potuto aggiornare il valore delle variabili moltiplicative che verranno utilizzate per modificare i corrispondenti parametri dell'evento da inviare al Player. Il come vengano effettuata queste modifiche verrà visto nella parte riguardante l'esecuzione espressiva del file MIDI (vedi sezione 2.8), in quanto queste costanti saranno soggette a ulteriori modifiche in base all'espressività che l'utente vuole aggiungere al brano.

## 2.8 Esecuzione espressiva di un file MIDI

Si dice esecuzione espressiva di un file MIDI quando l'utente interagisce con il programma CaRo utilizzando l'interfaccia. L'interfaccia può essere utilizzata sempre, indipendentemente dai file caricati nel programma, cioè può essere utilizzata

- caricando solo il file MIDI;
- caricando il file MIDI e il file MusicXML corrispondente.

In entrambi i casi l'atto che rende espressiva l'esecuzione del file MIDI, è l'interazione che si crea tra l'utente e CaRo; infatti durante l'esecuzione del brano, l'utente potrà muovere il mouse all'interno dell'interfaccia. Durante il movimento, il variare della posizione del mouse agirà in real-time sull'esecuzione del brano, modificandone le intenzioni espressive in base alla posizione. In questo contesto risulta molto importante la seconda scheda del form Opzioni (vedi Figura 2.7).

### 2.8.1 Interfaccia di Caro 2.0

L'interfaccia del CaRo, che per comodità viene riportata in Figura 2.24, fornisce all'utente cinque intenzioni espressive (nel codice sono detti aggettivi), le quali hanno delle coordinate ben precise nello spazio 2D:

| aggettivi | coordinate |
|---|---|
| Bright-Brillante | (0.945,0.52) |
| Hard-Duro | (0.35,0.91) |
| Light-Leggero | (0.82,0.195) |
| Soft-Morbido | (0.4,0.065) |
| Heavy-Pesante | (0.09,0.74) |

Queste coordinate, per essere utilizzate, vengono inserite in due vettori:

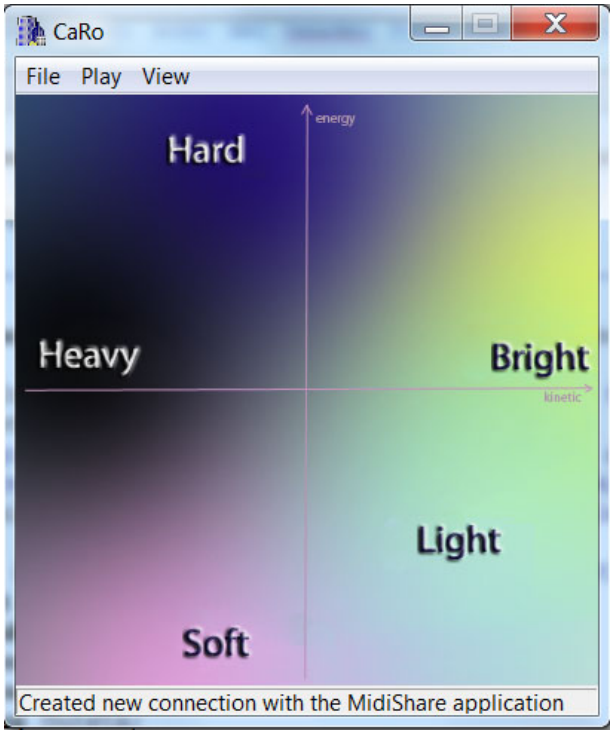



Figura 2.24: Interfaccia utente di CaRo 2.0

```
const float braniX[NumeroAggettivi]={0.945,0.35,0.82,0.4,0.09};
const float braniY[NumeroAggettivi]={0.52,0.91,0.195,0.065,0.74};
```

Ogni aggettivo ha delle costanti moltiplicative associate, che permettono di modificare i campi degli eventi di tipo nota, rendendola così più adatta all'intenzione espressiva. Muovendo il mouse sopra ad un aggettivo, l'utente può applicare al brano quella particolare intenzione espressiva, e può passare da un aggettivo ad un altro in maniera graduale (smooth morphing) producendo nel brano dei cambiamenti morbidi tra diverse intenzioni espressive. Come spiegato nella tesi sviluppata dal collega Davide Ganeo [36], a seguito di molte analisi alcuni parametri sono risultati importanti e molto più incisivi di altri per la riproduzione delle intenzioni espressive. Essi sono:

- il Ktempo, che và ad agire sulla `durata` dell'evento nota, e anche sull'istante di inizio della nota successiva;
- la Mvelocity, una costante moltiplicativa che viene applicata allo scarto tra la `velocity` dell'evento nota e le velocity media (nel codice indicata con `KVMediaOut`, e inizializzata a 64);
- la Kvelocity, che và ad agire sul campo `velocity` dell'evento nota, cioè sull'intensità di esecuzione sonora della nota;
- il Klegato, che và ad agire sulla `durata` dell'evento nota.

Come detto sopra ogni intenzione espressiva ha un valore per questi quattro parametri. Possono capitare due situazioni:

1. se nell'interfaccia il mouse giace perfettamente sopra un aggettivo, vengono usati direttamente i quattro parametri corrispondenti all'aggettivo per applicarli ai campi dell'evento di tipo nota corrente;

2. se invece il mouse si trova in un punto casuale dello spazio 2D, diverso dalle ciinque coordinate degli aggettivi, vengono pesate le distanze tra il puntatore del mouse ed ogni intenzione espressiva. Poi viene calcolata la media ponderata di ogni parametro sui cinque aggettivi, pesando in base al reciproco delle distanze. Qui di seguito è mostrato il codice relativo a questi calcoli.

```
for (int i=0; i<NumeroAggettivi; i++){
   Distanze[i]=(asseX-braniX[i]*Wlarghezza)^2+(asseY-braniY[i]*Waltezza)^2;
   Pesi[i]=1.0/Distanze[i];             //meno distante è, e più peso c'è
   SommaPesi=SommaPesi+Pesi[i];
}
for (int i=0; i<NumeroAggettivi; i++){
   Ktempo=Ktempo+modelloKtempo[i]*Pesi[i]/SommaPesi;
   Mvelocity=Mvelocity+modelloMvelocity[i]*Pesi[i]/SommaPesi;
   Kvelocity=Kvelocity+modelloKvelocity[i]*Pesi[i]/SommaPesi;
   Klegato=Klegato+modelloKlegato[i]*Pesi[i]/SommaPesi;
}
```

   Dove gli array che hanno la parola `modello` come prefisso, contengono i valori dei parametri delle cinque intenzioni espressive, che vengono in precedenza estratti dalla seconda scheda del form Opzioni.

### 2.8.2 Esecuzione: la funzione *OnPerformanceTimer*

Verrà visto, dal punto di vista implementativo, come vengono gestite nel complesso le modifiche agli eventi di tipo nota provocate dalle informazioni estratte dal file MusicXML, e le modifiche provocate dall'interazione tra utente e interfaccia del CaRo. Tutto questo è presente nella funzione **OnPerformanceTimer**. Come già detto questo metodo viene chiamato ciclicamente ogniqualvolta scade il timer, che in CaRo viene settato a 10ms. Il timer viene fatto partire per la prima volta dalla funzione `PlayExpressiva`, la quale viene chiamata appena viene premuto il tasto *Play with Expressiveness* dal menù *Play*.

Siccome la funzione viene chiamata ogni 10ms, e siccome le note del brano hanno delle durate che in generale non sono sincronizzabili con intervalli di 10ms, all'interno

dell'`OnPerformanceTimer` sono necessari dei controlli per evitare che una nota del brano venga manipolata più di una volta. Per farlo vengono utilizzate le seguenti variabili:

- `InizioCiclo`. Viene inizializzata all'inizio della funzione tramite `MidiGetTime()`, e quindi contiene l'istante temporale di inizio dell'`OnPerformanceTimer`.

- `StopTime`. All'inizio della funzione `StopTime` contiene l'istante temporale di inizio del prossimo evento di tipo nota da manipolare. Quando l'`OnPerformanceTimer` finisce di analizzare una nota, `StopTime` viene impostato col valore di inizio dell'evento nota successivo.

L'intero corpo dell'`OnPerformanceTimer` viene eseguito solo se la funzione viene chiamata dopo l'istante temporale di inizio del prossimo evento nota da manipolare. Per far questo basta inserire questo controllo:

```
if (InizioCiclo>StopTime) { ...
```

Passato questo controllo, la funzione controlla la posizione del mouse e calcola i valori dei quattro parametri rispetto alle cinque intenzioni espressive dell'interfaccia di CaRo. Poi per evitare che una nota venga manipolata più di una volta, viene fatto un ulteriore controllo sul tempo

```
while (MidiGetTime()>StopTime) { ...
```

Prima di entrare nel while, StopTime indica il tempo di inizio (già oltrepassato) dell'evento nota che si sta manipolando. All'interno del while invece `StopTime` viene modificato con il tempo di inizio dell'evento nota successivo. In questo modo si entrerà nel while solo una volta, cioè solo quando la nota corrente verrà manipolata per la prima volta; successivamente `StopTime` avrà un valore sicuramente superiore al `MidiGetTime()`.
All'interno del while viene fatto un ulteriore controllo, necessario per evitare che il tempo di esecuzione di `OnPerformanceTimer` sia superiore al tempo necessario al Timer per scadere. Tale situazione viene indicata all'utente con un messaggio di saturazione nella *StatusBar*, che invita ad aumentare la *risoluzione* presente nel form Opzioni. Il codice che effettua questo controllo è il seguente:

```
if (MidiGetTime()-InizioCiclo>0.9*Risoluzione) {
      Saturazione=true;
      StatusBar1->SimpleText="Saturation! Increase resolution";
 }
else Saturazione=false;
```

Tale soluzione risulta ottima in quanto aumentando la saturazione aumenta anche la durata del timer, e quindi la funzione ha più tempo per completare la sua esecuzione.

Proseguendo con il codice di `OnPerformanceTimer`, all'interno del ciclo while vengono eseguite tutte le altre operazioni rimanenti:

- la sincronizzazione tra evento nota e le informazioni aggiuntive di tale nota contenute nell'array contenitore;

- individuazione del tipo di informazioni aggiuntive, e applicazione di queste informazioni alle costanti moltiplicative `Ktempo - Mvelocity - Kvelocity - Klegato` (vedi sezione 2.7);

- aumento dell'intensità della melodia per le note contenute nel channel 0. Per farlo basta moltiplicare il `Ktempo` di queste note per il fattore `parametriCmelodia`, che è contenuto nel form Opzioni e ha un valore maggiore di 1;

- applicazione delle costanti moltiplicative ai campi dell'evento nota corrente;

- avanzare nella sequenza di eventi MidiShare dall'evento nota corrente, al successivo evento di tipo `typeNote` (gli eventi intermedi vengono mandati al Player ed inserite nella sequenza `SeqEspressiva`). Ci sono due casi:

  1. se non esiste un evento nota successivo a quello corrente, la *StatusBar* viene settata con *Stop*, e viene mandato a video il messaggio di richiesta di salvataggio dell'esecuzione espressiva del brano appena conclusa (vedi Figura 2.25). Si tratta di una MessageBox nella quale l'utente deve decidere se salvare il brano, premendo il tasto *Si*, o meno (tasto *No*). Alla pressione del tasto *Si* viene richiamato il metodo `MenuFileSaveClick`, creato appositamente per compiere questa funzione (descritto nella sezione 2.9);
  2. se invece esiste un evento di tipo nota successivo, allora vengono aggiornate le variabili `OnsetNeutro - OnsetEspressivo - StopTime`:

     ```
     OnsetNeutro=EvNeutro->date;
     OnsetEspressivo=OnsetEsprPrec+Ktempo*(OnsetNeutro-OnsetNeutroPrec);
     StopTime=IstanteInizio+OnsetEspressivo;
     ```

     dove `EvNeutro->date` è il `date` del nuovo evento nota prima delle modifiche espressive, e `IstanteInizio` indica l'istante temporale di pressione del tasto *Play with Expressiveness* dal menù *Play*. Dal momento che `OnsetEspressivo` è rispetto all'istante temporale di esecuzione della funzione `PlayExpressiva`, `StopTime` viene così impostato con il valore di inizio dell'evento nota successivo a quello appena analizzato da `OnPerformanceTimer`.

### 2.8.3 Applicazione delle modifiche alle note

Come già accennato nella sezione precedente, verrà illustrato ora come vengono effettivamente applicate le costanti moltiplicative all'evento nota corrente. Come già spiegato tali costanti vanno ad agire su certi campi dell'evento `typeNote`.

La `durata` dell'evento nota corrente (campo numero 2) viene moltiplicata per `Klegato` e `Ktempo`, e poi viene preso il minimo tra il risultato di questa moltiplicazione e il valore massimo della durata per la nota corrente che è stato calcolato ed inserito nel vettore `DurataMax`.

La `velocity` dell'evento nota (campo numero 1) viene aggiornata nel seguente modo: viene modificato lo scarto tra la `velocity` originale e `KVMediaOut` (impostato a 64) moltiplicandolo per `Mvelocity`, e viene modificato il valore medio della velocità moltiplicandolo per `Kvelocity`.

```
(MidiGetField(EvEspressivo,1)-KVMediaOut)*Mvelocity+Kvelocity*KVMediaOut
```

Poi viene preso il minimo tra il valore appena ottenuto e il valore massimo della velocità (`KVMaxOut`), e il valore massimo tra il risultato di quest'ultima operazione e il valore minimo della velocità (`KVMinOut`).

Il valore `date` dell'evento nota (indica l'istante temporale della nota) viene impostato in base all'istante di pressione del tasto *Play with Expressiveness* (`IstanteInizio`), alla `latenza` (di default vale 100ms, ed è estratta dal form Opzioni), e in base al valore di `OnsetEspressivo` che viene calcolato in precedenza.

EvEspressivo->date=IstanteInizio+Latenza+OnsetEspressivo;

Se il `date` risultante della nota rimane inferiore al tempo corrente (`MidiGetTime()`), la nota non verrà mai processata dal Player. Per risolvere questo problema basta agire sulla latenza, chiedendo all'utente di aumentarla tramite il form Opzioni.

```
if (EvEspressivo->date < MidiGetTime())
{
      Saturazione=true;
      StatusBar1->SimpleText="Saturation! Increase latency";
}
```

In caso di saturazione l'evento nota non viene mandato ne al Player, ne inserito nella sequenza MidiShare `SeqEspressiva`, ma il valore della variabile `Saturazione` viene impostata di nuovo a false e si passa all'evento successivo. Se invece non c'è nessuna saturazione:

- viene creato un nuovo evento `tmpKeyOnOff` di tipo `typeKeyOn` nel quale vengono inseriti i valori espressivi appena trovati (`date, velocity`), e i valori del pitch, della porta e del channel dell'evento nota originale. Poi tale evento viene inviato al Player

  ```
  MidiSend(ourRefNum,MidiCopyEv(tmpKeyOnOff));
  ```

- viene creato un nuovo evento che identifica la fine della nota appena creata (cioè un evento `typeKeyOff`). Per farlo viene modificato l'evento `typeKeyOn` precedente, visto che è già stato utilizzato, settando a 0 la `velocity`, e il `date` viene aumentato della durata che dovrebbe avere la nota espressiva

  ```
  tmpKeyOnOff->date=tmpKeyOnOff->date+MidiGetField(EvEspressivo,2);
  ```

Come per la funzione `ParserMusicXML`, durante tutte le azioni svolte dalla funzione `OnPerformanceTimer` vengono messe in delle variabili delle informazioni che aiutano lo sviluppatore a capire se le operazioni svolte dalla funzione sono corrette o meno. Queste variabili sono delle stringhe globali (`info` per `ParserMusicXML`, e `info2` per `OnPerformanceTimer`) il cui contenuto viene poi stampato in file di testo contenuti nella cartella del programma (`FileInfo.txt` per `ParserMusicXML` , e `FileInfo2.txt` per `OnPerformanceTimer`).

### 2.8.4 Attivazione e disattivazione dell'interfaccia

Una cosa molto importante che non è stata spiegata finora è come attivare o disattivare l'interfaccia del CaRo. Infatti premendo il tasto *Play Neutral* l'utente può effettuare un'esecuzione neutra del brano se l'interfaccia è disattivata, mentre premendo il tasto *Play with Expressiveness* effettua un'esecuzione espressiva del brano se l'interfaccia è attivata. Ovviamente se l'utente non carica il file MusicXML, l'esecuzione neutra non può esistere, ma solo quella meccanica (premendo *Play Mechanical*) e quella espressiva (se l'interfaccia è attiva e premendo *Play with Expressiveness*).
Di default all'avvio dell'applicazione CaRo, l'interfaccia utente è disattivata; infatti andando sulla seconda scheda (*Parameters*) del form Opzioni, si può notare che tutti i parametri del form sono impostati a 1. I settaggi di questi valori vengono fatti nel costruttore della classe `TFormOpzioni`, implementato nel file `Options.cpp`. Questi parametri sono le costanti moltiplicative che verranno poi applicate ai campi degli eventi di tipo nota: se valgono tutte 1, non influiranno in nessun modo sugli eventi. Per attivare l'interfaccia ci sono due possibilità:

- utilizzare il pulsante *LOAD Parameters from File* per caricare dei parametri precedentemente salvati in un file di testo;

- oppure modificare a mano le singole caselle di testo per modificare le costanti moltiplicative.

Questa operazione può essere fatta prima o dopo del caricamento dei file in CaRo, e anche in mezzo a più esecuzioni espressive e non dello stesso brano.
Ecco perché nel caso di pressione del tasto *Play Neutral*, tutte le variabili relative alle informazioni presenti nella seconda scheda del form Opzioni, vengono settate a 1.
Per quanto riguarda la terza scheda (*MusicXML Parameters*) del form Opzioni invece, i parametri di default sono estratti dal file `MusicXMLparameters3.txt`, un file di testo dentro la cartella *Midi1-MusicXML-Parameters* del progetto. Anche in questo caso il settaggio di questi valori di default viene fatto nel costruttore della classe `TFormOpzioni`. Il settaggio è una semplice scansione del file di testo che estrae ad ogni riga le informazioni corrispondenti ad una casella di testo della scheda. Il contenuto del file `MusicXMLparameters3.txt` è riportato qui sotto:

```
1 Cstaccato0
1 Cstaccato1
1 Cstaccato2
0,7 Cstaccato3
1 Caccento0
1 Caccento1
1,2 Caccento2
1 Caccento3
1,2 Crespiro0
1 Crespiro1
1 Crespiro2
0,8 Crespiro3
1 Ctenuto0
1 Ctenuto1
1 Ctenuto2
1,2 Ctenuto3
-0,1 DeltaLegato0
0 DeltaLegato1
0,2 DeltaLegato2
0 DeltaLegato3
0,7 Dynamics0 pp - pianissimo
0,8 Dynamics1  p - piano
0,9 Dynamics2 mp - mezzo-piano
1 Dynamics3 mf - mezzo-forte
```

```
1,1 Dynamics4  f - forte
1,2 Dynamics5 ff - fortissimo
1,6 Cmelodia
```

## 2.9 Salvare il brano eseguito

La funzione **MenuFileSaveClick** è stata creata appositamente nel file `modello.cpp` per gestire i casi in cui l'utente decide di salvare l'esecuzione del brano premendo il tasto *Save* dal menù, oppure quando preme il tasto *Stop*, oppure quando termina il brano musicale. Questo solo nel caso in cui l'utente esegue il brano con *Play with Expressiveness*. Il messaggio a video (MessageBox) che chiede all'utente se salvare o no il brano in formato MIDI, si presenta quando termina la canzone, oppure quando l'utente la arresta premendo il tasto *Stop* dal menù. Se l'utente decide di salvare il brano (vedi Figura 2.25), si apre una

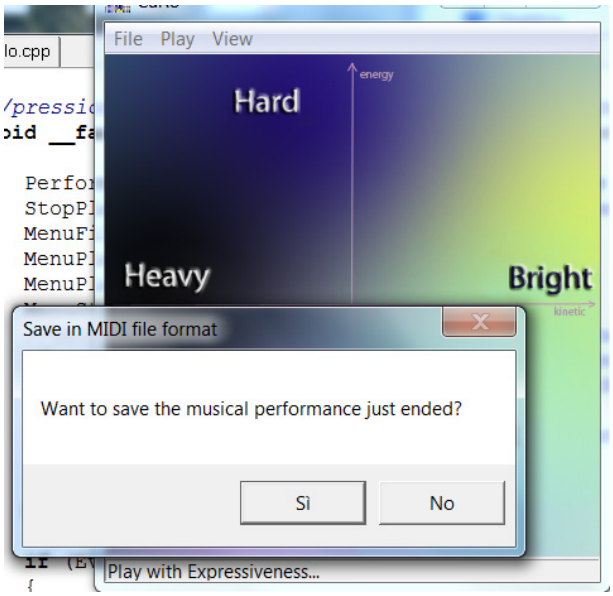



Figura 2.25: Richiesta all'utente di salvataggio del brano appena eseguito

finestra Windows (strumento `SaveDialog` del C++Builder) che permette di selezionare il percorso di salvataggio e di scegliere il nome del file. Nel codice questo strumento è stato chiamato `FileSaveDialog`:

```
if(FileSaveDialog->Execute())
{
      NomeFile=FileSaveDialog->FileName.c_str();
      MidiFileSave( NomeFile, SeqEspressiva, &InfoRecEspr);
}
```

`Execute()` apre la finestra di dialogo di Windows, e ritorna true quando l'utente clicca su *OK* per salvare il file. `MidiFileSave` invece è un metodo del Player che permette di

salvare una sequenza. `InfoRecEspr` è un oggetto di tipo `MidiFileInfos` che deve essere riempito con le informazioni sul tipo di file da salvare (MIDI) e sul tempo (tick per nota da quarto) del brano.

## 2.10 Miglioramenti e sviluppi futuri

Rispetto alla versione 1.4 di CaRo, questo lavoro di tesi ha permesso di ampliare le funzionalità dell'applicazione portandola alla release 2.0. Come visto e descritto in questo capitolo, i miglioramenti riguardano la gestione di nuovi file (`MusicXML`) per il reperimento di informazioni utili ad eseguire il brano in maniera più naturale, come se lo facesse un interprete umano (*Play Neutral*). Questo ha permesso all'utente, che muove il mouse all'interno dell'interfaccia, di agire in modo espressivo su due tipi di brani:

- il primo è il brano meccanico stampato nello spartito;

- il secondo è il brano neutro, come se fosse già stato eseguito da un musicista.

Il risultato dell'influenza di aggettivi di espressività (soft, hard ecc.), produce notevoli differenze all'ascolto a seconda della tipologia di brano su cui andiamo ad agire.

Attualmente per utilizzare l'applicazione CaRo, l'utente è costretto a caricare un file MIDI, e poi a scelta può caricare o meno il file MusicXML corrispondente. Un possibile miglioramento potrebbe essere quello di togliere questo obbligo, e rendere il caricamento del file MIDI facoltativo. Quindi l'utente per utilizzare l'applicazione può effettuare uno dei seguenti passaggi:

- caricare solo il file MIDI. In questo caso il file MIDI potrebbe contenere, al posto di una performance meccanica, una performance neutra registrata da un pianista umano oppure ottenuta da un sistema automatico (ad esempio attraverso il DM - vedi sezione 1.2.4). Quindi, si possono ottenere solo due tipi performance, a seconda del file MIDI caricato: meccanica ed espressiva se il midi contiene il brano con le stesse caratteristiche del brano corrispondente allo spartito musicale, oppure neutra ed espressiva, se il MIDI è una registrazione della performance da un umano o da un sistema automatico;

- caricare solo il file MusicXML. In questo caso le note e la melodia del brano, che prima venivano prese dal file MIDI, vengono prese direttamente dal file MusicXML; infatti, tale file contiene tutte le informazioni necessarie per poter eseguire il brano, quindi basterebbe migliorare e potenziare la funzione `ParserMusicXML` in modo che possa raccogliere anche le informazioni sulle note.

Con l'aggiunta della seconda opzione, è bene tenere attiva anche l'opzione uno, in caso l'utente abbia a disposizione solo il file MIDI, e volesse utilizzare l'interfaccia del CaRo per espressivizzare tale brano.

Ulteriore miglioramento potrebbe essere quello di dare all'utente ulteriori possibilità e nuove opzioni per caricare i file musicali nell'applicazione; infatti, non sempre l'utente ha a disposizione il file MIDI o il file MusicXML. L'utente potrebbe disporre solo della copia cartacea dello spartito, e quindi per poter utilizzare CaRo dovrebbe in qualche modo recuperare il file MIDI corrispondente, o nel peggior delle ipotesi crearselo autonomamente (usando Finale ad esempio). L'idea sarebbe quella di portare lo spartito cartaceo nel computer utilizzando lo scanner, e poi dotare CaRo degli strumenti necessari a interfacciarsi con i programmi (gli OCR musicali) che riconoscono lo spartito, le note e i vari simboli che compongono il pezzo musicale contenuto nel foglio scanerizzato.
Una volta raccolte tutte le informazioni dall'immagine scannerizzata dello spartito, possono essere direttamente utilizzate per i tre tipi di esecuzione offerti da CaRo, oppure si può implementare una routine che crei un file MusicXML che contenga queste informazioni, così si può sfruttare la parte di esecuzione che è già attualmente gestita dal CaRo.

Un altro miglioramento molto importante da implementare, riguarda l'interfaccia grafica. Attualmente l'interfaccia è molto semplice, e permette una basilare interazione utente-applicazione per cambiare l'espressività musicale scegliendo tra gli aggettivi proposti dall'immagine di sfondo. L'idea sarebbe quella di cambiare questa interfaccia "statica", e renderla personalizzabile dall'utente:

- possibilità di cambiare sfondo;
- possibilità di scegliere gli aggettivi da usare scegliendo da un insieme, anche più ampio, di aggettivi espressivi;
- possibilità di utilizzare simboli o immagini al posto delle etichette degli aggettivi

Inoltre L'opzione del cambiamento di interfaccia nasce dalla richiesta di poter passare ad un ambiente di sviluppo più intuitivo, e a un compilatore più aggiornato, che usi librerie grafiche multipiattaforma. Il programma CaRo infatti utilizza come librerie grafiche VCL (Visual Component Library), che sono state sviluppate dalla Borland, ed integrate in C++Builder, e che possono essere utilizzate solo su piattaforma Windows. Oltre a questo, col cambiamento delle librerie grafiche si potrebbe passare ad un ambiente di sviluppo più odierno, più semplice e veloce da utilizzare.

Altra idea molto interessante sarebbe quella di rendere l'applicazione CaRo un applet online scaricabile da internet. Questo aprirebbe nuove frontiere e andrebbe verso direzione dell'obiettivo di tale applicazione, cioè rendere fruibile ad utenti non musicalmente esperti degli strumenti utili a modificare e creare musica a loro piacimento.

# Capitolo 3

# Valutazione delle performance musicali

La valutazione delle performance musicali è comune a molte pratiche musicali educative, però la ricerca che chiarisce l'insieme di fattori che influenzano tali valutazioni è ancora relativamente scarsa.
Verrà ora visto un punto della letteratura corrente sui principali problemi che vanno ad influenzare le valutazioni delle performance, e verrà proposto un modello di valutazione delle performance musicali [59] che identifica e che ci permette di discutere di alcuni dei principali elementi che possono affliggere la valutazione dei giudici in competizioni, audizioni, recite e esami [60].

## 3.1 Il concetto di valutazione

La valutazione di performance musicali è un processo attraverso il quale si cerca di bilanciare e sintetizzare le varie qualità di una performance eseguita da un individuo, con lo scopo di fornire un giudizio (esempio una descrizione qualitativa, un punteggio per una graduatoria, ecc.). La valutazione delle performance da parte di giudici e insegnanti non è priva di difficoltà, l'affidabilità tra i valutatori non è molto alta, e i pregiudizi spesso influenzano i risultati. Studi in questo ambito hanno portato a numerosi e diversi contesti e strategie di valutazione. È stato dimostrato che una valutazione formale di una performance deve essere concettualizzata come un sistema complesso che contiene numerose influenze, dipendenti tra di loro.

## 3.2 Un modello di valutazione delle performance

In Figura 3.1 è presente una rappresentazione schematica dei principali problemi che riguardano la valutazione delle performance. Il modello illustra un insieme complesso di fattori che affliggono la performance e la sua valutazione, ed includono il contesto, i fattori musicali e non musicali, gli strumenti di valutazione e/o i criteri di valutazione,

le caratteristiche del musicista e del valutatore, e il feedback al musicista. Quest'ultima influenza sottolinea il doppio ruolo della valutazione: descrizione della performance, e guida per il miglioramento della performance. Il contesto della performance (`Performance Context`) è una delle influenze principali per la valutazione, e include almeno quattro fattori.

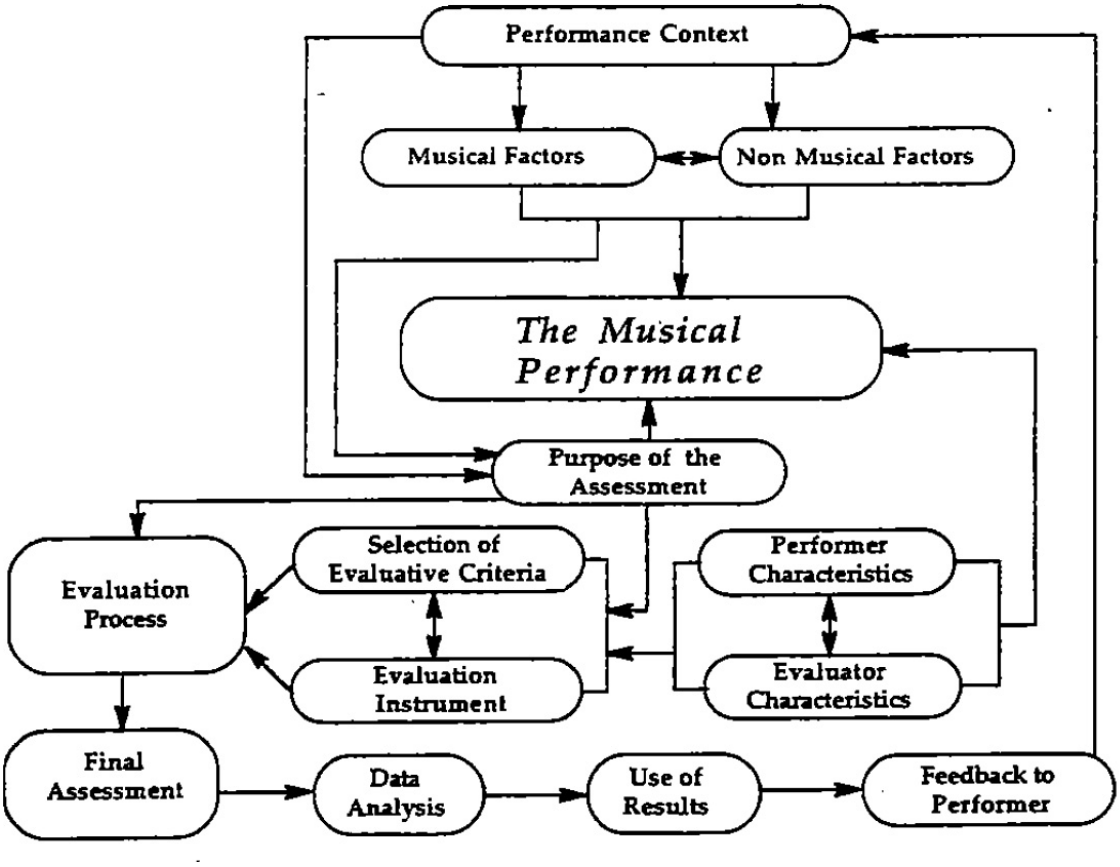



Figura 3.1: A Process Model of Assessing Musical Performance (adottato da Landy e Farr [48])

- Per primo lo scopo della valutazione (`Purpose of the Assessment`), che indica se il musicista sta eseguendo una competizione musicale, un festival, un esame, una recita, un'audizione, o se sta partecipando ad un progetto di ricerca sulla musica. Lo scopo influenza fortemente il modo con cui il giudice ascolta l'esecuzione musicale, e quindi poi anche come valuta la performance.

- Per secondo il tipo di performance (Type of Performance), la quale andrà ad influenzare il giudizio. McPherson [56] [57] propose cinque tipi distinti di performance musicali: sight reading (esecuzione a prima vista), esecuzione di un repertorio già provato, esecuzione del brano prendendo le informazioni dalla memoria del musicista, esecuzione ad orecchio (by ear), e improvvisazione. McPherson [57] [58]

afferma l'importanza di sviluppare diverse strategie per valutare ognuno di questi stili di performance, perché i criteri di valutazione necessitano di giudizi critici molto diversi tra le cinque abilità. Altre ricerche hanno tentato di sviluppare misure specifiche per ogni tipo di abilità. Per esempio Lidral [51] ha descritto una competizione di sola improvvisazione jazz, utilizzando uno strumento chiamato JIAF (Jazz Improvisers' Adjudication Form).

L'esecuzione di un brano utilizzando strumenti musicali diversi deve essere valutata in modo diverso, perché strumenti diversi implicano abilità tecniche diverse, e sono associati a diversi repertori.

Il pregiudizio affligge in maniera più o meno evidente in base al tipo di valutazione della performance utilizzato. Ad esempio Radocy [67] chiese a degli studenti di musica, di valutare un certo numero di performance. Gli studenti vennero ripartiti in cinque gruppi, ognuno con una 'condizione di pregiudizio' diversa (da nessun pregiudizio, e alto pregiudizio). Il pregiudizio nei vari gruppi fu ottenuto introducendo, prima della valutazione, delle informazioni false riguardo il musicista e il compositore. Gli studenti nel gruppo 'nessun pregiudizio' non hanno detto nulla riguardo il musicista, mentre gli studenti di un gruppo con 'pregiudizio moderato' hanno dato informazioni ingannevoli riguardo all'artista (ad es. al musicista è stata data l'etichetta di assistente di alto livello, oppure di ex esecutore di sinfonie). Così Radocy ha osservato un diffuso effetto di pregiudizio, e ha trovato che le performance con strumenti tipo il pianoforte, sono molto più suscettibili agli effetti del pregiudizio rispetto ad altri strumenti (esempio tromba o orchestra).

- Per terzo le proporzioni della performance (Performance Proportions), cioè se la performance è di un solista o di un gruppo di persone. Essa affligge la valutazione perché modifica la parte estetica della performance, e perché è soggetta ad influenze che non sono solo musicali. Per le gare corali, alcune ricerche suggeriscono che larghi gruppi di persone riducono la quantità di movimento fisico per evitare di far apparire il palco troppo pieno. Alcune osservazioni illustrano che le influenze non musicali nella valutazione variano in base alla grandezza della performance.

- Infine sia la performance che la valutazione vengono influenzate dall'ambiente della performance (Performance Environment), cioè la grandezza e l'acustica dello spazio in cui avviene la performance, e gli strumenti (casse, amplificatori, ecc.) a disposizione del musicista. Per esempio possono sorgere problemi di bilanciamento quando la voce e gli strumenti sono posizionati ad una distanza diversa rispetto al microfono.

I prossimi due punti del modello che vengono analizzati riguardano i fattori musicali e non musicali, i quali influenzano sia la performance sia la valutazione.

- I fattori musicali (`Musical Factors`) comprendono la scelta del repertorio, la forma e la struttura della musica, la grandezza del gruppo, l'abilità degli esecutori, e il tipo di strumento. Le espressioni della performance devono anche riflettere un'interpretazione implicita o esplicita del musicista. Se i giudici sono convinti dell'interpretazione, allora le loro valutazioni saranno molto positive. Infine, come già spiegato in precedenza, la valutazione della performance dipende dal tipo di strumenti coinvolti.

- I fattori non musicali (`Non Musical Factors`) possono avere un'influenza profonda nella valutazione (quando le influenze non musicali non sono pianificate, vengono considerati gli effetti del pregiudizio). Ad esempio Flores e Ginsburgh [34] hanno evidenziato un pregiudizio molto elevato in una competizione musicale per la regina Elisabetta. Esaminarono il risultato di 10 competizioni col violino che includevano 120 musicisti, e di 11 competizioni col pianoforte che includevano 132 musicisti. Le due classifiche finale dei musicisti erano in un ordine pressoché opposto all'ordine di esecuzione di ogni musicista, e quindi vennero condotte analisi per determinare una possibile relazione tra queste due variabili. Venne così sottolineato il fatto che le performance eseguite all'inizio avevano meno probabilità di essere ai vertici della classifica, mentre le performance eseguite per ultimo avevano una probabilità molto più elevata di arrivare tra i primi posti.

  Oltre all'effetto dell'ordine appena citato, ci sono altri fattori non musicali che possono avere effetti sull'abilità del performer nel suonare un repertorio musicale, o sulla percezione del giudice che deve valutare. Ad esempio l'interazione del musicista con altre persone, come l'insegnante o altri musicisti, può aiutare od ostacolare una performance. Inoltre ogni tipo di distrazione prima o durante la performance può avere effetti molto importante sulla qualità dell'esecuzione. Per non parlare di un possibile guasto dello strumento durante un'esecuzione.

La cosa più critica per qualsiasi valutazione è la qualità della performance musicale (`The Musical Performance`). Questo è mostrato al centro del nostro modello, ed è usato riferendosi all'atto reale dell'esecuzione musicale. Tuttavia, il modello mostra che ci sono molte altre influenze che vanno ad agire sulla performance e la sua valutazione.

Tra le tante influenze è da considerarsi molto importante quella relativa alle caratteristiche personali del musicista (`Performer Characteristics`), come ad esempio la tecnica e le abilità musicali. Tuttavia, un fattore che è spesso trascurato riguarda i processi mentali del performer prima e durante l'esecuzione. Questi includono il possibile nervosismo del performer, se il musicista si sente capace ad affrontare quell'esecuzione, se crede nel suo successo o nel suo fallimento, e se riesce a mantenere la concentrazione in maniera positiva lungo tutta l'esecuzione. Un'altra caratteristica importante per il performer

considera la consistenza di quello che il musicista riesce ad affrontare nelle performance, quindi riguarda il grado di familiarità con il luogo, gli strumenti, e l'esecuzione in pubblico. Altre caratteristiche includono l'apparenza, il modo di vestire del musicista, e la sua personalità, che vengono espresse dalle sua propensione al rischio, e dalla volontà di esibirsi in un ambiente confidenziale o aperto a tutti. Attualmente nessuna ricerca affronta questo tipo di fattori.

Interessante il fatto che alcune ricerche non hanno stabilito se ci sono differenze di sesso nello stile delle performance, sebbene differenze di sesso siano state riportate in studi sullo sviluppo musicale e sulle scelte dello strumento, come ad esempio Abels e Porter [2].

Le caratteristiche del valutatore (`Evaluator Characteristics`) invece, influenzano fortemente il risultato di ogni valutazione, e includono personalità, esperienza e abilità musicale, pratica nel giudicare, familiarità col performer e col repertorio. Il sesso non è di aiuto alla predizione di una possibile preferenza musicale, e non c'è evidenza nella differenza dei sessi. Tuttavia, le preferenze musicali possono essere fortemente modificate dalla personalità e dallo stato emozionale del performer e/o del giudice. Se i giudici sono incerti sulle loro abilità di fornire un giudizio obiettivo, allora sono molto più suscettibili di altri agli effetti del pregiudizio. Duerksen [23] ha trovato che i giudici sono generalmente più critici se la valutazione viene fatta su degli studenti anziché su dei musicisti professionisti. Nonostante gli insegnanti vedono l'autovalutazione come un componente prezioso per il miglioramento delle performance, Hunter e Russ [42] hanno riportato che senza pratica nel giudicare, gli studenti tendono a formare opinioni distorte delle loro abilità. In questo modo la valutazione di studenti dello stesso livello tende ad essere poco realistica. In questo studio sono stati reclutati degli studenti, che dopo un periodo di training hanno perfezionato la loro abilità, e hanno fornito delle valutazioni più obiettive. Le strategie di valutazione dipendono anche dalla familiarità che il giudice ha con il repertorio. Infatti vengono usati diversi criteri per lavori familiari e non. La valutazione di performance non familiari è più difficile, ed è più soggetta agli effetti del pregiudizio. La personalità, lo stato d'animo e l'attitudine del giudice sono molto importanti, perché un esaminatore che appare confidenziale e incoraggiante, è molto probabile che metta il musicista a suo agio, rispetto ad un giudice che appare impaziente e distaccato.

Lo scopo della valutazione (Purpose of the Evaluation) va a determinare la scelta dei criteri e degli strumenti di valutazione. I criteri variano a seconda della tipologia dello scopo, il quale può essere pedagogico, di ricerca, e di punteggio per stilare una graduatoria in competizioni e audizioni. Infatti, quando la musica è eseguita in un contesto non competitivo, è valutata diversamente rispetto a quando c'è l'obiettivo di raggiungere un certo punteggio in una sfida. Ad esempio gli studenti e gli insegnanti vedono spesso lo scopo della valutazione pedagogica in modo diverso, e quindi avranno opinioni diverse

delle performance.

La scelta dei criteri di valutazione (`Selection of Evaluative Criteria`) e degli strumenti di valutazione (`Evaluation Instrument`) modificano fortemente il processo di valutazione. Sono due gli scopi principali nello sviluppo di criteri standard e negli strumenti di valutazione, e sono il miglioramento della validità e dell'affidabilità delle valutazioni [50] [82]. La validità è il grado con il quale lo strumento di valutazione misura le variabili che si intende misurare; ad esempio l'eccellenza della performance, le abilità di improvvisazione, la lettura a prima vista. L'affidabilità invece è il grado di corrispondenza tra giudici diversi, o la consistenza con la quale un dato giudice valuta un insieme di performance in occasioni diverse. L'aumento di una delle due abilità, non va a influenzare l'aumento dell'altra. Per minimizzare il problema di affidabilità basse, dei ricercatori hanno sviluppato dei criteri di valutazione predeterminati, con il tentativo di migliorare l'affidabilità. Harold Abeles [1] ha affermato che:

*rating scales improve evaluation because adjudicators*
*must use a common set of evaluative dimensions rather than*
*develop their own subjective criticisms*

cioè le scale di valutazione migliorano la valutazione perché i giudici devono usare un insieme comune di valutazione invece dei propri criteri personali. In contrasto agli studi che indicano l'uso di criteri di valutazione predeterminati e specifici, si sono affacciati nella letteratura anche dei punti di vista alternativi. Per esempio Mills [55] espresse preoccupazione, negli ultimi due anni, riguardo l'uso di questi metodi di valutazione in una scuola superiore dell'Inghilterra. In questa scuola veniva dati voti da 0 a 3 in ognuna di cinque categorie di abilità (accuratezza delle note, accuratezza del ritmo, phrasing, controllo dello strumento, e tecnica adeguata al pezzo da suonare), e in ognuna di cinque categorie di interpretazione (dinamiche efficaci, tempo appropriato, senso opportuno degli stili, senso di coinvolgimento della musica, e senso della performance). Questa visione si basa sul presupposto che la performance musicale si possa spezzare in più componenti. Tuttavia, Mills ritenne che questa cosa non aveva molto senso, in quanto i musicisti trovavano questo metodo molto difficoltoso da applicare. Infatti Mills non analizzava se le singole note erano eseguite perfette, se il ritmo era adeguato ecc., ma osservava se la performance la sorprendeva, nonostante i possibili errori che potevano esserci.

I benefici di un programma di training per i giudici non sono stati ben chiariti. Si assume spesso che il training possa aumentare l'affidabilità del giudice, e diminuire i pregiudizi riguardo razza, sesso o convinzioni personali. Un importante scopo del training è quello di stabilire dei criteri consistenti tra i giudici, anche se i criteri enfatizzati dipendono

dai performer che vengono utilizzati per il training (quindi potrebbero essere enfatizzati criteri diversi a seconda dei performer scelti).

Il processo di valutazione (`Evaluation Process`) è qualche volta influenzato dai vincoli fisici, ad esempio la distanza tra valutatore e la performance, l'uso dello spartito musicale, l'ambiente fisico. Inoltre il processo che porta a una valutazione può essere guidato dall'aspettazione che il giudice ha sul musicista, o dalla conoscenza o dall'impatto visivo del performer. Quest'ultimo punto può avere un effetto o positivo o negativo sulla valutazione, specialmente quando il performer appare rigido, freddo, rilassato, e/o coinvolto. Schumann, citato da Davidson [20], fece la seguente osservazione

*If Liszt played behind a screen, a great deal of the poetry would be lost*

cioè "Se Liszt esegue un pezzo dietro ad uno schermo, sarebbe stato perduto un grande della poesia".
Davidson trovò che la performance visuale (cinematica) può comunicare e chiarire le intenzioni espressive del performer; la performance infatti, include mani, braccia, e movimenti della testa. Una conseguenza di questo è che potrebbe essere opportuno, in alcune circostanze, incoraggiare i giudici a considerare sia le informazioni visive che uditive per valutare una performance.

## 3.3 Analisi dei dati raccolti durante la valutazione

La valutazione finale (`Final Assessment`) deve essere riportata come una classificazione, un grado, oppure con una relazione. Una relazione scritta può essere riassunta e discussa, mentre un grado o una classificazione potrebbero essere analizzati per assicurare che ci sia un consenso adeguato tra i giudici.
L'analisi dei dati (`Data Analysis`) di valutazione ha il grande obiettivo di fornire feedback ai performer che vengono esaminati, e possono riguardare le procedure, i criteri di valutazione, gli strumenti utilizzati per la valutazione, e i giudici utilizzati. Alcune analisi hanno rivelato un numero importante di risultati sulla valutazione.
Wapnick [81] ha indicato che l'affidabilità non è significativamente legata alle seguenti cose:

- se il giudice è capace di usare gli strumenti di valutazione;
- le abilità di esecuzione del giudice;
- la durata delle performance valutate;
- se il giudice ascolta una performance live o registrata;

- se il giudice ha accesso allo spartito musicale;
- se il giudice usa una scala di valutazione.

I risultati di una valutazione di una performance potrebbero essere riportati per conto loro, o assieme ad altre informazioni. Alcuni risultati potrebbero essere usati (`Use of Results`) per scopi pratici, come ad esempio decidere i posti di collocamento, o scopi diagnostici. Dare dei feedback (`Feedback to Performer`) ai performer è appropriato, e aiuta a migliorare le performance future.

Nei paragrafi precedenti è stato proposto un modello che potrebbe essere usato per strutturare le ricerche future in questo ambito. Infatti, com'è stato reso evidente, gli studi esistenti sulla valutazione delle performance musicali sono estremamente limitati. Questo modello quindi serve allo scopo di avvisare i ricercatori sui principali divari nella ricerca, che devono essere colmati.
Emergono un numero importante di conclusioni dagli studi appena presentati. Per primo è che sarebbe necessario un approccio più globale e interdisciplinare, che cerchi di chiarire non solo gli strumenti di misura e i contesti musicali nei quali possono avvenire le performance, ma anche altri fattori come la personalità, e i pregiudizi sociali e culturali che influenzano la valutazione del giudice. Inoltre la conoscenza da parte del performer dei criteri utilizzati per la valutazione, può portare ad un miglioramento della performance. Infine, una delle variabili più importanti è proprio il giudice; infatti Fiske [24] ricorda giustamente che

*una valutazione di un performer non significa nulla finché non*
*conosciamo quanto affidabile sia il giudice che valuta la performance*

Alcune questioni ancora irrisolte sono le seguenti:

- fino a che punto un giudice sia abile a fornire una valutazione affidabile sotto contesti e scopi diversi (esame di musica, audizione, competizione ecc.);
- in quali situazioni e in che modo il training può aiutare ad alleviare alcuni dei problemi che minano l'affidabilità dei giudici.

## 3.4 Il Test di Turing applicato ai sistemi automatici per le performance musicali

Approcci procedurali o algoritmi per generare musica sono stati esplorati già da cinquanta anni. Occasionalmente, i ricercatori hanno tentato di valutare il successo di questi sistemi che generano musica, misurando la qualità percepita o la conformità di stile dell'output musicale. Questi test sono spesso condotti in forma di confronti tra output del computer, e output che non sono del computer. Alan Turing propose e usò un "Imitation Game" [77] come struttura per il confronto. In questo contesto, si assume che, se l'output della macchina suona simile (o è preferito) all'output umano, allora la macchina ha avuto successo. La natura di questo successo è stata messa raramente in discussione, ed è stata spesso interpretata come un'evidenza di successo di un sistema per generare musica.
Ariza C. [4] sostiene che i propositi di Turing non possono essere applicati durante l'esecuzione e la valutazione dei sondaggi fatti dagli ascoltatori di una performance musicale. Questo è un po' il filone conduttore del tema che viene affrontato nei paragrafi successivi. Infatti, verrà esaminato il concetto di Test di Turing musicale (TT musicale), verranno esaminati diverse varietà di test, e verrà mostrato che il test di Turing musicale non è attualmente conforme al modello di Turing. L'uso del TT nella valutazione di sistemi che generano musica, è superflua e potenzialmente ingannevole; infatti in un contesto musicale, non fornisce alcuna valutazione aggiuntiva rispetto a quella data da un ascoltatore.

### 3.4.1 Il Test di Turing (TT)

Nel 1950 Alan Turing [77] ideò un metodo per rispondere alla questione

*Può una macchina pensare?*

Il nome dato al suo metodo originale fu **The Imitation Game**, il quale include una interrogazione umana che, attraverso un'interfaccia di testo usata per rimuovere il suono nel parlato e l'apparenza visiva del soggetto, comunica con 2 agenti. Un agente è umano, e l'altro una macchina. Se l'interrogazione, attraverso i discorsi, non riesce a distinguere l'umano dalla macchina, allora la macchina (nella visione di Turing) è in grado di pensare. È importante far notare che Turing non definisce il pensare o l'intelligenza [18], e non sostiene che passare il test dimostri il pensiero o l'intelligenza [39].
Turing basò il suo test su un party game, nel quale lo scopo dell'interrogazione era tentare di distinguere il sesso di due agenti nascosti. Inoltre predisse che per l'anno 2000 un'interrogazione media, dopo cinque minuti di conversazione, potrebbe dare un'identificazione corretta non più del 70% delle volte. Altri invece dissero che ci sarebbero voluti di più di cento anni per far in modo che una macchina passasse regolarmente il test.

Turing rispose ad una questione sollevata da Geoffrey Jefferson [45], il quale constatò che non è sufficiente, per la mente di una macchina, usare solo le parole.

> *Esso dovrebbe essere in grado di creare concetti, e di trovare per se stesso parole adatte a esprimere cose in più rispetto alla conoscenza che già possiede.*

Appena al di là della capacità di creare concetti, Jefferson continua a suggerire che una tale macchina deve avere emozioni e consapevolezza di sé.

> *Siamo d'accordo che una macchina eguaglia un cervello almeno quando sarà in grado di scrivere un sonetto o comporre un concerto a causa di pensieri e emozioni sentite, e non per una caduta casuale di simboli; cioè non deve solo scrivere qualcosa, ma conoscere quello che ha scritto.* [45]

Per più di cinquanta anni ci sono state tremende discussioni e critiche al Test di Turing. Come fece notare Halpern [38], Turing è "*uno dei più stampati, citati, quotati, parafrasati, allusi, e generalmente si fa riferimento a documenti filosofici mai pubblicati*".

Un reclamo ampiamente citato, è chiamato *Chinese Room Argument* (CRA), ed è stato presentato da John Searle [72]. Esso discute il fatto che un umano traduce il cinese semplicemente attraverso una manipolazione diretta di simboli, e attraverso procedure di ricerca in tabelle. In questo caso, Searle dice che l'uomo ha solo la conoscenza della sintassi, e che questa cosa non può quindi essere vista come l'avere della conoscenza semantica. Questo umano infatti, con delle procedure di ricerca sufficienti, potrebbe apparire come uno che comunica in cinese anche se non ha nessuna conoscenza della lingua cinese. Da questo concetto Searle [38] sostiene che "*l'abilità di fornire buone risposte non implica necessariamente che il fornitore di queste risposte stia pensando; passare il TT non è una dimostrazione di intelligenza attiva*".

La questione sul fatto che il TT o altri test simili siano sufficienti a misurare il pensiero delle macchine, sarà discussa nel prossimo futuro [38] [71]. La risposta a questa domanda non è rilevante per gli studi attuali. Il TT, indipendentemente dalle sue potenzialità per la misurazione, fornisce almeno un benchmark ([1]) simbolico di un tipo di attitudine della macchina. Inoltre il TT ha ispirato altre forme di test di confronto tra l'output umano e della macchina.

### 3.4.2 La musica come mezzo nel Test di Turing

Per testare l'output di sistemi che generano musica, il Test di Turing potrebbe essere modificato assumendo la parte estetica, la musica o altre forme creative, come oggetto

[1] Test software volti a fornire una misura delle prestazioni riguardanti diverse operazioni

del test. Nel caso della musica, questo significa rimpiazzare completamente o in parte il testo scritto, con simboli o forme sonore. In questa occasione verranno presentati due modelli di questi tipi di test. Sebbene qualche volta si usi il linguaggio e il formato del TT, questi test fondamentalmente alterano il ruolo dell'interrogazione, e quindi non sono dei veri Test di Turing (infatti sono detti "test giocattolo" - *Toy Test* [39]).

- **Musical Directive Toy Test (MDtT)**. Usando l'interfaccia del computer, l'interrogazione manda una direttiva musicale a due agenti compositori. Uno dei compositori è una macchina, l'altro un umano. La direttiva musicale potrebbe essere il genere o lo stile, oppure potrebbe essere astratta (qualcosa del tipo "*scrivi musica triste*", "*scrivi una marcia*", o "*componi un gioco musicale*"). Le direttive potrebbero contenere musica, come frammenti melodici o ritmici sui quali l'agente compositore deve comporre. Le direttive vengono ricevute da entrambi gli agenti, i quali creeranno poi la musica. Dopo un po' di tempo avremo le due musiche prodotte in un certo tipo di formato (esempio spartito musicale, audio digitale, o audio digitale sintetico). Un MDtT flessibile potrebbe permettere all'utente che interroga di mandare un numero desiderato di direttive musicali. Un MDtT potrebbe prendere la forma di una chiamata e risposta musicale in real-time tra colui che interroga e gli agenti compositori. L'utente che interroga deve poi tentare di distinguere l'umano dalla macchina. Questo test conserva alcuni aspetti dell'interazione, e nella comunicazione sostituisce il linguaggio naturale con la musica.

- **Musical Output Toy Test (MOtT)**. In questo test, due agenti compositori forniscono a colui che interroga uno spartito, un audio digitale sintetico, e un audio digitale di una performance registrata. Uno dei compositori è una macchina, l'altro è un umano. La musica fornita può essere legata in termini di stile, strumentazione, o risorse musicali 'grezze', ma non è una composizione nuova in risposta a qualche direttiva specifica (come nel MDtT). Ogni agente potrebbe fornire dei lavori musicali multipli. Basandosi solo su questi lavori, colui che interroga deve tentare di distinguere l'umano dalla macchina. Questo test mantiene solo il confronto cieco dell'output tra due sorgenti, mentre l'interazione permessa dal Test di Turing è stata rimossa.

Nei due test precedenti il mezzo di comunicazione tra colui che interroga e i due agenti è la musica, quindi l'utente che interroga può contare molto fortemente sui propri giudizi musicali. Questo è in contrasto con il Test di Turing, il quale permette qualsiasi forma di discorso cieco attraverso la sintassi comune o il linguaggio naturale scritto, ed è quindi progettato per rimuovere visioni e valutazioni sonore soggettive.
Le interrogazioni nel TT tentano di distinguere tra umani e macchine bassandosi su costruzioni e contenuti linguistici. Le critiche su MDtT e MOtT sono senza limiti, perché

liberi di impiegare una vasta gamma di giudizi musicali. Senza le direttive musicali il MOtT, ancor più del MDtT, può provocare giudizi musicali inaffidabili e non rappresentativi. Questi giudizi possono essere influenzati da associazioni storiche e culturali sui stili musicali, aspettazioni su come la macchina e l'umano potrebbero suonare, o assunzioni su cosa potrebbe essere possibile con la tecnologia odierna.
Sia l'MDtT che MOtT danno un sondaggio del giudizio musicale, e non ne determinano il pensiero o l'intelligenza.

MDtT e MOtT sono spesso usati come un modo di argomentare il successo dei sistemi che generano musica. Tuttavia la connessione automatica tra i giudizi musicali positivi e il successo dei sistemi è discutibile. Alcuni output dei sistemi potrebbero non essere rappresentativi del sistema, e la critica è in una posizione debole a giudicare che cosa sia o non sia rappresentativo. Un sistema generativo può essere mal progettato, difficile da usare, non affidabile, o incapace di varietà. Pearce et al. [61] constatarono che

*la valutazione della musica prodotta dai sistemi rivela poco*
*riguardo la loro utilità come strumento compositivo.*

Sebbene Wiggins and Smaill dicono che la musica è un'attività intelligente, ammettono anche che alcuni tipi di attività musicali sembrano essere una risposta quasi inconscia e involontaria [85]. La musica non è necessariamente un'attività intelligente, ed non è certamente un test per l'intelligenza. La musica potrebbe essere percepita come un'attività intelligente, anche quando la sua creazione è il risultato di un'attività involontaria, irrazionale o algoritmica. Mentre la musica può essere un'attività molto umana, l'applicazione del Test di Turing alla musica ignora che la musica apparentemente di successo può provenire da fonti con poco pensiero o nessuno. L'anima, la mente, il pensiero, l'intelligenza, e la creatività sono fattori comuni deboli, anche se esteticamente sono musicalmente di successo.
Questo problema dei Test di Turing sulla musica è parte dei più grandi problemi dei derivati dal TT. I Test di Turing musicali sono una applicazione non corretta del TT, e possono portare a una sopravvalutazione dei sistemi che generano musica.

### 3.4.3 Discrimination Test (DT)

Test alternativi per il confronto alla cieca, non associati al TT, producono affermazioni implicite molto diverse da quelle che potrebbe dare il Test di Turing. Per queste ragioni è importante identificare i test di discriminazione (**Discrimination Test - DT**), come un tipo di sondaggio di ascolto che evita alcuni degli errori del TT musicale. Il Discrimination Test è simile al MOtT. Sebbene non sia esente da problemi di valutazione dei giudizi

musicali, tale test, quando propriamente vincolato, permette la generalizzazione di questi giudizi tra gruppi selezionati.
Modelli computazionali, come i sistemi che generano musica con obiettivi molto diversi da quelli dei strumenti creativi, richiedono strategie di valutazione particolari. Pearce e Wiggins proposero il DT, dove *la musica generata può essere valutata chiedendo a soggetti umani di assegnare l'appartenenza di questa musica a differenti insieme di dati che contengono pezzi composti dagli umani.* [62]. Se il pezzo composto dal sistema non può essere distinguibile da quello composto dall'umano *possiamo concludere che le composizioni della macchina sono indistinguibili dai pezzi composti dall'uomo.*
Nonostante la somiglianza tra il Discrimination Test e il Test di Turing, Pears e Wiggins hanno evidenziato alcune differenze significative. Il DT non è progettato per testare il pensiero delle macchine, ma per determinare *l'appartenenza o meno delle composizioni musicali in un insieme di pezzi di musica composti dall'uomo.* Inoltre hanno notato che l'elemento critico dell'interazione è stato rimosso: *nel nostro test i soggetti sono dei semplici ascoltatori passivi: non c'è interazione con la macchina.* Pearce e Wiggins argomentarono che sia il TT che il DT sono dei test comportamentali; i test sono usati per decidere se un comportamento può essere incluso in un insieme: "*l'insieme dei comportamenti intelligenti*" nel caso del Test di Turing, e "*l'insieme dei pezzi musicali di uno stile particolare*" nel caso del Discrimination Test.
Pears e Wiggins sostengono che "*le composizioni finali della macchina sono valutate oggettivamente in un sistema chiuso, che non lascia spazio a valutazioni soggettive o a meriti estetici*", nonostante evidenzino che all'interno del sistema chiuso del DT, i valori estetici sono difficili da rimuovere dai giudizi musicali.

### 3.4.4 Considerazioni finali sull'uso del TT musicale

I sistemi che generano musica non guadagnano niente nell'associare il loro output con il Test di Turing, anzi, la sovrastima dei sistemi causata da questa associazione, può svalutare la creatività reale nella progettazione e nell'interfaccia di questi sistemi.
Finché le macchine non raggiungeranno l'autonomia, è probabile che gli essere umani continueranno a formare, modificare, e manipolare l'output delle macchine per soddisfare le loro richieste estetiche. Quindi allo stato attuale l'uso del Test di Turing, nella valutazione di sistemi che generano musica, è superflua e non fornisce alcuna valutazione aggiuntiva rispetto a quella data da un ascoltatore.

## 3.5 Rencon Workshop

Rencon (Performance Rendering Contest) è un progetto di ricerca a livello mondiale nell'ambito dell'informatica musicale, il quale organizza delle gare (contest) per sistemi computazionali automatici che generano performance musicali espressive, in modo autonomo o in modo interattivo. Lo scopo è quello di selezionare il miglior sistema in grado non solo di riprodurre un brano musicale, ma soprattutto di eseguirlo aggiungendoci personalità, emotività, ed espressività. Nel campo dell'informatica, la valutazione di questi sistemi è richiesta perché la sensualità e la bellezza è molto importante per un musicista, inoltre una valutazione soggettiva delle performance generate è importante per la ricerca sui sistemi che generano le performance. Le valutazioni che si ottengono durante il contest, dove si riuniscono diversi sistemi e si sfidano tra di loro, andranno a stimolare gli sforzi scientifici in questo campo. Rencon iniziò nel 2002 con questa prospettiva. Col passare degli anni oltre all'obiettivo di incitare, attraverso il whorkshop, il miglioramento delle tecniche di rendering, è nata anche la voglia di iniziare a trattare e a manipolare anche con le interfacce di performance per l'espressione umana, contribuendo alla modellazione delle tecniche sulle attività mentali umane, per poi applicarle all'istruzione, alla fruizione e alla creazione di nuova musica.
Nel 2011 il Rencon Workshop si è svolto per la prima volta in Italia il 6 luglio 2011, presso il Dipartimento di Ingegneria dell'Informazione (DEI [2]) dell'Università degli Studi di Padova [3], durante l'8th Sound and Music Computing Conference (SMC 2011 [4]). Gli obiettivi che il workshop si propone per il futuro sono:

- nel 2025 non ci saranno differenze tra le performance eseguite automaticamente da un sistema computazionale, e le performance eseguite da un essere umano. Si potranno creare dei CD di performance create con i sistemi computazionali.

- nel 2050 si vincerà il Chopin Contest [5].

- nel 2052 si vincerà sia il Tchaikovsky Contest [6] sia il Chopin Contest. I sistemi automatici per il rendering delle performance inizieranno ad insegnare ai musicisti umani.

[2] www.dei.unipd.it
[3] www.unipd.it
[4] http://smc2011.smcnetwork.org/
[5] Il Chopin Contest (o Chopin Competition, o International Chopin Piano Competition) è una competizione al pianoforte che si tiene a Varsavia in Polonia, in onore di Frédéric Chopin. Il contest è una delle poche competizioni dedicate itneramente al lavoro di un singolo compositore.
[6] Il Tchaikovsky Contest (o International Tchaikovsky Competition) è una competizione musicale classica che si tiene ogni quattro anni a Mosca in Russia. La competizione è rivolta ai pianisti, violinisti e violoncellisti dai 16 ai 30 anni di età, e ai cantanti dai 19 ai 32 anni.

- nel 2100 un musicista umano, addestrato da un sistema automatico per il rendering delle performance, vincerà il Chopin Contest.

### 3.5.1 Stage del workshop

Rencon si è svolto in due differenti stage.

1. **Stage I**. Lo Stage I si è svolto il 27 marzo 2011, circa quattro mesi prima del workshop vero e proprio del 6 luglio 2011. In questa prima fase, svoltasi interamente tramite internet, vengono raccolti dei pezzi musicali suonati al piano tramite i sistemi automatici dei partecipanti, i quali verranno poi valutati basandosi sull'impressione artistica (tramite una giuria di musicisti), e sui meriti tecnici del brano (tramite un comitato tecnico). Alla fase successiva sono stati ammessi solo otto sistemi, e quello che segue è un elenco degli otto finalisti dello Stage I nell'ordine di arrivo (dal primo all'ottavo posto):

    - YQX di Sebastian Flossmann (Johannes Kepler University), Maarten Grachten (Johannes Kepler University) e Gerhard Widmer (Johannes Kepler University).
    - Director Musices (DM) di Erica Bisesi (KTH), Anders Freiberg (KTH) e Richard Parncutt (University of Graz).
    - Virtual Philharmony (VP) di Takashi Baba (Kwansei Gakuin University), Mitsuyo Hashida (Kwansei Gakuin University) e Haruhiro Katayose (Kwansei Gakuin University).
    - Shunji System di Shunji Tanaka (Kwansei Gakuin University), Mitsuyo Hashida (Kwansei Gakuin University) e Haruhiro Katayose (Kwansei Gakuin University). Operater: Mami Yamaguchi.
    - Kagurame Phase-II di Taizan Suzuki (Picolab Co., LTD), Tatsuya Hino (Shibaura Institute of Technology), Masahiro hibasaki (Shibaura Institute of Technology) e Yukio Tokunaga (Shibaura Institute of Technology).
    - Usapi di Keiko Teramura (Kyoto University) e Shin-ichi Maeda (Kyoto University).
    - Kagurame Phase-III di Taizan Suzuki (Picolab Co., LTD), Tatsuya Hino (Shibaura Institute of Technology), Masahiro Hibasaki (Shibaura Institute of Technology) e Yukio Tokunaga (Shibaura Institute of Technology).
    - CaRo 2.0 di Sergio Canazza (Università di Padova), Giovanni De Poli (Università di Padova), Antonio Rodà (Università di Padova), Massimiliano Barichello (Università di Padova) e Davide Ganeo(Università di Padova). Performer: Davide Tiso. [16]

2. **Stage II**. Lo Stage II si è svolto a Padova il 6 luglio 2011, ed è il workshop vero e proprio nel quale si sono riuniti tutti i partecipanti che hanno passato la prima fase. Il sistema di rendering, sviluppato e portato da ogni partecipante dello Stage II, dovrà eseguire per due volte un brano per pianoforte in maniera espressiva (Performance A e Performance B), in modo tale da rendere l'esecuzione simile ad una performance musicale umana. Tutte e due le esecuzioni espressive del brano selezionato verrano ascoltate e poi valutate dai partecipanti del Rencon Workshop (musicalmente esperti e non), e dagli utenti online che seguivano la manifestazione.

   Per il rendering espressivo è stato selezionato il brano *Sonata No. 8 Op. 13 III. Allegro di Beethoven* dalla seguente lista di 20 brani:

   J. S. Bach: Wohltemperierte Klavier I-1 BWV846 Prelude.

   J. S. Bach: Invention No. 15 BWV786.

   J. S. Bach: The Little Notebook for Anna Magdalena Bach, No. 1 Menuette in G major (no.24).

   T. Badarzewska: A Maiden's Prayer.

   L.v. Beethoven: Piano Sonata No. 8, 3rd Mov.

   L.v. Beethoven: Bagatelle No. 25 in A minor "For Elise".

   J. Brahms: Hungarian Dances No. 5 in F-sharp minor.

   F. Chopin: Nocturne No. 2 in E Flat Major, Op.9-2.

   F. Chopin: Etude No. 3 in E major, Op.10 Nr.3 "Chanson de L'adieu".

   F. Chopin: Waltz No.9 in A flat major, Op.69 Nr.1 "L'adieu".

   E. Elgar: Salut d'amour.

   G. Faure: Sicilienne in G minor, Op. 78.

   G.F. Händel: The Harmonious Blacksmith.

   F. Liszt: 2 Konzertetüden, S.145, No. 1 "imparare".

   F. Mendelssohn: Songs Without Words Op. 30 No. 6, fis-moll "Venezianisches Gondellied".

   W. A. Mozart: Piano Sonata K. 545, 1st Mov.

   W. A. Mozart: Piano Sonata K. 331, 3rd Mov. "Turkish march".

   D. Scarlatti: Sonata in C major K. 159, Allegro.

   R. Schumann: Abegg Variations Op.1: 1. Theme.

   P. I. Tchaikovsky: The Seasons: 7. July.

Ad ogni partecipante è stato consegnato il brano in formato MusicXML (1.0/2.0) e Standard MIDI File (SMF format 1). Inoltre è stata fornita la partitura del brano in forma cartacea. La partitura da eseguire contiene note e alcuni segni espressivi, ma nessuna indicazione sulla struttura delle frasi e sugli accordi. Ai partecipanti sono dati 60 minuti per sistemare il proprio software e generare attraverso il proprio sistema l'esecuzione espressiva richiesta. Durante il processo di rendering espressivo non sono permesse modifiche manuali del MIDI prodotto, e nemmeno l'ascolto del MIDI prima che questo venga pubblicamente eseguito.

- Per i sistemi computazionali automatici e non interattivi (Director Musices, Usapi, Kagurame Phase-II, Kagurame Phase-III, Shunji System e YQX), ogni interpretazione espressiva generata dal sistema viene salvata in formato MIDI, e successivamente viene riprodotta sul pianoforte, controllato dal computer, in maniera autonoma.
- Per i sistemi computazionali automatici e interattivi (VirtualPhilarmony e CaRo 2.0), ogni interpretazione espressiva veniva eseguita in real-time dal sistema direttamente sul pianoforte, con l'aiuto di un performer.

Per l'esecuzione delle performance espressive prodotte dai sistemi in gara, è stato utilizzato un pianoforte Yamaha Disklavier (vedi Figura 3.2). Disklavier è il marchio di una serie di pianoforti prodotti da Yamaha Corporation. Le varie forme e tipologie di Disklavier sono essenzialmente dei pianoforti moderni elettromeccanici che usano sensori ottici a LED, che permettono l'esecuzione delle note e che usano i pedali indipendentemente da qualsiasi operatore umano. Molti modelli si basano su pianoforti acustici reali e sono stati costruiti con sensori ed elementi elettromeccanici che non interferiscono con la normale esecuzione dello strumento. Inoltre hanno la possibilità di salvare dei dati e riprodurli in un altro momento; tali dati possono essere anche performance musicali eseguite da pianisti umani. I Disklavier sono dotati di molte porte di ingresso: per i dati MIDI, e anche da molti altri dispositivi di memorizzazione come floppy disks, cd-rom, usb, e cavi seriali.

### 3.5.2 Valutazione delle performance e risultati

Come già detto ogni performance, cioè Performance A e Performance B di ogni partecipante allo Stage II, viene valutata singolarmente appena dopo la sua esecuzione sia online che per via cartacea. La somma dei due punteggi decreta il punteggio dello Stage II di ogni partecipante, andando così a formare una classifica che decreta il vincitore della seconda fase di Rencon 2011. Per problemi tecnici i sistemi Kagurame Phase-II e Kagurame Phase-III non hanno potuto partecipare a questa fase.

Figura 3.2: Il pianoforte Yamaha Disklavier

Le due performance musicali del sistema CaRo 2.0 sono state eseguite in real-time dal maestro Davide Tiso il quale, grazie alla sua abilità di musicista, ha saputo dare al brano prodotto un'espressività convincente che ha permesso al sistema CaRo 2.0 (il sistema che genera performance espressive dell'Università di Padova) di vincere, alla sua prima partecipazione, il secondo stage dell'edizione 2011 del Rendering Contest.

Di seguito è riportata la classifica dello Stage II, e la classifica finale del Rencon.

**CLASSIFICA STAGE II**

| Entrant System | Performance A | | | Performance B | | | TOTAL |
|---|---|---|---|---|---|---|---|
| | Net | Paper | Total | Net | Paper | Total | SCORE |
| 1st: CaRo 2.0 | 56 | 464 | 520 | 61 | 484 | 545 | 1065 |
| 2nd: YQX | 64 | 484 | 548 | 65 | 432 | 497 | 1045 |
| 3rd: VirtualPhilarmony | 46 | 452 | 498 | 32 | 428 | 460 | 958 |
| 4th: Director Musices | 33 | 339 | 372 | 13 | 371 | 384 | 756 |
| 5th: Shunji System | 24 | 277 | 301 | 20 | 277 | 297 | 598 |

**CLASSIFICA FINALE**

| Entrant System | Rank of Stage I | Rank of Stage II | Score of I + II |
|---|---|---|---|
| 1st: YQX | 1 | 2 | 3 |
| 2nd: VirtualPhilarmony | 2 | 3 | 5 |
| 3rd: Director Musices YQX | 3 | 4 | 7 |

Come si può vedere CaRo 2.0 ha vinto lo Stage II, e YQX ha vinto l'*SMC-Rencon Award*, che indica la vittoria finale del contest. Inoltre è stato dato un premio speciale sulla tecnica della performance (*SMC-Rencon Technical Award*) al vincitore dello Stage I, cioè sempre YQX.

## 3.6 Valutazione questionario (progetto)

Durante il Rencon Workshop 2011 (SMC-Rencon), appartenente all'8th Sound and Music Computing Conference, e tenutosi a Padova il 6 luglio 2011, sono state eseguite delle performance musicali utilizzando dei software per aggiungere espressività al brano musicale inizialmente proposto (vedi sezione 3.5). Ogni partecipante al workshop utilizzava il proprio sistema eseguendo due performance per il brano scelto; ognuna di queste performance poteva poi essere valutata in due modi diversi:

- online, tramite un sito creato appositamente che permetteva la votazione solo in dei tempi limitati post esecuzione dei brani;

- in maniera cartacea, tramite dei fogli che venivano distribuiti ai partecipanti presenti nella sala del workshop.

L'uso dei fogli aveva il vantaggio di lasciare più tempo per esprimere il proprio giudizio, ma avevano lo svantaggio di permettere la votazione solo alle persone presenti, un numero assai limitato rispetto al numero di persone che potrebbero votare tramite internet.
Oltre ai fogli riguardanti la valutazione delle performance ascoltate, è stato distribuito un questionario, preparato dal professor Giovanni De Poli, da compilare in anonimato e relativo al modo di valutare le performance ascoltate dai presenti durante il workshop.
Alla termine della gare i questionari sono stati raccolti, e poi successivamente analizzati. Verranno ora mostrate le analisi eseguite e le considerazioni fatte sulle risposte che sono state date.

### 3.6.1 Raccolta dati in Excel

Il primo passo è stato quello di portare tutte le informazioni contenute nei questionari in un formato più fruibile per effettuare l'analisi dei dati. In un'unica scheda o foglio di lavoro di Excel (chiamato *Dati*) sono state quindi inserite le risposte di tutti i 40 questionari raccolti durante il workshop, separando le risposte appartenenti a domande diverse. Ad ogni riga della tabella Excel corrispondono le risposte di uno spettatore.

- Le prime tre colonne sono relative alla prima domanda del questionario "*Your evaluation was made with reference to*", alla quale bisognava dare una sola risposta scegliendo tra tre possibili scelte:

  "*master degree music student*"

  "*music teacher*"

  "*top-level performer*"

- Le successive otto colonne sono relative alla seconda domanda a risposta multipla "*Which are the main factors that influenced your judgment?*". Le possibili scelte sono le seguenti: *the performance*

  "*is able to highlight elements related to the musical structure (phrasing, counterpoint)*"

  "*contains unexpected but interesting choices*"

  "*is consistent with the suitable musical style (from an historical point of view)*"

  "*is consistent with the favourite style (from a subjective point of view)*"

  "*is consistent with the style of a famous performer*"

  "*is able to convey emotional contents*"

  "*contains evident musical errors (things that a human would never do)*"

  "*is consistent with the actual performance context (concert hall, classroom, automatic performance contest)*"

- Le successive 3 colonne sono relative a dati che riguardano lo spettatore:

  "*musical training*" (less than 5 years, or 5 years or more)

  "*gender*" (M o F)

  "*age*"

- L'ultima colonna è invece relativa ad una domanda aperta, alla quale lo spettatore poteva rispondere quello che voleva. La domanda è la seguente "*Which (English or Italian) adjectives would you suggest to be used in Rencon 2012 evaluation stage?*"

Notazioni adottate nella tabella Excel che raccoglie tutte le risposte:

- nelle prime due domande si indica con 1 la risposta data, 0 altrimenti;

- nella domanda relativa al *musical training* con 1 si indica "5 years or more", con 0 "less than 5 years";

- nella domanda relativa al *gender* con 1 si indica "M", con 0 "F";

- nella domanda relativa a *age* c'è semplicemente l'età scritta;

- nella domanda aperta invece viene copiato quello che lo spettatore ha scritto nel foglio del questionario.

Durante l'inserimento dei dati alcuni spettatori non hanno risposto a tutte le domande, altri hanno dato una risposta scritta nella prima domanda (che prevedeva una scelta). In questi casi è stato messo il valore 0 per le risposte errate che riguardano la prima domanda, il simbolo \ per le risposte riguardanti i dati personali, ed è stato cambiato lo sfondo in rosso per indicare l'errore commesso. Come verrà visto meglio dopo, è da notare che lo 0 che viene messo sulle tre scelte della prima domanda nel caso di errori, rende la risposta completamente diversa da qualsiasi altra risposta corretta (che avrà almeno una scelta selezionata con 1) .
Un porzione della tabella Excel si può vedere nella Figura 3.3

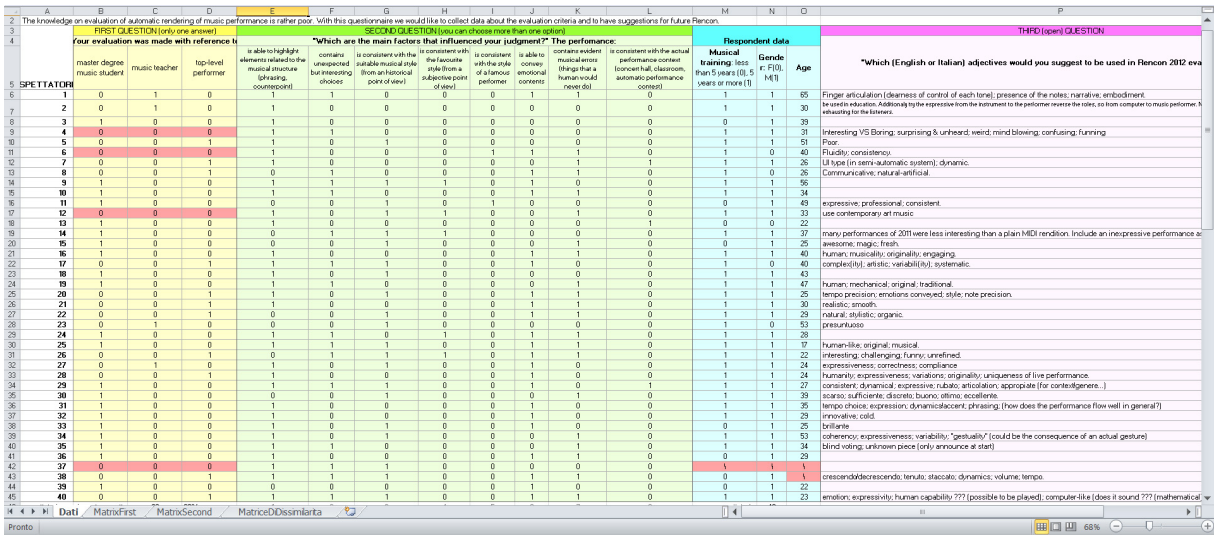


Figura 3.3: Il foglio di calcolo *Dati*

### Analisi statistica dei dati

Il secondo passo è stato quello di fare un'analisi statistica dei dati raccolti.

Nella prima domanda è da notare l'alto numero di risposte sbagliate che sono state date (10% del totale). La causa più accreditata ad aver causato un così basso numero di risposte corrette, potrebbe essere la domanda posta male, creando quindi incertezza nel momento in cui uno spettatore deve segnare una delle tre risposte. Un suggerimento per migliorare questa domanda sarebbe quello di renderla più chiara nella sua formulazione, in modo che lo spettatore capisca che la domanda chiede su che base poggia la valutazione dello spettatore, e non la formazione musicale dello spettatore; infatti, sembra che molti spettatori l'abbiano interpretata in quest'ultimo modo.

L'interpretazione corretta è la seguente: *se ad esempio lo spettatore da come voto 6 ad una esecuzione, la scala (da 1 a 10) che ha usato, è per la valutazione di un master degree music student, di un music teacher o di un top-level performer?*.

Andando ad osservare solo le risposte corrette invece, e come si può anche vedere in Figura 3.4, possiamo notare che un 58,3% degli spettatori ha valutato le performance rispetto un master che riguarda la musica, il 30,6% rispetto un performer, e un 11,1% rispetto un insegnante di musica. Si può concludere che la maggior parte degli spettatori (58,3%) ha valutato basandosi sullo stesso tipo di scala di valutazione (master degree music student), anche se a mio parere molte risposte sono state date errate (e un po' a caso per gli spettatori non esperti di musica) a causa dei problemi citati sopra.

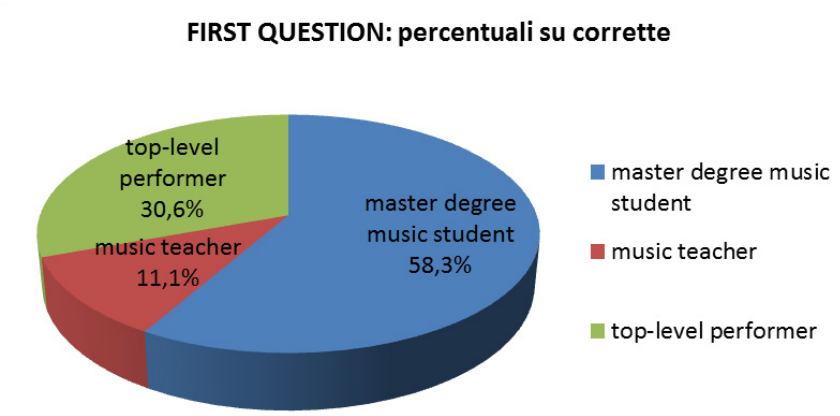



Figura 3.4: Risposte alla prima domanda del questionario

Nella seconda domanda si può notare come sia stata fortemente scelta la risposta numero 1: è stata segnata dal 77,5% dei spettatori (abilità nel sottolineare elementi relativi alla struttura musicale). Numerosi voti hanno avuto anche la risposta numero 6 col 57,5% (abilità nel trasmettere contenuti emozionali), la risposta numero 3 (consistenza con lo stile della musica, dal punto di vista storico) e numero 7 (presenza di errori musicali non creati dall'umano) entrambe col 52,5%, e la risposta numero 2 col 40% (scelte inaspettate ma interessanti). Le altre risposte hanno avuto percentuali molto più basse. Ecco l'elenco delle percentuali per ogni risposta:

| | |
|---|---|
| 77,5% | is able to highlight elements related to the musical structure (phrasing, counterpoint) |
| 40,0% | contains unexpected but interesting choices |
| 52,5% | is consistent with the suitable musical style (from an historical point of view) |
| 1,5% | is consistent with the favourite style (from a subjective point of view) |
| 5,0% | is consistent with the style of a famous performer |
| 57,5% | is able to convey emotional contents |
| 52,5% | contains evident musical errors (things that a human would never do) |
| 7,5% | is consistent with the actual performance context (concert hall, classroom, automatic performance contest) |

Come si può vedere dal grafico a barre in Figura 3.5, ogni spettatore ha dato in media 3,05 risposte (la media è stata calcolata).

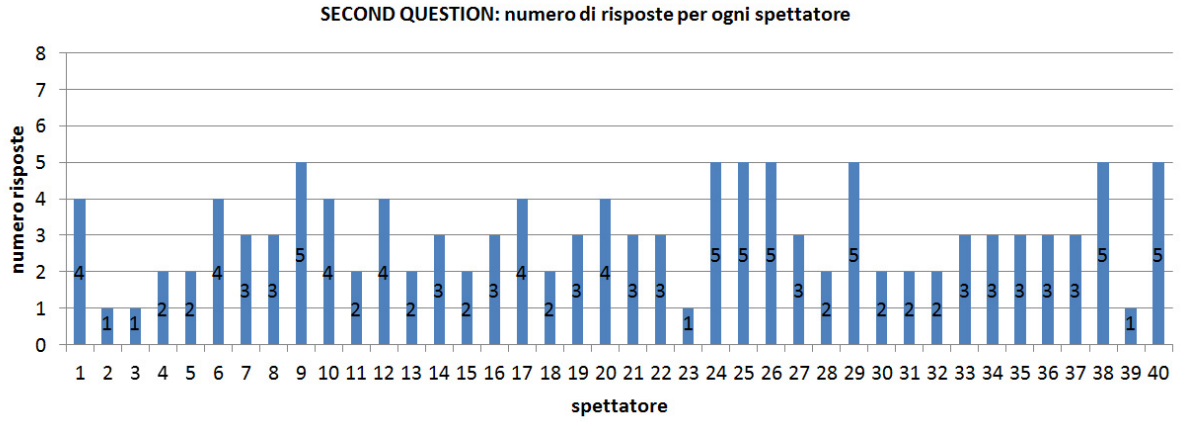



Figura 3.5: Numero di scelte che ha dato ogni spettatore alla seconda domanda del questionario

Nella domanda riguardante i dati personali, si può solo osservare che la maggior parte degli spettatori sono maschi di 35 anni (85% del totale), e che il 77,5% degli spettatori hanno una conoscenza musicale superiore ai 5 anni. Osservando i dati raccolti dalla tabella però, non si evidenziano particolari distinzioni tra sesso, età, e conoscenza musicale.

### Matrice di dissimilarità

Il prossimo passo consiste nel costruire la matrice di dissimilarità, sulla quale possono venire effettuate le analisi Cluster e MDS. La matrice di dissimilarità, o matrice delle distanze, è una matrice n x n dove l'elemento inserito nella cella [i,j] indica la distanza tra l'elemento i e l'elemento j. Nel nostro caso, essendoci 40 spettatori, la matrice sarà

40 x 40, e le distanze vengono calcolate in questo modo: la cella [i,j] conterrà il numero di risposte discordanti tra lo spettatore i e lo spettatore j.

- Nel prima domanda del questionario abbiamo una singola scelta tra 3 possibili risposte, quindi la distanza tra i e j verrà incrementata di 1 solo se le risposte date dai due spettatori sono diverse.

- La seconda è una domanda a scelta multipla tra 8 possibili risposte quindi, considerando una sola tra le otto risposte, la distanza viene incrementata di 1 se quella risposta è stata segnata da solo uno tra gli spettatori i e j. In sostanza se le 8 risposte tra i due spettatori sono tutte quante diverse, la distanza potrebbe incrementare di 8.

- La domanda riguardante i dati personali dello spettatore, e la domanda aperta, non vengono considerati per il calcolo della matrice di dissimilarità.

Per costruire tale matrice è stato utilizzato sempre Excel. Il lavoro è stato diviso in tre parti (due matrici parziali, e la matrice finale di dissimilarità) per rendere più semplice e meno confusionaria la formulazione delle formule nelle celle del foglio di calcolo.
In una nuova scheda (*MatrixFirst*) è stata creata una matrice calcolando solo le distanze estraibili dalla prima domanda del questionario. Per vedere se le risposte di due spettatori alla prima domanda sono diverse, basta considerare i tre valori interi (0 e 1), che indicano le tre risposte, come un numero binario a 3 cifre: due risposte di due spettatori sono diverse se i numeri binari corrispondenti sono diversi. Quindi per trasformare questo in una formula di Excel basta utilizzare la notazione posizionale per trasformare il numero binario in decimale, e il costrutto SE per controllare se i due numeri in base 10 sono uguali.
Esempio di formula (le risposte del primo spettatore sono ad esempio nelle celle B6-C6-D6, e quelle del secondo in B7-C7-D7):

```
= SE( (Dati!B6*4+Dati!C6*2+Dati!D6*1)=(Dati!B7*4+Dati!C7*2+Dati!D7*1) ; 0 ; 1 )
```

La matrice si può vedere in Figura 3.6.
La matrice relativa alle distanze estraibili dalla seconda domanda è stata creata in un'altra scheda di Excel (*MatrixSecond*). La distanza tra due spettatori è stata calcolata come sommatoria di `SE`, uno per ogni risposta. Un esempio di formula relativa ad una cella della matrice (le risposte del primo spettatore sono nelle celle E6:L6, e quelle del secondo in E7:L7):

```
= SE(Dati!E6=Dati!E7;0;1) + SE(Dati!F6=Dati!F7;0;1) + SE(Dati!G6=Dati!G7;0;1)
+ SE(Dati!H6=Dati!H7;0;1) + SE(Dati!I6=Dati!I7;0;1) + SE(Dati!J6=Dati!J7;0;1)
            + SE(Dati!K6=Dati!K7;0;1) + SE(Dati!L6=Dati!L7;0;1)
```

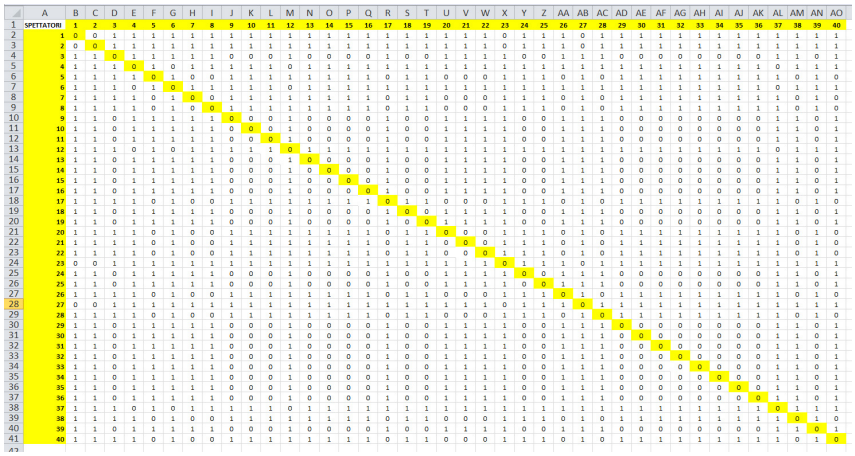


Figura 3.6: Prima matrice parziale (scheda MatrixFirst)

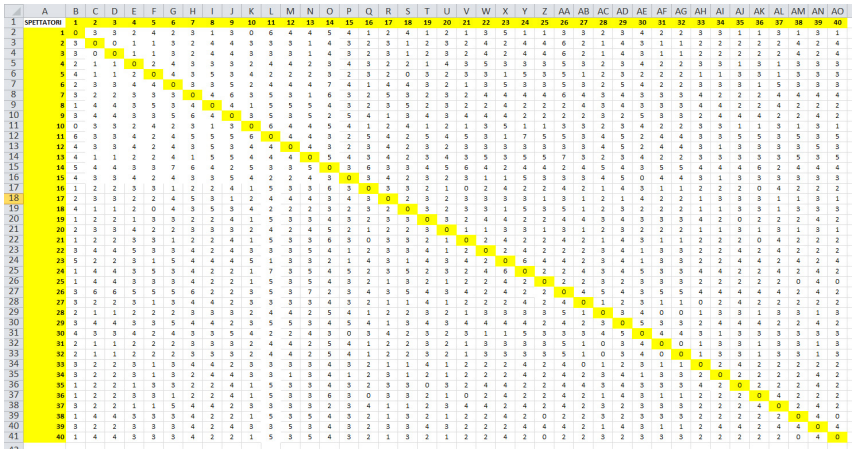


Figura 3.7: Seconda matrice parziale (scheda MatrixSecond)

In Figura 3.7 si può vedere la matrice relativa alla seconda domanda.
Per concludere non resta che creare la matrice di dissimilarità completa inserendola in un'altra scheda di Excel (*MatriceDiDissimilarita*). Tale matrice è molto semplice da creare, in quanto consiste nella somma dei valori, contenuti nelle celle corrispondenti, delle altre due matrici create in precedenza. La matrice di dissimilarità è quindi quella in Figura 3.8.

### 3.6.2 Elaborazione in R

Una volta costruita la matrice di dissimilarità, può essere utilizzata per effettuare delle analisi Cluster e/o MDS. **R** [7] è un ambiente di sviluppo open source specifico per l'analisi statistica dei dati, e quindi verrà utilizzato per effettuare queste analisi utilizzando la

[7] http://www.r-project.org/

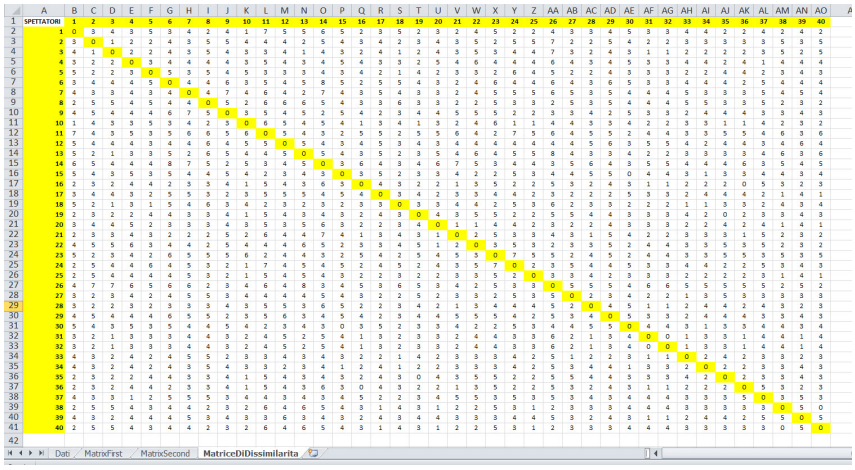


Figura 3.8: Matrice di dissimilarità (scheda MatrixSecond)

matrice di dissimilarità come input. R utilizza un linguaggio orientato agli oggetti, e ora analizzeremo il codice che è stato prodotto per effettuare le analisi a noi necessarie.

Per leggere, scrivere e manipolare dati da un file Excel di tipo XLS o XLSX basta utilizzare il Package `XLConnect`, e caricare il contenuto nel `workbook` (`wb`). Il package `XLConnect` richiede solo un'installazione Java nel computer, e lavora sulle principali piattaforme: Windows, UNIX/Linux and Mac ("R Data Import/Export", n.d.).

```
require("XLConnect")
wb <- loadWorkbook("MatriceExcel2007.xlsx", create = FALSE)
summary(wb)
```

con il commando `summary` viene mostrato a video (nel terminale di R) un riassunto del contenuto del workbook, cioè vengono mostrati le schede del file Excel (worksheets), i nomi, le schede nascoste, ecc. Poi dobbiamo caricare solo i dati contenuti nella scheda MatriceDiDissimilarita, perché i dati contenuti nelle altre schede non ci servono per l'elaborazione in R. Per farlo basta utilizzare il metodo `readWorksheet`, il quale legge i dati dalla scheda indicata, e li butta in una matrice (matr) interpretando la prima riga come i nomi delle colonne.

```
matr <- readWorksheet(wb, sheet = "MatriceDiDissimilarita")
```

Per poter essere utilizzata la matrice deve essere una matrice quadrata. A causa di come sono disposti i dati nel foglio Excel, la colonna A di Excel contiene le etichette dei 40 spettatori, e quindi tale colonna è presente anche in `matr`. Per far in modo di avere una matrice quadrata contenente solo i dati da elaborare, dobbiamo togliere la prima colonna.

```
matrTXT <-matr[, -1]
```

Per effettuare la Cluster Analisy abbiamo bisogno della matrice delle distanze (o di dissimilarità), nel nostro caso ce l'abbiamo già, ed è `matrTXT`. Il passaggio successivo è quello di effettuare il processo di identificazione di gruppi (cluster) per poter osservare le similarità tra i vari gruppi ("Statistica con R: Cluster Analysis in R", n.d.). Ci sono due tipi di approcci: supervisionato e non supervisionato.

1. nel supervisionato cataloghiamo gli oggetti in base a dei criteri prestabiliti;
2. nel non supervisionato la classificazione è esplorativa, e si basa sulle caratteristiche rilevate per singola unità statistica. Ci sono 2 tipi di clustering non supervisionato: il clustering partitivo e il clustering gerarchico.
    - nel **clustering partitivo** si definisce l'appartenenza ad un gruppo utilizzando la dissimilarità da un punto rappresentativo del cluster (centroide);
    - nel **clustering gerarchico** viene invece creata una gerarchia di partizioni caratterizzata da un numero crescente o decrescente di gruppi (albero o dendrogramma). In questo tipo ci sono tanti algoritmi, i principali hanno due tipi di approccio.

      Approccio bottom-up (il più usato), nel quale si creano tanti cluster quanti sono gli oggetti (singleton), e per passi successivi si uniscono gli oggetti tra loro più vicini, fino ad ottenere un gruppo con tutti gli oggetti (**hierarchical agglomerative clustering**).

      Approccio top-down, nel quale da un unico cluster contenente tutti gli oggetti, si splitta l'insieme fino ad ottenere clusters contenenti ciascuno un solo oggetto (hierarchical divisive clustering)

R ha una grande varietà di funzioni che permettono la cluster analisi. Come verrà visto dopo, sono stati provati i seguenti metodi: clustering partitivo, e hierarchical agglomerative clustering.

Prima di effettuare qualsiasi tipo di analisi cluster, si consiglia di rimuovere o di stimare i dati mancanti (non è il nostro caso, perché la matrice `matrTXT` contiene tutti i valori), e di standardizzare le variabili per permetterne il confronto ("Quick-R: Cluster Analysis", n.d.).

```
matrTXT<- na.omit(matrTXT)
matrTXT<- scale(matrTXT)
```

### Multi Dimensional Scaling (MDS)

L'MDS è una tecnica esplorativa dei dati che permette di ottenere una rappresentazione di N oggetti in uno spazio a k dimensioni, partendo da informazioni relative alla similarità

(o dissimilarità) tra ciascuna coppia di oggetti. Di solito viene utilizzato per visualizzare la dissimilarità tra oggetti in uno spazio 2D o 3D. Il Multi Dimensional Scaling non è una procedura che permette di riorganizzare gli elementi in un modo esatto, ma permette di trovare una configurazione che approssima nel modo migliore le distanze osservate. Partendo da una matrice quadrata contenente la somiglianza di ogni elemento di riga con ogni elemento di colonna, l'algoritmo MDS è un metodo iterativo che assegna a ogni elemento una posizione in uno spazio K-dimensionale stabilito a priori. In pratica questa tecnica parte con un sistema con tante dimensioni quanti gli elementi del sistema, e riduce le dimensioni fino ad un certo numero k. Nel fare questo c'è un'inevitabile perdita di informazione, e per questo motivo esistono diversi algoritmi di scaling multidimensionale che cercano di minimizzare l'errore. In particolare si distinguono algoritmi metrici e non-metrici. Gli algoritmi metrici è un soprainsieme degli MDS classici, che generalizza la procedura di ottimizzazione di una serie di funzioni di perdita e di matrici di ingresso. Gli algoritmi non-metrici invece trovano sia relazioni monotoniche non parametriche tra le dissimilarità nella matrice, sia le distanze euclidee tra gli elementi e la locazione di ogni elemento nello spazio ("Multidimensional Scaling", n.d.).

R fornisce delle funzioni per il classical (metric) e il nonmetric multi dimensional scaling. Se assumiamo di avere N oggetti (N righe) misurati su p variabili numeriche (p colonne), possiamo misurare la distanza tra gli oggetti in maniera anche visuale (in uno spazio k-dimensionale). Il Nonmetric MDS invece, è eseguito utilizzando la funzione `isoMDS()`, contenuta nel package `MASS` (da aggiungere al progetto R). Se utilizziamo direttamente la funzione `isoMDS`, esce errore, perché tra due oggetti (il numero 15 e il numero 30), c'è distanza nulla o negativa ("R-help archive April 2004", n.d.). Quindi è necessario rimuovere i duplicati prima di calcolare le distanze. Per farlo basta utilizzare la funzione `unique()` su `matrTXT`, la quale mi cancella una delle due righe uguali, facendo quindi diminuire il numero di righe a 39. La stessa cosa si ottiene utilizzando la funzione `cmdscale()` come nel codice qui sotto

```
dMDS2 <- dist(matrTXT)    #di default usa la distanza euclidea tra le righe
#k indica le dimensioni dello spazio
fitMDS2 <- isoMDS(dMDS2,  y=cmdscale(dMDS2,k = 2), k=2) )
fitMDS2                   #visualizza i risultati su terminale R
x2 <- fitMDS2$points[,1]
y2 <- fitMDS2$points[,2]
plot(x2,y2,xlab="Coordinate 1",ylab="Coordinate 2",main="Nonmetric MDS",type="n")
text(x2,y2,labels=row.names(matrTXT),cex=.7)
```

Il grafico è rappresentato in Figura 3.9.

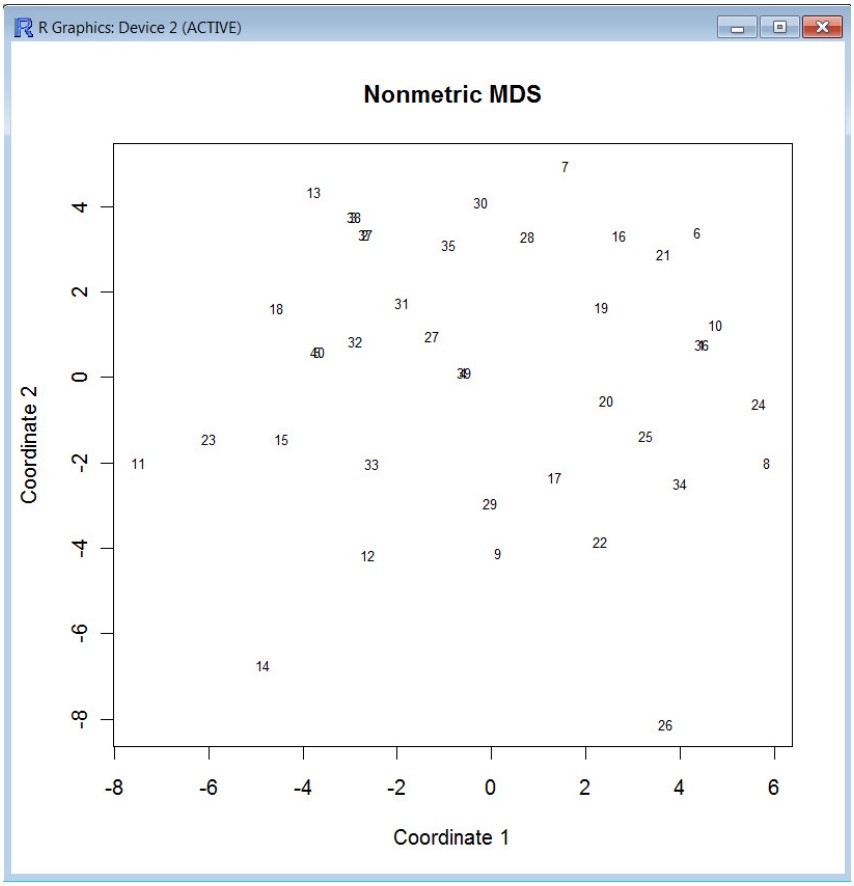



Figura 3.9: Nonmetric Multi Dimensional Scaling

### Individuazione numero di cluster

Certi metodi utilizzati per la cluster analisi, necessitano di un numero di cluster in ingresso per funzionare. Per i problemi di determinazione del numero di cluster da utilizzare, nei quali non c'è una soluzione migliore delle altre, si possono utilizzare diversi approcci. L'approccio qui utilizzato è simile al "*Scree Test*" (*Scree* = ghiaione) dell'analisi fattoriale, che consiste nel disegnare un grafico utilizzando la matrice in ingresso. Se si prende una matrice di correlazione, la si può decomporre in combinazioni indipendenti e pesate della variabili originali. Queste combinazioni corrispondono ai numeri diversi di cluster; ognuno di questi numeri (o fattori) ha una varianza a lui associata: se il fattore è importante avrà una varianza molto elevata. Quindi il grafico consiste nell'ordinare i fattori secondo la loro varianza nel grafico. Tale grafico contiene una curva che sembra l'incontro tra una scogliera e il terreno. Osservando il grafico, dobbiamo decidere dove la scogliera finisce e dove inizia il terreno. Il punto che sceglieremo indicherà, in maniera più o meno corretta, il numero di cluster (il fattore) da utilizzare. Come si vede in Figura 3.10 non è semplice scegliere il punto adeguato; lo stacco sembra essere tra il numero 2 e il numero 8. Nella nostra analisi è stato deciso di provare con 2, 3, e 4 come numero di cluster.

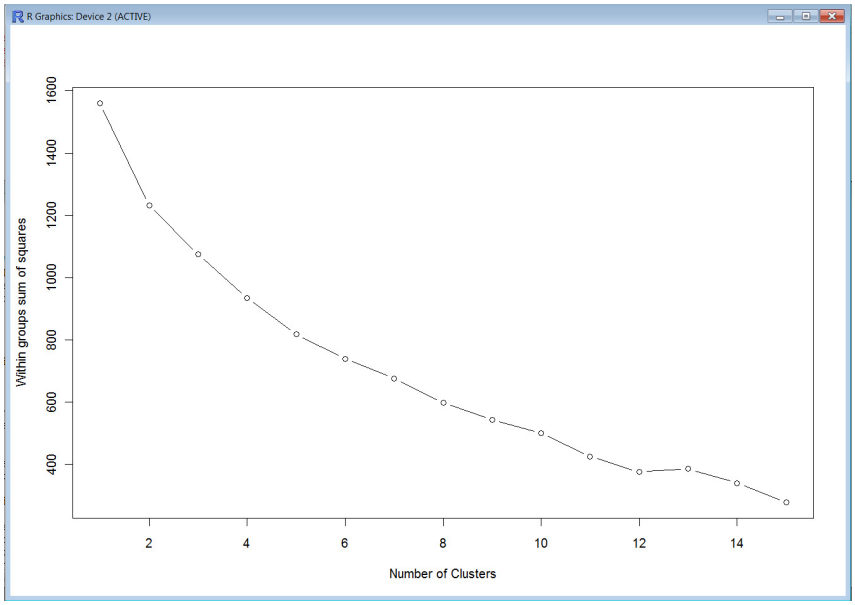



Figura 3.10: Grafico ottenuto dallo Scree Test (Scree Plot)

Il codice R che permette di visualizzare lo Scree Plot è il seguente

```
wss <- (nrow(matrTXT)-1)*sum(apply(matrTXT,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(matrTXT, centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")
```

### K-means clustering

K-means clustering è il metodo di partizionamento più popolare, che permette di suddividere gli oggetti in K partizioni sulla base dei loro attributi. L'obiettivo che l'algoritmo si prepone è di minimizzare la varianza totale intra-cluster. Ogni cluster viene identificato mediante un centroide [8]. L'algoritmo segue una procedura iterativa. Inizialmente prende k punti "di ingresso" (casualmente o usando alcune informazioni euristiche), e crea K partizioni assegnando ogni punto al cluster del punto di ingresso più vicino. Poi calcola il centroide di ogni gruppo, il quale diventerà il nuovo punto di ingresso per calcolare una nuova partizione dei dati, e così via, finché l'algoritmo non converge. L'algoritmo cerca quindi di minimizzare l'errore quadratico medio tra ogni punto del cluster e il suo centroide. Esso richiede in anticipo il numero di cluster da estrarre, per questo motivo vengono utilizzati 2, 3 o 4 cluster, come scelto in precedenza. La riga di codice che permette questo tipo di clustering è la seguente

```
( mKMEANS2<-kmeans(matrTXT, 4, iter.max = 2000) )    #esempio con 4 cluster
```

[8] Il centroide è anche detto punto medio, o baricentro, o centro di massa.

Dove `iter.max` indica il numero massimo di iterazioni permesse, e le parentesi che racchiudono l'intera riga di codice permettono di stampare sul terminale R tutte le informazioni prodotte dal clustering. Attraverso le informazioni stampate a schermo è possibile vedere a che cluster appartengono ognuno dei 40 elementi della matrice (gli spettatori).
Qui di seguito si può vedere il risultato prodotto (ogni numero della seconda riga indica il numero del cluster nel quale può trovarsi lo spettatore scritto sulla prima riga).

Caso con 2 cluster

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 1 |

| 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 |

Caso con 3 cluster

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 2 | 2 | 2 | 3 | 3 | 1 | 1 | 3 | 2 | 2 | 2 | 2 | 2 | 3 | 1 | 2 | 3 | 1 | 3 |

| 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 1 | 1 | 2 | 3 | 1 | 2 | 3 | 3 | 2 | 2 | 3 | 3 | 2 | 1 | 3 | 1 |

Caso con 4 cluster

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 1 | 3 | 4 | 4 | 2 | 1 | 4 | 3 | 1 | 2 | 3 | 3 | 3 |

| 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 4 | 3 | 3 | 1 | 2 | 2 | 1 | 4 | 2 | 2 | 2 | 4 | 3 | 3 | 1 | 3 | 2 | 3 |

Una versione più robusta (al rumore) di K-means, e basata sui medoid, può essere impiegata utilizzando la funzione `pam()` al posto di `kmeans()`. La funzione per utilizzare necessita del package `fpc` , che verrà quindi incluso all'inizio del file. Questo algoritmo è molto simile al K-means, nel senso che entrambi cercano di minimizzare l'errore quadratico medio tra i punti di un cluster e il suo centroide. La differenza sta nel fatto che in K-means il centroide è la pura media di tutti i punti del cluster, mentre in **K-medoids** viene usato il punto collocato più centralmente (in questo modo il centroide è uno dei punti a disposizione). Anche in questo caso vengono utilizzati 2, 3, e 4 cluster, e il codice è il seguente

```
( mKMEANSpam4<-pam(matrTXT, 4) )       #esempio con 4 cluster
```

Anche in questo caso si possono vedere su che cluster cade ogni spettatore.

Caso con 2 cluster

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 1 | 1 | 1 |

| 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 1 |

Caso con 3 cluster

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 2 | 3 | 2 | 3 | 1 | 1 | 1 | 3 | 3 | 2 | 3 | 3 | 2 | 1 | 3 | 2 | 1 | 2 |

| 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 1 | 1 | 1 | 2 | 2 | 1 | 3 | 2 | 2 | 2 | 3 | 2 | 2 | 1 | 1 | 2 | 1 |

Caso con 4 cluster

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 4 | 3 | 3 | 1 | 1 | 3 | 4 | 4 | 2 | 4 | 4 | 2 | 1 | 4 | 3 | 1 | 2 |

| 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 3 | 1 | 1 | 2 | 2 | 1 | 4 | 2 | 2 | 2 | 4 | 3 | 2 | 3 | 1 | 2 | 1 |

### Hierarchical clustering

L'algoritmo di K-means ci da delle partizioni molto semplici e piatte, perché le partizioni non hanno nessuna organizzazione o struttura al loro interno ("Distance Between Partitions", n.d.). Quindi può capitare di volere un clustering gerarchico, che è raffigurato tramite un albero o dendrogramma. Come già visto, ci sono due approcci per questo tipo di clustering: bottom-up (*agglomerative*) e top-down (*divisive*). L'algoritmo di base dell'*agglomerative clustering* è molto semplice:

- ad ogni punto corrisponde un cluster;
- finché non c'è un unico cluster, si trova la coppia di cluster più vicina, e si fondono (*merge*) assieme;
- alla fine si ritorna il grafico rappresentante l'albero delle fusioni.

Ci sono moltissimi approcci per questo tipo di cluster. Uno da utilizzare in R è il **metodo di Ward**, il quale dice che la distanza tra due cluster indica di quanto la somma dei quadrati delle distanze deve aumentare quando si fa il merge, cioè la distanza tra due cluster viene considerata come il costo di merging tra due cluster. All'inizio dell'algoritmo la somma dei quadrati è nulla, perché ogni punto è nel proprio cluster. Nelle iterazioni successive, il metodo di Ward cerca di mantenere l'incremento di questa distanza il più basso possibile. Questo metodo è sia greedy, sia vincolato dalla scelta dei cluster da fondere. Il codice per utilizzare questo metodo in R è il seguente

```
d <- dist(matrTXT, method = "euclidean") # distance matrix
fit1 <- hclust(d, method="ward")
plot(fit1) # display dendogram
```

```
groups <- cutree(fit1, k=4) # cut tree into 4 clusters
# draw dendogram with red borders around the 2, 4, 6 and 8 clusters
rect.hclust(fit1, k=2, border="yellow")
rect.hclust(fit1, k=3, border="red")
rect.hclust(fit1, k=4, border="blue")
rect.hclust(fit1, k=6, border="green")
rect.hclust(fit1, k=8, border="gray")
```

come si vede dal codice è stato scelto di disegnare nel grafico dei riquadri di colore diverso (giallo, rosso, blu, verde e grigio) che indicassero la suddivisione in 2, 3, 4, 6 e 8 cluster. Il dendogramma è rappresentato nella Figura 3.11.

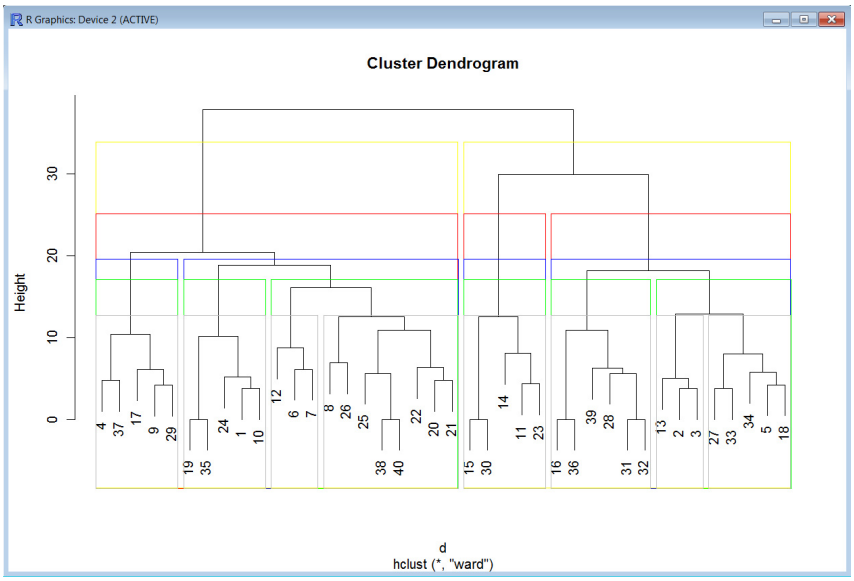



Figura 3.11: Dendogramma dell'*agglomerative clustering*

### 3.6.3 Elaborazione dei risultati

Osservando il grafico dell'analisi MDS fatta in precedenza (Figura 3.9), non si rilevano particolari raggruppamenti di spettatori, e risulta quindi difficile poter dividere l'intero insieme in più di un cluster. Questo potrebbe indicare che molto probabilmente non ci sono marcate differenze tra le risposte date ai quesiti del questionario. Nonostante questo, è utile tentare di visualizzare una suddivisione in cluster (ottenuta con uno dei metodi di clustering) sul grafico MDS.
Vengono ora riscritte le suddivisioni in cluster trovate con i metodi K-medoids e Hierarchical clustering; poi gli elementi del grafico MDS vengono colorati in base a tali suddivisioni: Figura 3.12 per il metodo K-medoids, e Figura 3.13 per il clustering gerarchico.
Suddivisioni in cluster con il metodo K-medoids e 2 cluster:

Cluster1: 1, 4, 6, 7, 8, 9, 10, 16, 17, 19, 20, 21, 22, 24, 25, 26, 28, 29, 35, 36, 38, 40

Cluster2: 2, 3, 5, 11, 12, 13, 14, 15, 18, 23, 27, 30, 31, 32, 33, 34, 37, 39

Suddivisioni in cluster con il metodo K-medoids e 3 cluster:

Cluster1: 1, 8, 9, 10, 17, 20, 22, 24, 25, 26, 29, 37, 38, 40

Cluster2: 2, 3, 4, 6, 13, 16, 19, 21, 27, 28, 31, 32, 33, 35, 36, 39

Cluster3: 5, 7, 11, 12, 14, 15, 18, 23, 30, 34

Suddivisioni in cluster con il metodo K-medoids e 4 cluster:

Cluster1: 1, 8, 9, 17, 20, 22, 25, 26, 29, 38, 40

Cluster2: 2, 3, 13, 16, 21, 27, 28, 31, 32, 33, 36, 39

Cluster3: 4, 6, 7, 10, 19, 24, 35, 37

Cluster4: 5, 11, 12, 14, 15, 18, 23, 30, 34

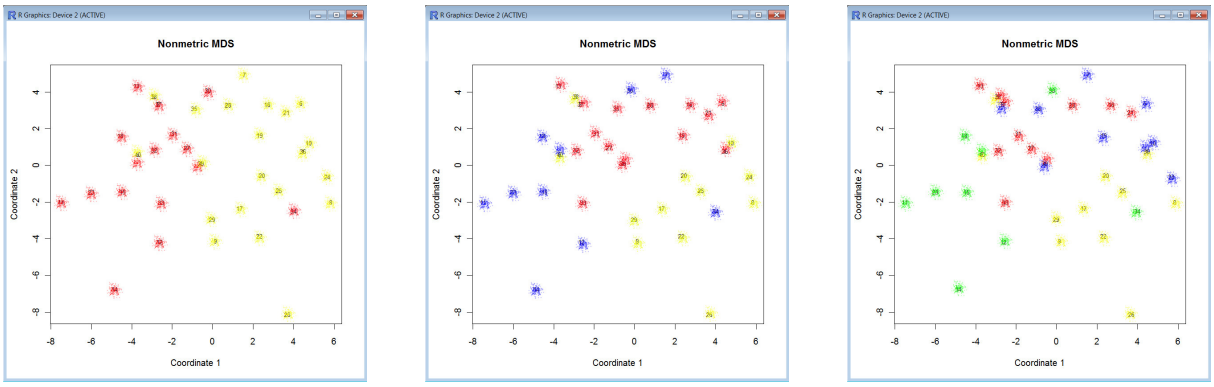



Figura 3.12: Grafico MDS + metodo K-medoids con suddivisione in 2, 3 e 4 cluster

Suddivisioni in cluster con il metodo del clustering gerarchico e 2 cluster:

Cluster1: 1, 4, 6, 7, 8, 9, 10, 12, 17, 19, 20, 21, 22, 24, 25, 26, 29, 35, 37, 38, 40

Cluster2: 2, 3, 5, 11, 13, 14, 15, 16, 18, 23, 27, 28, 30, 31, 32, 33, 34, 36, 39

Suddivisioni in cluster con il metodo del clustering gerarchico e 3 cluster:

Cluster1: 1, 4, 6, 7, 8, 9, 10, 12, 17, 19, 20, 21, 22, 24, 25, 26, 29, 35, 37, 38, 40

Cluster2: 11, 14, 15, 23, 30

Cluster3: 2, 3, 5, 13, 16, 18, 27, 28, 31, 32, 33, 34, 36, 39

Suddivisioni in cluster con il metodo del clustering gerarchico e 4 cluster:

Cluster1: 4, 9, 17, 29, 37

Cluster2: 1, 6, 7, 8, 10, 12, 19, 20, 21, 22, 24, 25, 26, 35, 38, 40

Cluster3: 11, 14, 15, 23, 30

Cluster4: 2, 3, 5, 13, 16, 18, 27, 28, 31, 32, 33, 34, 36, 39

Come si poteva osservare prima di colorare il grafico MDS, i punti sono distribuiti in tale grafico in modo abbastanza omogeneo, e quindi risulterà molto difficile che ci siano più di 3 cluster nell'insieme degli spettatori. Infatti come si può osservare dalla Figura 3.12 e dalla Figura 3.13, la suddivisione degli spettatori in 4 gruppi è difficilmente fattibile, in

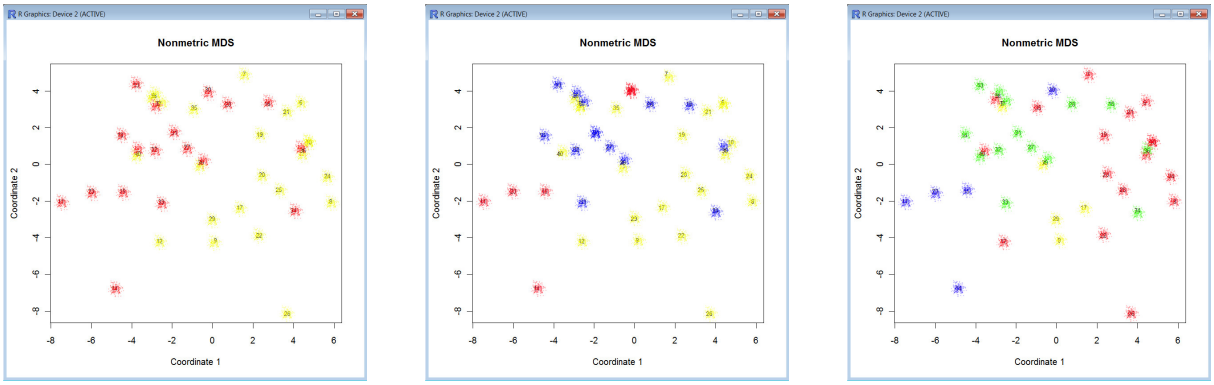



Figura 3.13: Grafico MDS + metodo Hierarchical clustering con suddivisione in 2, 3 e 4 cluster

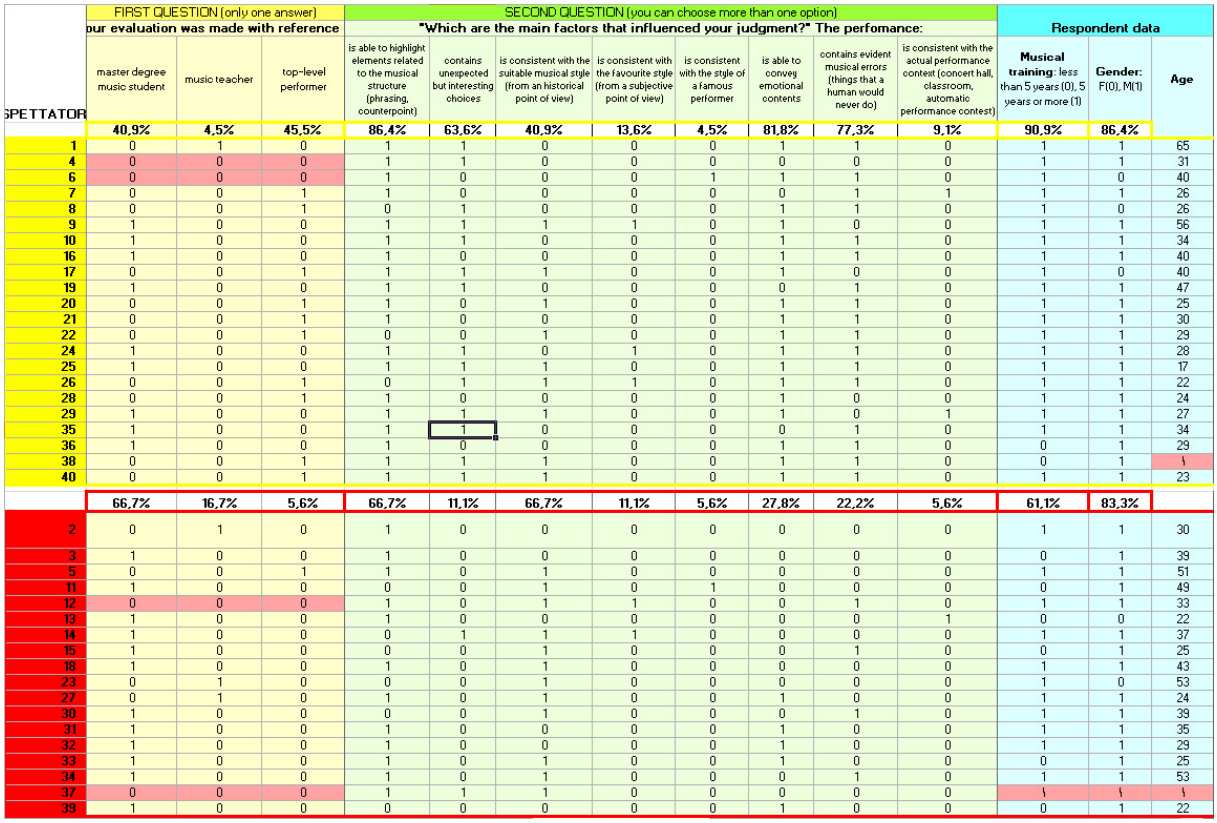
quanto gli spettatori appartenenti a cluster diversi hanno delle posizioni che si mescolano tra di loro nello spazio 2D, e non vanno a creare delle aree nette di colore diverso.

Per quanto riguarda la suddivisione in 2 o 3 cluster, si può osservare che il metodo k-medoids con 2 cluster è quello che evidenzia in maniera più netta due aree di colore diverso. Per questo motivo è stata presa questo tipo di suddivisione per elaborare le risposte date dai 40 spettatori in base al loro cluster di appartenenza. Con Excel sono state raggruppate le righe degli spettatori in base ai due gruppi, e si è cercato di trovare delle differenze nelle risposte date (vedi Figura 3.14).

Innanzitutto osservando gli spettatori appartenenti ai due cluster, si può notare che non c'è nessuna distinzione di sesso fra i vari gruppi, in quanto le donne e gli uomini sono equamente distribuiti. Invece, per quanto riguarda il *musical training*, si può osservare che gli spettatori appartenenti al cluster 1 (di colore giallo nella Figura 3.14) sono un po' più esperti di musica rispetto gli spettatori del cluster 2 (di colore rosso). Per individuare immediatamente questa differenza tra i due gruppi, potremmo etichettare il cluster 1 con "musicisti", e il cluster 2 con "non musicisti".

La seconda risposta alla second question (*contains unexpected but interesting choices*), è stata scelta da 16 spettatori, e la maggioranza di questi sono presenti nel cluster 1. Infatti, se in Figura 3.14 si vanno a osservare le percentuali di scelta della risposta numero due nei due cluster, si può constatare che il gruppo dei "musicisti" è più interessato a delle scelte musicali inaspettate e più interessanti, rispetto al gruppo dei "non musicisti". Dalla risposta numero tre invece (*is consistent with the suitable musical style (from an historical point of view)*), si deduce che i "non musicisti", essendo poco esperti, preferiscono un'esecuzione attinente allo stile della canzone, e che non abbia niente aggiunte particolari al brano. Questa valutazione risulta consistente con le osservazioni fatte per la risposta numero due.

Nel cluster 1 sono presenti anche la maggioranza dei spettatori che hanno scelto la risposta numero 6 (*is able to convey emotional contents*, e la risposta numero 7 (*contains evident*



| SPETTATOR | FIRST QUESTION (only one answer) | | | SECOND QUESTION (you can choose more than one option) | | | | | | | | Respondent data | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Your evaluation was made with reference | | | "Which are the main factors that influenced your judgment?" The perfomance: | | | | | | | | | | |
| | master degree music student | music teacher | top-level performer | is able to highlight elements related to the musical structure (phrasing, counterpoint) | contains unexpected but interesting choices | is consistent with the suitable musical style (from an historical point of view) | is consistent with the favourite style (from a subjective point of view) | is consistent with the style of a famous performer | is able to convey emotional contents | contains evident musical errors (things that a human would never do) | is consistent with the actual performance context (concert hall, classroom, automatic performance contest) | **Musical training**: less than 5 years (0), 5 years or more (1) | **Gender**: F(0), M(1) | **Age** |
| | **40,9%** | **4,5%** | **45,5%** | **86,4%** | **63,6%** | **40,9%** | **13,6%** | **4,5%** | **81,8%** | **77,3%** | **9,1%** | **90,9%** | **86,4%** | |
| **1** | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 65 |
| **4** | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 31 |
| **6** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 40 |
| **7** | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 26 |
| **8** | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 26 |
| **9** | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 56 |
| **10** | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 34 |
| **16** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 40 |
| **17** | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 40 |
| **19** | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 47 |
| **20** | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 25 |
| **21** | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 30 |
| **22** | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 29 |
| **24** | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 28 |
| **25** | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 17 |
| **26** | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 22 |
| **28** | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 24 |
| **29** | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 27 |
| **35** | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 34 |
| **36** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 29 |
| **38** | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | \ |
| **40** | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 23 |
| | **66,7%** | **16,7%** | **5,6%** | **66,7%** | **11,1%** | **66,7%** | **11,1%** | **5,6%** | **27,8%** | **22,2%** | **5,6%** | **61,1%** | **83,3%** | |
| **2** | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 30 |
| **3** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 39 |
| **5** | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 51 |
| **11** | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 49 |
| **12** | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 33 |
| **13** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 22 |
| **14** | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 37 |
| **15** | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 25 |
| **18** | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 43 |
| **23** | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 53 |
| **27** | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 24 |
| **30** | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 39 |
| **31** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 35 |
| **32** | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 29 |
| **33** | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 25 |
| **34** | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 53 |
| **37** | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | \ | \ | \ |
| **39** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 22 |

Figura 3.14: Risposte dei spettatori suddivise per cluster (K-medoids con 2 cluster)

*musical errors (things that a human would never do)*).

La prima opzione della first question (domanda a scelta singola), ci suggerisce una sostanziale differenza tra i due gruppi, che va in qualche modo a giustificare le differenze sulle risposte appena evidenziate. Nel secondo cluster la gran parte degli spettatori, quelli musicalmente meno esperti, hanno valutato le performance musicali facendo riferimento ad un *master degree music student*, mentre le persone che hanno valutato con riferimento a un *top-level performer* sono tutte presenti (tranne una) nel cluster 1 dei "musicisti". Questo sta a indicare che, giustamente, gli spettatori più esperti di musica hanno una scala di valutazione più raffinata rispetto a quelli meno esperti, mentre quelli meno esperti hanno una valutazione più semplice e approssimativa, andando ad osservare gli elementi più semplici da interpretare come la struttura musicale (infatti la risposta numero 1 della domanda a scelta multipla è stata quella scelta di più dal cluster 2).

# Capitolo 4

# Conclusioni

La musica è sempre stata una forza trainante per l'innovazione della tecnologia, e le nuove tecnologie hanno aperto la strada a nuove forme di interazione con la musica e con l'espressione musicale. La definizione di modelli informatici, per rappresentare l'espressività di un'esecuzione musicale, è utile per cercare di capire come e in che modo si possano esprimere delle intenzioni espressive in una performance.
Lo studio delle modalità con cui un computer può comunicare dei contenuti espressivi utilizzando la musica, può diventare molto importante sia nell'insegnamento che nell'educazione, per fornire strumenti di interazione più semplici e intuitivi da utilizzare. Sviluppare un modello per le persone con limitate capacità di movimento, può ad esempio fornire vari modi di accesso alla terapia musicale (musicoterapia), e le interfacce di riconoscimento dei gesti devono essere progettate specificatamente in accordo con le capacità di movimento degli utenti. Per non parlare di tutte le possibili applicazioni sul web che potrebbero avere questi sistemi.
In questo lavoro di tesi sono stati esposti solo alcuni dei risultati prodotti inerenti all'espressività musicale. Questo campo della ricerca è ancora molto giovane, e le possibili strade da percorrere sono molteplici. Nonostante questo, si sono iniziati a vedere i primi risultati positivi, e le occasioni di confronto con altre persone che lavorano in questo campo (ad esempio la conferenza Rencon) incoraggiano ancora di più lo sviluppo di questo tipo di studi.

# Bibliografia


[1] Abeles, H. F. (1973). Development and validation of clarinet performance adjudication scale. *Journal of Research in Music Education*, 21(3), 246-255.

[2] Abeles, H. F. & Porter, S. Y. (1978). The sex-stereotyping of musical instruments. *Journal of Research in Music Education*, 26, 65-75.

[3] Arcos, J. L., de Mántaras, R. L., & Serra, X. (1998). Generating expressive musical performances with SaxEx. *Journal of New Music Research*, 27(3), 194-210.

[4] Ariza, C. (2009). The interrogator as critic: The turing test and the evaluation of generative music systems. *Computer Music Journal*, 33(2), 48-70, Summer 2009.

[5] Baba, T., Hashida, M., Katayose, H. (2011). VirtualPhilharmony: a conducting system focused on a sensation of conducting a real orchestra *In Proc. Of Rencon 2011 (Peer Review), Padova, Italy*. Kwansei Gakuin University.

[6] Borchers, J., Lee, E., & Samminger, W. (2004). Personal orchestra: a real-time audio/video system for interactive conducting. *Multimedia Systems*, 9(5):458-465.

[7] Bresin R. & Battel G. U. (2000). Articulation strategies in expressive piano performance. Analysis of legato, staccato, and repeated notes in performances of the andante movement of Mozart's sonata in G major (K 545). *Journal of New Music Research*, 29(3):211-224.

[8] Bresin R. & Friberg A. (2000). Emotional coloring of computer-controlled music performances. *Computer Music Journal*, 24(4):44-63.

[9] Bresin R. & Widmer G. (2000). Production of staccato articulation in Mozart sonatas played on a grand piano. Preliminary results. *TMH-QPSR, Speech Music and Hearing Quarterly Progress and Status Report*, 2000(4):1-6.

[10] Camurri, A., et al. (2000). EyesWeb: Toward Gesture and Affect Recognition in Interactive Dance and Music Systems. *Computer Music Journal* 24(1):57-69.

[11] Camurri, A., B. Mazzarino, & G. Volpe. (2004). Analysis of Expressive Gesture: The EyesWeb Expressive Gesture Processing Library. In A. Camurri and G. Volpe, eds. *Gesture-Based Communication in Human-Computer Interaction*, LNAI 2915. Vienna: Springer Verlag, pp. 460-467.

[12] Canazza, S., Poli, G. D., Drioli, C., Rodà, A., & Vidolin, A. (2000a). Audio morphing different expressive intentions for multimedia systems. *IEEE MultiMedia*, 7(3), 79-83.

[13] Canazza, S., et al. (2000b). Real-Time Morphing Among Different Expressive Intentions in Audio Playback. *Proceedings of the 2000 International Computer Music Conference*. San Francisco, California: International Computer Music Association, pp. 356-359.

[14] Canazza, S., et al. (2003). Expressive Director: A System for the Real-Time Control of Music Performance Synthesis. *In R. Bresin, ed. Proceedings of the Stockholm Music Acoustics Conference 2003, Volume II*. Stockholm: Speech, Music and Hearing, KTH, pp. 521-524.

[15] Canazza, S., De Poli, G., Drioli, C., Rodà, A., & Vidolin A. (2004). Modeling and control of expressiveness in Music Performance. *Proceedings of IEEE*, (92)4, pp. 686-701.

[16] Canazza, S., Rodà, A., Poli, G. D., Barichello, M., Ganeo, D. (2011). CaRo 2.0: A system for social active listening and expressive performing of music content. *In Proc. Of Rencon 2011 (Peer Review)*, Padova, Italy.

[17] Cook, J. (1994). Agent Reflection in an Intelligent Learning Environment Architecture for Composition. In M. Smith, A. Smaill and G. A. Wiggins, eds. *Music Education: An Artificial Intelligence Approach*. London: Springer Verlag, pp. 3-23.

[18] Copeland, B. J. (2000). The Turing Test. *Minds and Machines* 10:519-539.

[19] Dahl, S. (2004). Playing the Accent: Comparing Striking Velocity and Timing in an Ostinato Rhythm Performed by Four Drummers. *Acta Acustica united with Acustica* 90(4):762-776.

[20] Davidson, J. (1993). Visual perception of performance manner in the movements of solo musicians. *Psychology of Music*, 21, 103-113.

[21] Delgado, M., Fajardo, W., & Molina-Solana, M. (2010). A state of the art on computational music performance. *Expert Systems with Applications*, 38(2011), 155-160.

[22] De Poli, G. (2004). Methodologies for Expressiveness Modeling of and for Music Performance. *Journal of New Music Research* 33(3):189-202.

[23] Duerksen (1972). Some effects of expectation on evaluation of recorded musical performance. *Journal of Research in Music Education*, 20, 268-272.

[24] Fiske, H. E. (1994). Evaluation of vocal performances: Experimental research evidence. (74-103). In G. Welch & T. Murao (Eds.), *Onchi and Singing Development*, London: David Fulton Publishers.

[25] French, R.M. (2000). The Turing Test: The First 50 Years. *Trends in Cognitive Sciences* 4(3):115-122.

[26] Friberg, A. (1991). Generative rules for music performance: A formal description of a rule system. *Computer Music Journal*, 15(2), 56-71.

[27] Friberg, A. (2005). Director Musices 2.6 Manual. Available online at `www.speech.kth.se/music/performance/download`.

[28] Friberg, A. (2006). pDM: An expressive sequencer with real-time control of the KTH music performance rules movements. *Computer Music Journal*, 30(1), 56-71.

[29] Friberg, A., et al. (1998). Musical Punctuation on the Microlevel: Automatic Identification and Performance of Small Melodic Units. *Journal of New Music Research* 27(3):271-292.

[30] Friberg, A., Colombo, V., Frydén, L., & Sundberg, J. (2000a). Generating musical performances with director musices. *Computer Music Journal*, 24(3), 23-29.

[31] Friberg, A., J. Sundberg, & Frydén, L. (2000b). Music from Motion: Sound Level Envelopes of Tones Expressing Human Locomotion. *Journal of New Music Research* 29(3):199-210.

[32] Friberg, A., & Battel, G. U. (2002). Structural Communication. In R. Parncutt and G. E. McPherson, eds. *The Science and Psychology of Music Performance*. London: Oxford University Press, pp. 199-218.

[33] Friberg, A., & Bresin, R. (2008). Real-time control of music performance. In Polotti, P., & Rocchesso, D. (Eds.), *Sound to Sense - Sense to Sound: A state of the art in Sound and Music Computing* (pp. 279-302). Berlin: Logos Verlag.

[34] Flores, R. G., & Ginsburgh, V. A. (1996). The Queen Elisabeth musical competition: How far is the final ranking? *The Statistician*, 45(1), 97-104.

[35] Gabrielsson, A. (1995). Music and the mind machine. Expressive intention and performance. Berlin: Springer [pp. 35-47].

[36] Ganeo, D. (2011). CaRo 2.0: un modello per l'esecuzione espressiva della musica basato sul perceptual parametric space. *Tesi magistrale in Ingegneria Informatica (discussione 24 ottobre 2011)*. Dipartimento di Ingegneria dell'Informazione. Università degli Studi di Padova (Italy).

[37] Grachten, M., Goebl, W., Flossmann, S., & Widmer, G. (2009). Phase-plane representation and visualization of gestural structure in expressive timing. *Journal New Music Research*, 38(2), 183-195.

[38] Halpern, M. (2006). The Trouble with the Turing Test. *The New Atlantis* 11:42-63.

[39] Harnad, S. (2000). Minds, Machines and Turing. *Journal of Logic, Language and Information* 9(4):425-445.

[40] Holland, S. (2000). Artificial Intelligence in Music Education: A Critical Review. In E. R. Miranda, ed. *Readings in Music and Artificial Intelligence*. Amsterdam: Harwood Academic Publishers, pp. 239-274.

[41] Hunt, A., R. Kirk, & M. Neighbour (2004). Multiple Media Interfaces for Music Therapy. *IEEE Multimedia* 11(3):5058.

[42] Hunter & Russ (1996). Peer assessment in performacne studies. *British Journal of Music Education*, 13, 67-78.

[43] Ilmonen, T. (2000). The virtual orchestra performance. *In Proceedings of the CHI 2000 Conference on Human Factors in Computing Systems*, Haag, Netherlands, pages 203-204. Springer Verlag.

[44] Ishikawa, O., Aono, Y., Katayose, H., & Inokuchi, S. (2000). Extraction of musical performance rules using a modified algorithm of multiple regression analysis. *In Proceedings of the 2000 international computer music conference*, San Francisco (pp. 348-351).

[45] Jefferson, G. (1949).The Mind of Mechanical Man. *British Medical Journal* 1:1105-1110.

[46] Juslin, P. N. (2001). Communicating emotion in music performance: A review and a theoretical framework. In P. N. Juslin and J. A. Sloboda, editors, *Music and emotion: Theory and research*, pages 305-333. Oxford University Press, New York, 2001.

[47] Juslin, P. N., Friberg, A., & Bresin, R. (2002). Toward a Computational Model of Expression in Performance: The GERM Model. *Musicae Scientiae Special Issue* 2001-2002:63-122.

[48] Landy, F. J., & Farr, J. L. (1980). Performance rating. *Psychological Bulletin*, 87(1), 77-107.

[49] Lee, E., Nakra, T. M., & Borchers, J. (2004). You're the conductor: Arealistic interactive conducting system for children. *In Proc. of NIME 2004*, pages 68-73.

[50] Lehman, P. R. (1968). *Tests and Measurements in Music*. Prentice Hall, Inc. Englewood Cliffs, New Jersey.

[51] Lidral, K. A. (1997). The solo contest for jazz improvisors. *Jazz Educators Journal*, 29(6), 40-43.

[52] M. Mancini, R. Bresin, & C. Pelachaud (2007). A virtual head driven by music expressivity. *IEEE Transactions on Audio, Speech and Language Processing*, 15(6):1833-1841.

[53] Marrin Nakra, T. (2000). Inside the conductor's jacket: analysis, interpretation and musical synthesis of expressive gesture. *PhD thesis*, MIT, 2000.

[54] Mathews, M. V. (1989). The conductor program and the mechanical baton. *Current Directions in Computer Music Research*, Mathews M. and J. Pierce J. editors, pages 263-282. The MIT Press, Cambridge, Mass, 1989.

[55] Mills, J. (1991). Assessing musical performance musically. *Educational Studies*, 17(2), 173-181.

[56] McPherson, G. E. (1993). Factors and abilities influencing the development of visual, aural and creative performance skills in music and their educational implications. Doctor of Philosophy, University of Sydneey, Australia. Dissertation Abstracts International, 54/04-A, 1227. (University Microfilms No. 9317278).

[57] McPherson, G. E. (1995a). Redefining the teaching of musical performance. *The Quarterly Journal of Music Teaching and Learning*, VI(2), 56-64.

[58] McPherson, G. E. (1995b). The assessment of musical performance: Development and validation of five new measures. *Psychology of Music*, 23, 142-161.

[59] McPherson, G. E. (1996). Factors influencing the assessment of musical performance. Unpublished paper presented at the *22nd World Conference of the International Society for Music Education*, Amsterdam, Holland, July, 1996.

[60] McPherson, G. E., & Thompson, W. F. (1998). Assessing Music Performance: Issues and Influences. *Journal of Research in Music Education*, 10(1), 12-24.

[61] Pearce, M., Meredith, D., & Wiggins, G. (2002). Motivations and Methodologies for Automation of the Compositional Process. *Musicae Scientiae* 6(2):119-147.

[62] Pearce, M., & Wiggins, G. (2001). Towards a Framework for the Evaluation of Machine Compositions. *Proceedings of the AISB01 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*. Brighton, UK: SSAISB, pp. 22-32.

[63] Pena, W. M., & Parshall, S. A. (1987). *Problem Seeking: An Architectural Programming Primer*, 3rd ed. Washington, DC: AIA Press.

[64] Peretz, I. (2001). Listen to the brain: a biological perspective on musical emotions. In P. N. Juslin and J. A. Sloboda, editors, *Music and emotion: Theory and research*, pages 105-134. Oxford University Press, New York.

[65] In Polotti, P., & Rocchesso, D., eds. (2008). *Sound to Sense - Sense to Sound: A state of the art in Sound and Music Computing*. Berlin: Logos Verlag.

[66] Puckette, M. (1996). Pure Data. *Proceedings of the 1996 International Computer Music Conference*. San Francisco, California: International Computer Music Association, pp. 269-272.

[67] Radocy (1976). Effects of authority figure biases on chaning judgments of musical events. *Journal of Research in Music Education*, 24, 119-128.

[68] Rigg, M. G. (1964). The mood effects of music: A comparison of data from former investigators. *Journal of Psychology*, 58, 427-438.

[69] Rinman, M. L., Friberg, A., Bendiksen, B., Cirotteau, D., Dahl, S., Kjellmo, I., Mazzarino, B., & Camurri, A. (2004). Ghost in the cave - an interactive collaborative game using non-verbal communication. In A. Camurri and G. Volpe, editors, *Gesture-based Communication in Human-Computer Interaction*, LNAI 2915, volume LNAI 2915, pages 549-556, Berlin Heidelberg. Springer-Verlag.

[70] Russell, J. A. (1980). A Circumplex Model of Affect. *Journal of Personality and Social Psychology* 39:1161-1178.

[71] Saygin, A. P., Cicekli, I., & Akman, V. (2000). Turing Test: 50 Years Later. *Minds and Machines* 10(4):463-518.

[72] Searle, J. R. (1980). Minds, Brains, and Programs. *Behavioral and Brain Sciences* 3(3):417-457.

[73] Sloboda, J. A., & Juslin, P. N. (2001). Psychological Perspectives on Music and Emotion. In P. N. Juslin and J. A. Sloboda, eds. *Music and Emotion: Theory and Research*. London: Oxford University Press, pp. 71-104.

[74] Sundberg, J., Askenfelt, A., & Frydén, L. (1983). Musical Performance: A Synthesis-by-Rule Approach. *Computer Music Journal* 7:37-43.

[75] Sundberg, J. (1993). How Can Music Be Expressive? *Speech Communication* 13:239-253.

[76] Todd, N. P. (1992). The dynamics of dynamics: A model of musical expression. *Journal of the Acoustical Society of America*, 91(6), 3540-3550.

[77] Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind* 59:433-460.

[78] Usa, S., & Mochida, Y. (1998). A Multi-modal Conducting Simulator. *In Proc. Int. Computer Music Conf. (ICMC'98)*, pp. 25-32.

[79] Wanderley, M., & Battier, M., eds. (2000). *Trends in Gestural Control of Music*. Paris: IRCAM.

[80] Wanderley, M. M., et al. (2005). The Musical Significance of Clarinetists' Ancillary Gestures: An Exploration of the Field. *Journal of New Music Research* 34:97-113.

[81] Wapnick, J., Flowers, P., Alegant, M., & Jasinskas, L. (1993). Consistency in piano performance evaluations. Journal of Research in Music Education, 41(4), 282-292.

[82] Whybrew, W. E. (1971). *Measurement and Evaluation in Music*. Wm. C. Brown Company Publishers Dubuque, Iowa.

[83] Widmer, G. (2003). Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146(2), 129-148.

[84] Widmer, G., & Goebl, W. (2004). Computational models of expressive music performance: The state of the art. *Journal of New Music Research*, 33(3), 203-216.

[85] Wiggins, G., & Smaill, A. (2000). Musical Knowledge: What Can Artificial Intelligence Bring to the Musician. In E. R. Miranda, ed. *Readings in Music and Artificial Intelligence*. Amsterdam: Harwood Academic Publishers, pp. 29-46.

[86] Xenakis, I. (1985). Music Composition Treks. In C. Roads, ed. *Composers and the Computer*. Los Altos, California: William Kaufmann.

[87] Zanon, P., & Poli, G. D. (2003). Estimation of parameters in rule systems for expressive rendering in musical performance. *Computer Music Journal*, 27(1), 29-46.

**Siti Web**

La lezione di chitarra (n.d), da http://www.lalezionedichitarra.it/lezioni/teoria/gli-accenti/1088 (ultimo accesso 25 luglio 2011)

MusicArrengers (n.d.), da http://www.musicarrangers.com/star-theory/t08.htm (ultimo accesso 25 luglio 2011)

Tecnica pianistica: il pedale (n.d.), da http://www.klavier.it/tecnica/pedale.htm (ultimo accesso 25 luglio 2011)

MIDI Messages (n.d.), da http://www.midi.org/techspecs/midimessages.php (ultimo accesso 1 agosto 2011)

R Data Import/Export (n.d.), da http://cran.r-project.org/doc/manuals/R-data.html (ultimo accesso 19 settembre 2011)

Statistica con R: Cluster Analysis in R (n.d.), da http://statisticaconr.blogspot.com/2010/06/cluster-analysis-in-r-1-hierarchical.html (ultimo accesso 19 settembre 2011)

Quick-R: Cluster Analysis (n.d.), dahttp://www.statmethods.net/advstats/cluster.html (ultimo accesso 19 settembre 2011)

Multidimensional Scaling (n.d.), da http://digilander.libero.it/metodiemodelli/MDS/MDS1.htm (ultimo accesso 19 settembre 2011)

R-help archive April 2004 (n.d.), da http://tolstoy.newcastle.edu.au/R/help/04/04/0530.html (ultimo accesso 19 settembre 2011)

Distance Between Partitions (n.d.), da http://www.stat.cmu.edu/~cshalizi/350/lectures/08/lecture-08.pdf (ultimo accesso 19 settembre 2011)

NIME: New Interface for Musical Expression (n.d.), http://www.nime.org/ (ultimo accesso 7 ottobre 2011)

NIME I 2001 (n.d.), http://www.nime.org/2001/ (ultimo accesso 7 ottobre 2011)

NIME I 2001: Whorkshop Proposal (n.d.), http://www.nime.org/2001/docs/proposal.pdf (ultimo accesso 7 ottobre 2011)

NIME Oslo 2011 (n.d.), http://www.nime2011.org/nime/ (ultimo accesso 7 ottobre 2011)

SMC 2011 - Sound and Music Computing Conference (n.d), http://smc2011.smcnetwork.org/rencon.htm (ultimo accesso 10 ottobre 2011)

SMC Rencon 2011 - Music Performance Rendering Contest for Computer Systems (n.d), http://smc2011.renconmusic.org/ (ultimo accesso 10 ottobre 2011)