UNIVERSITA'DEGLI STUDI DI MILANO

FACOLTA' DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Dipartimento di Informatica e Comunicazione

Corso di Laurea in Scienze e Tecnologie della Comunicazione Musicale (cl.26)


# PROTOTIPO MULTIPIATTAFORMA PER IL CONTROLLO DEI PARAMETRI DELLA SINTESI FM

Relatore: Prof. Luca Andrea LUDOVICO

Correlatore: Dott. Adriano BARATÈ

Tesi di:

Giuseppe Alessandro LOTÀ

Matricola: 738921

Anno Accademico 2011/2012

# Indice

# 0. Introduzione

Spesso nell'ambito della computer music e del video making, per la composizione dei diversi generi musicali, dalla classica al pop, dalla techno allo ambient sperimentale, passando anche per il cinema e la televisione, vi è l'esigenza di trovare e inserire dei suoni particolari spesso caratterizzati da spettri e ampiezze che variano nel tempo.

Tra i numerosi software di sintesi audio disponibili sul mercato emerge "Csound" che è anche il linguaggio di programmazione utilizzato nel medesimo software.

Csound si differenzia per la varietà di sintesi e di strumenti utente disponibili nonché per la presenza di un sistema di elaborazione del segnale e per la qualità del suono generato.

La tipologia di sintesi che è stata approfondita in questo elaborato è la sintesi "in modulazione di frequenza" (FM).

Csound, ha una grande potenzialità di creare suoni e forme d'onda complesse, non ottenibili con le altre tipologie di sintesi e perviene a buoni risultati nella creazione di suoni armonici e non (quali suoni di tamburi o campane).

Di contro questa tecnica comporta una scarsa prevedibilità del timbro ottenuto da parte dell'utente che necessita di una notevole familiarità con i controlli e una conoscenza teorica approfondita per poter raggiungere i risultati da lui sperati.

# 1. Tecnologie utilizzate

## 1.1 Il linguaggio C#

Il software è stato realizzato in C#[3] che è un linguaggio di programmazione ad alto livello, sviluppato da Microsoft, orientato agli oggetti, con la capacità di usare anche:

a. un approccio funzionale ossia il paradigma di programmazione in cui il flusso di esecuzione del programma assume la forma di una serie di valutazioni di funzioni matematiche;

b. event-driven: nel quale il flusso del programma è largamente determinato dal verificarsi di eventi esterni;

c. generico: in cui è prevista la creazione di costrutti di programmazione che possano essere utilizzati con tipi di dati diversi;

d. imperativo: in cui il programma viene inteso come un'insieme di istruzioni ciascuna delle quali può essere pensata come un ordine.

Inoltre è a tipizzazione statica cioè le cui variabili nascono e muoiono rimanendo dello stesso tipo.

Oltre ad essere statica la tipizzazione è anche "forte" ponendo dei vincoli sull'uso delle variabili in modo che siano consentite solo le operazioni legali su ciascuna variabile.

Dato che implementa il framework .Net dispone di un efficiente meccanismo di garbage collection ovvero di gestione della memoria e di gestione delle eccezioni avanzata.

Permette l'ereditarietà singola da classi ma è possibile ereditare da un numero infinito di interfacce.

L'ereditarietà è la possibilità, per una classe detta "classe derivata", di ereditare da un'altra (la classe base) le variabili, i metodi, le proprietà e di estendere il concetto della classe base e specializzarlo.

Può facilmente cooperare con altri linguaggi .Net oriented.

Altre caratteristiche di C#sharp sono **l'incapsulamento** con il quale esiste una sorta di confine attorno ad ogni oggetto che separa il suo comportamento esterno dalla implementazione interna ed il

**polimorfismo** che permette di creare metodi e proprietà con lo stesso nome il cui comportamento è ridefinibile in classi derivate.
Supporta la generazione di documentazione XML e tramite il progetto Mono è compatibile su piattaforme Linux e Mac.

## 1.2 Progetto Mono

Mono[1] è una piattaforma software per lo sviluppo di applicazioni, e in particolare rappresenta un'implementazione open source del Framework Microsoft .NET, basata sugli standard ECMA.
Mono include diversi componenti, tra cui:

- un compilatore C#, compatibile con tutte le versioni (fino alla 4.0) di C#, conforme alle specifiche ECMA;
- Mono runtime, un runtime che implementa il CLI (common language infrastructure) specificato dagli standard ECMA;
- Libreria di classi di base, compatibili con le classi del framework .NET di Microsoft;
- Libreria di classi di Mono, che aggiungono delle funzionalità alle funzionalità offerte dalle classi Microsoft, come quelle di supporto a GTK#.

Mono permette quindi di godere di tutti i benefici dell'utilizzo del framework .NET, come la popolarità e la gestione automatica della memoria, con il vantaggio di costituire un ambiente di sviluppo multipiattaforma: Mono funziona su Windows, Mac OSX, Linux e molti altri OS.
Inoltre il CLR (common language runtime) di Mono permette di scegliere tra diversi linguaggi di programmazione.
GTK# è una libreria che permette di creare applicazioni GNOME e interfacce grafiche usando Mono, compatibili con Windows, MacOSX e Linux.
GTK# su Windows permette di creare applicazioni graficamente simili alle applicazioni Windows classiche, mentre il suo utilizzo su MAC richiede il server X11 (un supporto alle interfacce utente grafiche).

### 1.3 Windows Forms Application

Windows Forms[2] è il nome dato all'interfaccia grafica di programmazione di applicazioni client (API).

Una applicazione Windows Forms è costituita da una o più finestre denominate "Form", che possono essere finestre di primo livello, finestre figlio o finestre di dialogo.

Una applicazione può supportare molti Form di diverso tipo e in un Form vengono inseriti controlli, per esempio pulsanti e caselle di riepilogo che permettono di definire l'interfaccia utente grafica del programma.

Microsoft Visual Studio .NET supporta la modalità di creazione GUI (Graphical user interface) tramite finestre di progettazione che visualizzano sullo schermo un Form e permettono di inserirvi componenti trascinandoli da una casella degli strumenti.

Le proprietà dei Form e dei controlli, per esempio il colore e il testo, vengono impostate tramite un editor delle proprietà, che semplifica inoltre l'aggiunta di gestori eventi.

All'interno della Casella degli strumenti di Visual Studio (Toolbox in inglese), i controlli sono divisi nelle seguenti categorie:

- *All Windows Forms*: che visualizza in un'unica scheda tutti i controlli disponibili;
- *Common Controls*, che contiene gli oggetti tipicamente presenti in ogni finestra (Label, TextBox, Button, ListBox, ComboBox, etc.);
- *Containers*, che raggruppa i controlli che permettono di gestire il layout del form, ovvero la disposizione degli oggetti;
- *menù & Toolbars*, contenente oggetti che permettono di aggiungere barre dei menù, barre degli strumenti, barra di stato e menù contestuali all'applicazione;
- *Data*, che contiene gli strumenti che permettono di lavorare con fonti di dati esterne (come i database);
- *Components*, che raggruppa oggetti i quali consentono di interagire con il sistema operativo (per gestire le Active Directory ed il registro eventi di Windows, monitorare le modifiche al file system, etc.);

- *Printing*, che contiene gli oggetti necessari per aggiungere le funzionalità di stampa all'applicazione;
- *Dialogs*, contenente i controlli che consentono di visualizzare le finestre di dialogo comuni di Windows, come Apri e Salva con nome.

### 1.4 Csound

Come detto precedentemente, Csound[4] è un linguaggio di programmazione timbrica e un software per la computer music che appartiene alla famiglia dei Music N.

Music N è una sigla, coniata a posteriori, con cui comunemente si indica un'insieme di linguaggi per la computer music sviluppati, complessivamente, nell'arco di circa quarant'anni. Benché realizzati da persone e in contesti differenti, i linguaggi in questione sono accomunati da alcune caratteristiche che hanno portato, appunto, a parlare di un'unica famiglia di appartenenza.

La prima versione di Csound è stata realizzata verso la metà degli anni Ottanta; nel corso degli anni successivi è stato sottoposto a numerose modifiche che ne hanno migliorato le prestazioni allargando i suoi campi di applicazione.

Csound è utilizzato ancora oggi, distribuito in forma gratuita e supportato da una vasta comunità di riferimento.

La versione attuale è la 5.18.0.

L'estensione di un file Csound è .csd che è diviso in sezioni da alcuni tag.

Le due sezioni più importanti sono la Orchestra e la Score.

- Orchestra: è la sezione composta da una intestazione detta header che descrive le informazioni di base degli strumenti ossia le macchine virtuali che si vogliono costruire;
- Score: è la sezione composta dai due tipi di istruzioni: funzioni e note. **Le funzioni** servono a creare le forme d'onda di cui possiamo scegliere le caratteristiche. **Le note** sono i semplici eventi sonori.

I due elementi sintattici principali di Csound sono le **variabili** e gli **opcode** .

I primi sono dei contenitori astratti dove vengono depositati i risultati delle operazioni che li seguono che sono gli opcode che elaborano determinati argomenti.

### 1.5 Sintassi Csound e opcode utilizzati

Un documento Csound è diviso in sezioni da alcuni tag simili a quelli usati nel linguaggio HTML proprio per poter incorporare suoni proprio nelle pagine internet.

Il tag `<CsoundSynthesizer>` identifica un documento Csound e viene aperto all'inizio del file e chiuso alla fine.

Il tag `<CsOptions>` situato subito dopo il tag di apertura `<CsoundSynthesizer>` identifica la sezione dei flags ossia dei dati numerici o letterali opzionali preceduti da un segno –(meno).

I flag più importanti:

-o fnam dove fnam è il nome del file audio da generare

-A genera un file audio in formato AIFF

–W genera un file audio in formato Wave

-m N livello dei messaggi visualizzati durante la sintesi. È la somma di: 1=ampiezza delle note; 2=messaggi di ampiezza fuori gamma; 4=messaggi di avvertimento

-d sopprime la visualizzazione delle forme d'onda generate

-t MM sintetizza il brano con metronomo MM

-H stampa sullo schermo un carattere girevole (heartbeat) durante la sintesi.

-z visualizza sullo schermo la lista degli opcode disponibili

I valori più importanti di default sono:

-o test
-m $7^4$

Dopo il tag di chiusura </CsOptions> vi è la sezione orchestra che è racchiusa tra i tag <CsoundInstruments> e come detto precedentemente è composta dall'Header e dagli strumenti:

Lo header contiene sempre 4 informazioni:

sr frequenza di campionamento audio (sample rate)

kr frequenza di controllo (control rate)

ksmps rapporto frasr e kr (per esempio se sr-48000 c kr=4800 allora ksmps=10); deve essere intero

nchnls (number of channels) numero di canali di uscita (I=mono, 2= stereo etc.)

Gli strumenti che seguono dipendono da queste informazioni. Per esempio, se scriviamo nello header che il numero di canali è 2, non potremo scrivere strumenti quadrifonici in quell'orchestra, ma solo strumenti stereofonici.

Esempio di header:

sr = 48000

kr 4800

ksmps = 10

nchnls = I[4]

Gli strumenti vengono scritti dopo l'Header e sono identificati dall'istruzione *instr* seguita dal numero dello strumento e con l'instruzione *endin* termina lo strumento.

All'interno dei singoli strumenti vi possono essere le variabili che sintatticamente sono composte da: nome della variabile, opcode e argomenti.

Esempio di variabile:

| **variabile** | **opcode** | **argomenti** |
|---|---|---|
| **Variabile1** | oscil | 10000,220, 1 |

L'opcode *out* scrive nel file audio il risultato depositato nella variabile scritta a destra dello stesso ed è l'ultimo opcode di uno strumento.

Gli opcode utilizzati nel presente elaborato sono i seguenti:

alla chiusura del tag <CsoundInstruments> termina la sezione orchestra e viene aperto subito dopo il tag <CsoundScore> con il

quale inizia la Score che come detto in precedenza è formata dalle funzioni e dalle note.
Le funzioni sono composte da 5 parametri fondamentali fissi illustrati nell'esempio di seguito:

| Numero della funzione | Creation time della partitura | Numeri di punti che definiscono la tabella | Metodo di generazione della forma d'onda | Ampiezza della fondamentale |
|---|---|---|---|---|
| f1 | 0 | 4096 | 10 | 1 |

Se viene selezionato la Gen10 (metodo di generazione n°10) dopo l'ampiezza fondamentale si possono scrivere degli altri numeri interi che rappresentano l'ampiezza delle armoniche successive alla fondamentale; il primo numero dopo l'ampiezza della fondamentale in tal modo rappresenta la prima armonica, e il secondo numero la seconda armonica e cosi' via.
Se una armonica ha ampiezza 0 significa che tale componente armonica nel suono risultante non esiste.

Le note invece sono composte da tre parametri fondamentali:

| strumento dell'orchestra che deve suonare | Momento di attacco della nota (in secondi) | Durata della nota (in secondi) |
|---|---|---|
| i1 | 0 | 2.5 |

In caso si volesse cambiare a ogni nota un argomento specifico di un opcode si sostituisce il valore fisso dell'argomento con *p* più il numero dell'argomento che si vuole controllare nello score partendo da 4 e aggiungo un parametro alla nota con cui si controlla il

parametro in questione; ad esempio se si vuole controllare l'argomento frequenza di un opcode *oscil* sostituisco il valore fisso della frequenza con p4 e nello score si aggiungerà un parametro dopo la durata della nota che controllerà la frequenza di quel preciso opcode oscil.
Se si volesse aggiungere un altro parametro da controllare o dello stesso opcode o di un altro occorre sostituire il l'argomento da gestire nello score con p5 e aggiungere un altro parametro nella nota dopo la frequenza dell'opcode oscili.

Esempio di orchestra e score in cui avviene controllo di argomenti dell'orchestra nello score:

```
instr1
sr = 44100
kr = 4410
ksmps = 10
nchnls = 1
```

```
 instr1
asuoni oscil  p4,p5, 1
          out    asuoni
          endin
```

```
f1  0  4096  10  1
;p1          p2          p3          p4          p5
i1           0           2           20000  110
```

Gli opcode utilizzati nell'elaborato sono i seguenti:

1) Oscil

2) Foscili
3) Line
4) Linseg
5) Expon
6) Expseg
7) Linen

Il primo simula un oscillatore e ha come argomenti al suo interno: ampiezza, frequenza e numero di funzione.
Il secondo implementa un oscillatore in FM con una portante e una modulante(versione a interpolazione di foscil) e ha come argomenti: Ampiezza, frequenza nominale, fattore di moltiplicazione della frequenza nominale per ottenere la frequenza portante(in caso la frequenza nominale sia 1 definisce direttamente la frequenza della portante), fattore di moltiplicazione della frequenza nominale per ottenere la frequenza della modulante(in caso la frequenza nominale sia 1 definisce direttamente la frequenza della modulante), indice di modulazione, numero della tabella (pari a uno nel presente elaborato), e fase(opzionale).
Il terzo crea eventi sonori che glissano seguendo una segmento retta da un valore di frequenza o ampiezza iniziale a un valore finale e ha come argomenti: Frequenza o ampiezza iniziale, durata della nota, frequenza o ampiezza finale.
Il quarto è analogo a Line ma permette la definizione di più segmenti retta in cui dopo ogni valore intermedio viene inserita la durata diviso il numero totale dei segmenti.
Il quinto invece crea eventi sonori che glissano seguendo una segmento esponenziale da un valore di frequenza o ampiezza iniziale a un valore finale e ha come argomenti: frequenza o ampiezza iniziale, durata della nota, frequenza o ampiezza finale.
Il sesto e analogo a Linseg ma si riferisce a segmenti di tipo esponenziale, e infine il settimo permette di creare invuppi d’ampiezza trapezoidali e ha come parametri: ampiezza massima

raggiunta dal suono,durata della fase di attacco del suono, durata complessiva della nota(nell'elaborato è pari a p3), durata della fase di estinzione del suono.

Esempi di scrittura degli opcode utilizzati nell'elaborato:

```
amod oscil 10000, 220, 1
aout foscili  icamp, 1, ipk, imk, indx, 1
kglis line 220,p3(durata della nota),440
kenvcar  linseg 40, p3, 23, p3, 67, p3, 34
kenvcar  expon 21, p3, 57
kenvcar  expseg 21, p3, 45, p3, 12, p3, 66
kenv linen 10000, 0.1, p3, 0.5
```

### 1.7 Sintesi del suono FM

La sintesi del suono per modulazione di frequenza ($FM^5$) permette di ottenere un gran numero di timbri complessi in maniera relativamente economica in termini di quantità di calcolo e in semplicità di utilizzo.

Il principio fondamentale consiste nel modulare la frequenza di un oscillatore detto portante con la frequenza di un altro oscillatore detto modulante che definisce la fondamentale del suono da sintetizzare e un indice permette di controllare la profondità della modulazione.

Nella sintesi FM gli oscillatori vengono chiamati operatori, e il primo sintetizzatore commercializzato che implementava questo tipo di sintesi, il Yamaha DX7 aveva la possibilità di controllare 6 operatori ognuno dei quali era un'onda sinusoidale, offrendo la possibilità di combinarli in diversi modi.


**Figura 1 - Yamaha DX7: primo sintetizzatore che implementò la sintesi FM**

Ovviamente con i progressi in ambito tecnologico in numero degli operatori gestibili è stato incrementato sia dai sintetizzatori che dai synth software potendoli combinare in un numero ancora più elevato di combinazioni ma nonostante ciò alcune software house e marchi produttrici di synth sono rimasti fedeli ai 6 operatori.

Se la frequenza dell'operatore portante e la frequenza dell'operatore modulante sono in rapporto intero gli spettri ottenuti sono armonici, altrimenti si ottengono delle famiglie di spettri inarmonici.

Oltre alla configurazione più semplice chiamata "Simple FM" in cui un solo operatore modulante modula un operatore portante è possibile modulare più portanti con una sola modulante nella configurazione chiamata "Parallel Carriers" oppure modulare con più modulanti una portante nella configurazione "Parallel Modulators", in modo da arricchire lo spettro con le componenti prodotte dalla modulazione di ciascuna modulante.

Inoltre ci sarebbe una terza tecnica di modulazione chiamata "Cascade" in cui un operatore modulante modula un altro modulante che a sua volta modula il portante offrendo la possibilità di ottenere spettri irregolari, più ricchi di componenti armoniche di frequenza elevata, ma per motivi tecnici non viene trattata nel presente elaborato.


**Figura 2 - configurazione "Simple FM"**

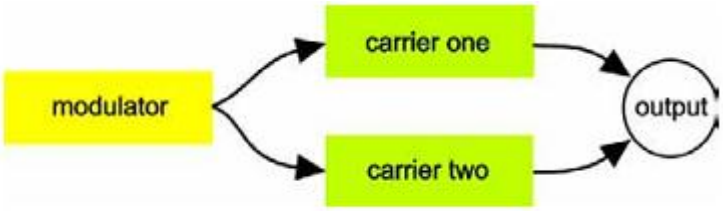


**Figura 3 - configurazione "Parallel Carriers"**

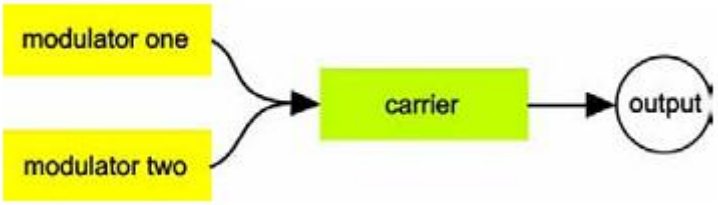


**Figura 4 - Configurazione "Parallel Modulators"**

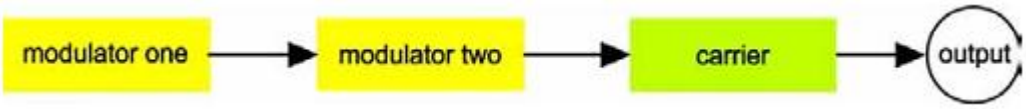


**Figura 5 - configurazione "Cascade"**

# 2. Stato dell'arte

Il mercato offre una vasta gamma di software per la sintesi FM del suono. Sostanzialmente tali software a livello di interfaccia sono

molto simili e si differenziano per la sofisticatezza degli algoritmi adottati ~~onde~~ per generare un suono sempre più realistico.

Tra i più noti marchi si citano: FM8 e Abisinth della statunitense Native Instruments, Toxic bio hazard della belga Image Line, Blue della statunitense Ron Papen.

L'FM8 è il più noto per la qualità di emulazione della sintesi FM originale. Le sue principali caratteristiche sono:

- Emulazione della sintesi FM originale
- Libreria di 256 presets
- Distorsione
- Filtri
- Potente modulazione
- Effetti (stereo chorus, flanger, delay...)
- Audio inputs
- Importazione programmi e suoni di originali synth FM
- Numerose modalità di visualizzazione:(rack, edit, easy edit, keyboard etc)
- Formati: stand-alone, Audio Units, VST, DXi, RTAS (sotto Pro Tools 7), Core Audio, DirectSound, ASIO


**Figura 6 - Esempio di schermata del synth software FM8**

Analogamente ai software succitati, Csound, che svolge le medesime funzioni, ha il principale handicap di disporre di un approccio alfa-numerico ossia di dover compilare le caratteristiche del timbro manualmente secondo la semplice ma rigorosa sintassi di detto

software, anche se nelle versioni più recenti sono stati introdotti degli strumenti che ne facilitano la compilazione.

Tra queste Utility sono da menzionare i Widgets dei componenti grafici di una interfaccia utente che facilitano l'iterazione con il codice come ad esempio knobs e sliders con il quale è possibile controllare i parametri del timbro che si sta creando.

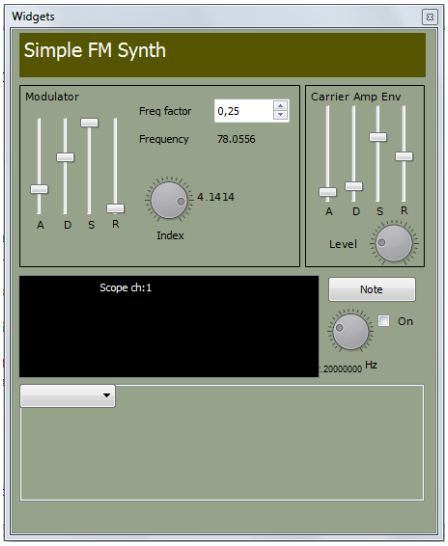


**Figura 7 - Esempio di Widget Csound**

Nella figura 1 vi è un esempio di Widget diviso in n. 3 sezioni:

1. la prima chiamata Modulator gestisce con 4 slider l'inviluppo della frequenza dell'onda modulatrice, con lo knob l'indice di modulazione e con una listbox il fattore di moltiplicazione della frequenza;
2. la seconda chiamata Carrier gestisce con 4 slider l'inviluppo dell'ampiezza dell'onda portante e con uno knob l'ampiezza della stessa;
3. la terza chiamata Note gestisce i singoli eventi sonori.

Inoltre con il tasto destro del mouse si può accedere facilmente alla lista degli **opcode** suddivisi per categorie facilitandone la ricerca e agli elementi sintattici generali come gli Header permettendo

all'utente di non dover ricordare l'esatta sintassi dei vari elementi di Csound.

Nel corso della sua storia Csound è stato affiancato da numerosi altri software o utilità che hanno contribuito e contribuiscono ancora oggi a facilitare l'uso di un linguaggio che per il suo approccio alfanumerico resta comunque piuttosto scomodo. A partire da Cscore, una delle prime utilità sviluppate dallo stesso Vercoe e pensato per facilitare la scrittura del file di partitura, si sono poi avute numerose utilità tra le quali ricordiamo almeno Cecilia, un'interfaccia grafica specifica per Csound, Blue (un ambiente di composizione per Csound basato su Java), Winsound (una versione di Csound per sistemi Windows), Macsound (versione di Csound per utenti Mac), QuteCsound (un ambiente di lavoro con interfaccia grafica) WinXound (un editor per Csound e CosundAV).

# 3. Descrizione dell'applicazione

## 3.1 Obiettivi dell'applicazione

L'obiettivo del software, come anticipato nell'introduzione, è quello di semplificare l'utilizzo di Csound mediante un'interfaccia grafica che richiede all'utente la sola conoscenza dei principi della sintesi FM mentre può ignorare la sintassi di Csound.

Tale semplificazione è stata resa possibile grazie all'introduzione di Widgets grafici come ad esempio gli knob e gli slider che permettono di inserire in maniera più agevole rispetto all'approccio alfanumerico di Csound.

## 3.2 Form principale del programma

All'avvio del programma viene visualizzato il form principale che è costituito da un menuStrip e un pannello di tipo tabpage formato da 2 pagine.

Nel menustrip compaiono le seguenti voci di menu: File, Orchestra, Configura e Inviluppo che di seguito verranno meglio analizzate.

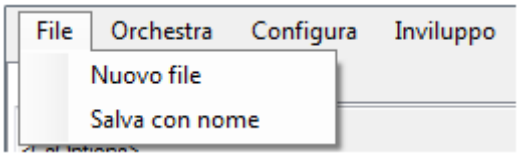


**Figura 8 - menu File**

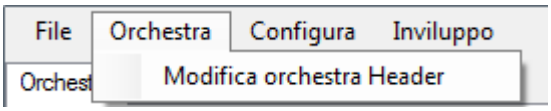


**Figura 9- menu Orchestra**

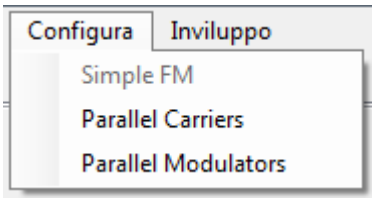


**Figura 10 - menu Configura**

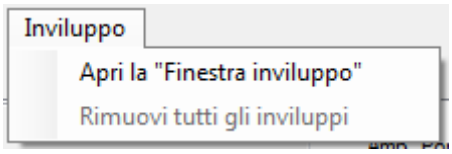


**Figura 11 - menu Inviluppo**

La prima tabpage del form principale chiamata “Orchestra” gestisce i parametri dell’Orchestra e al suo interno sono presenti una textbox multilinea nella parte sinistra nella quale vengono visualizzato il codice Csound relativo all’Orchestra, e nella parte destra i controlli relativi alla configurazione “Simple FM”, ossia tre knobs che controllano: il primo l’ampiezza dell’oscillatore portante, il secondo la frequenza dell’oscillatore portante e il terzo la frequenza dell’oscillatore portante; una trackbar verticale che controlla l’indice di modulazione, e delle textbox sotto ogni controllo che permettono una regolazione più fine dei parametri.

La seconda tabpage chiamata “Score” invece gestisce i parametri dello score e al suo interno sono presenti un’altra textbox multilinea nella parte sinistra che visualizza il codice Csound relativo allo Score, e nella parte destra due groupbox: nella prima ci sono i controlli che gestiscono i parametri della funzione e nella seconda ci sono i controlli che gestiscono i parametri della nota: tali parametri verranno esaminati nel dettaglio nei prossimi paragrafi.

### 3.3 Tabpage “Orchestra”

In questa sezione del form principale vengono visualizzati come affermato in precedenza una textbox multilinea nella parte sinistra della tabpage e di default nella parte destra i controlli relativi alla configurazione “Simple FM”.

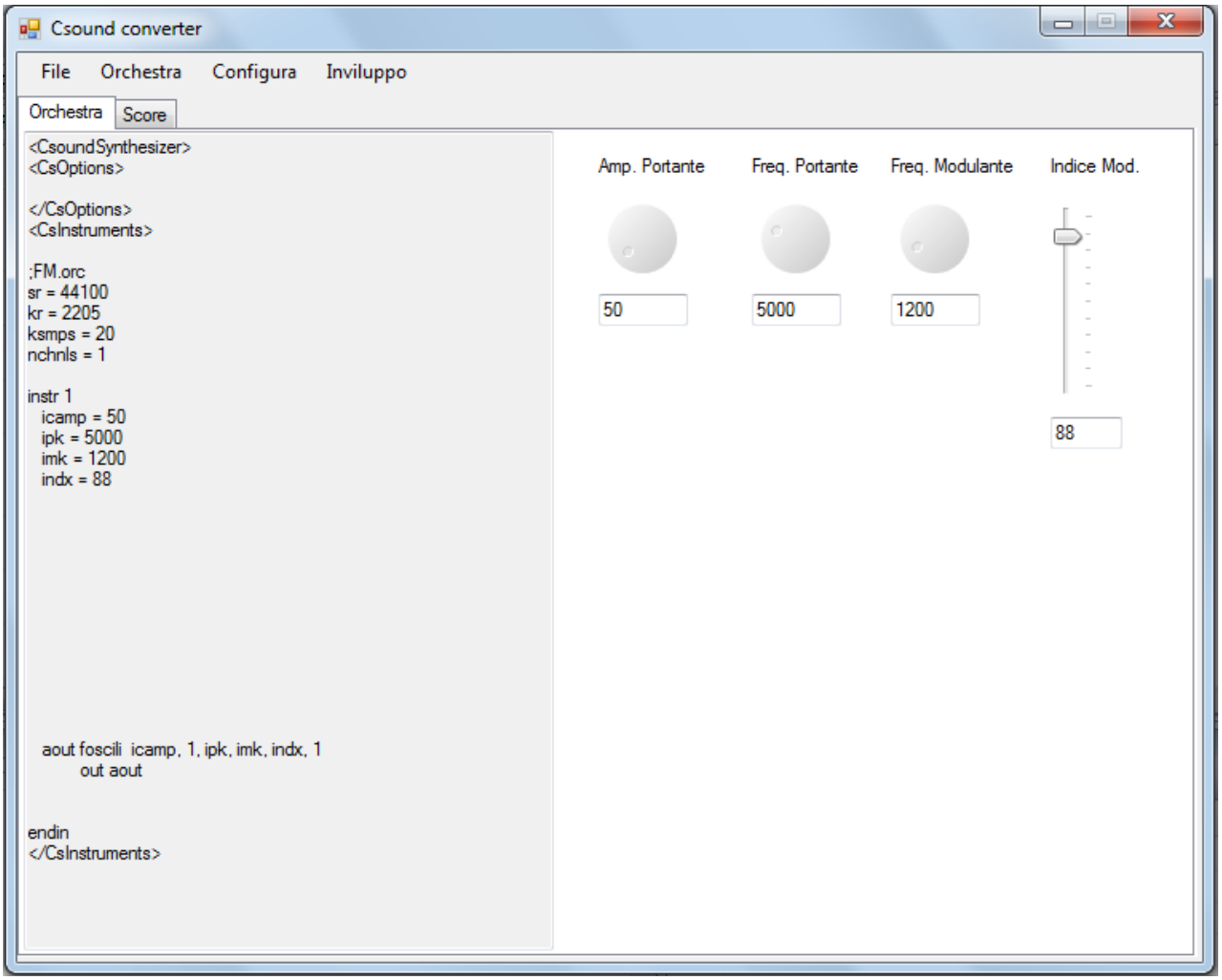


**Figura 12 - Schermata del form principale: tabpage "Orchestra" con parametri di default**

Nella textbox multilinea vengono di visualizzati al caricamento del form principale i tag di Csound e racchiusi dalle istruzioni “Instr1” e “Endin”.tali parametri di default:

- Header con: sr = 44100, kr = 2205, ksmps = 20. nchnls = 1.
- Ampiezza dell’oscillatore portante icamp = 50.
- Frequenza dell’oscillatore portante ipk = 5000
- Frequenza dell’oscillatore modulante imk = 1200

- Indice di modulazione indx = 88.

Dopo tali parametri vengono scritte sempre di default le due istruzioni finali dello strumento 1:

```
aout foscili  icamp, 1, ipk, imk, indx, 1
         out  aout
```

A seconda della scelta optata nel menu “Configura” viene cambiato il numero di controlli presenti nella parte destra della tabpage e di conseguenza vengono aggiunte e modificate alcune istruzioni della textbox mulinea; infatti scegliendo:

1. “Parallel Carriers” nel menu “Configurazione”, vengono aggiunti: uno knob che controlla l’ampiezza del secondo oscillatore portante e un altro knob che ne controlla invece la frequenza, entrambi con una textbox posizionata sotto il controllo.
2. “Parallel Modulators” nel medesimo menu, vengono aggiunti: uno knob che controlla la frequenza del secondo oscillatore modulante e una trackbar che ne controlla l’indice di modulazione, entrambi con una textbox posizionata sotto il controllo.
3. “Simple FM” nel medesimo menu, si ritorna alla configurazione di default eliminando ogni controllo aggiuntivo.

Nel menu “Configura” la voce “Simple FM” al caricamento del form è disabilitata perché già di default è presente tale configurazione, e tale voce viene riabilitata quando viene selezionata qualsiasi altra voce nello stesso menu.

Nella textbox multilinea in caso venga optata la configurazione “Parallel Carriers” vengono aggiunte sotto quelle già esistenti le variabili:

1. icamp2 = 50, che indica l'ampiezza del secondo oscillatore portante.
2. E ipk2 = 2300 che indica la frequenza del secondo oscillatore portante.

Inoltre le istruzioni finali dello strumento 1 diventano:

```
acar1 foscili  icamp, 1, ipk, imk, indx, 1
acar2 foscili  icamp2, 1, ipk2, imk, indx, 1
          out      acar1 + acar2
```

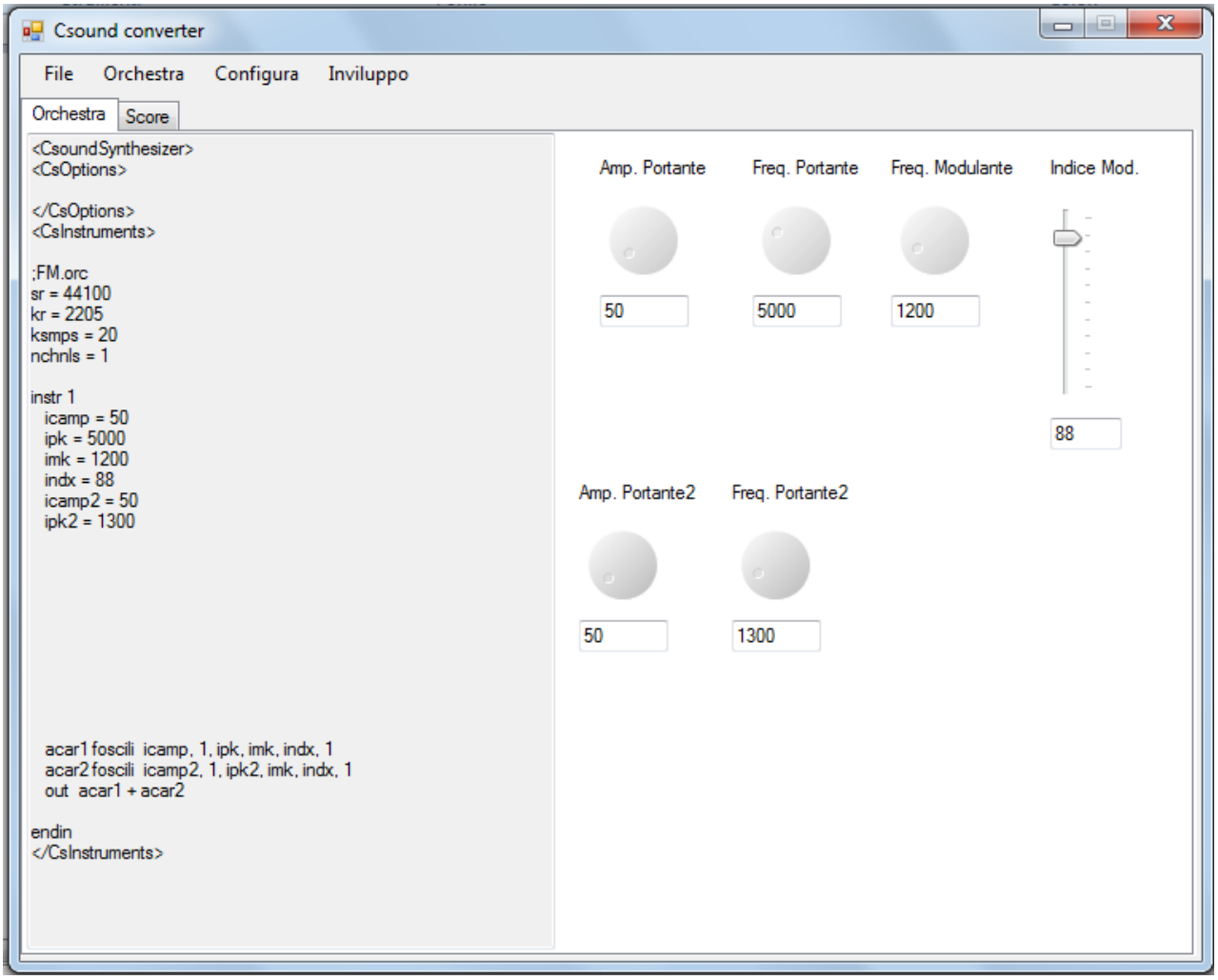


**Figura 13 - tabpage "Orchestra" con configurazione "Parallel Carriers"**

Nel caso invece che venga optata la configurazione "Parallel Modulators" vengono aggiunte sotto quelle già esistenti le variabili:

1. imk2 = 2300 che indica la frequenza del secondo oscillatore modulante.
2. indx2 =77 che indica l'indice di modulazione del secondo oscillatore modulante.

Inoltre le istruzioni finali dello strumento 1 diventano:

```
amod1 oscili  indx*imk, imk, 1
amod2 oscili  indx2*imk2, imk2, 1
acar oscili  icamp, ipk + amod1 + amod2, 1
     out     acar
```

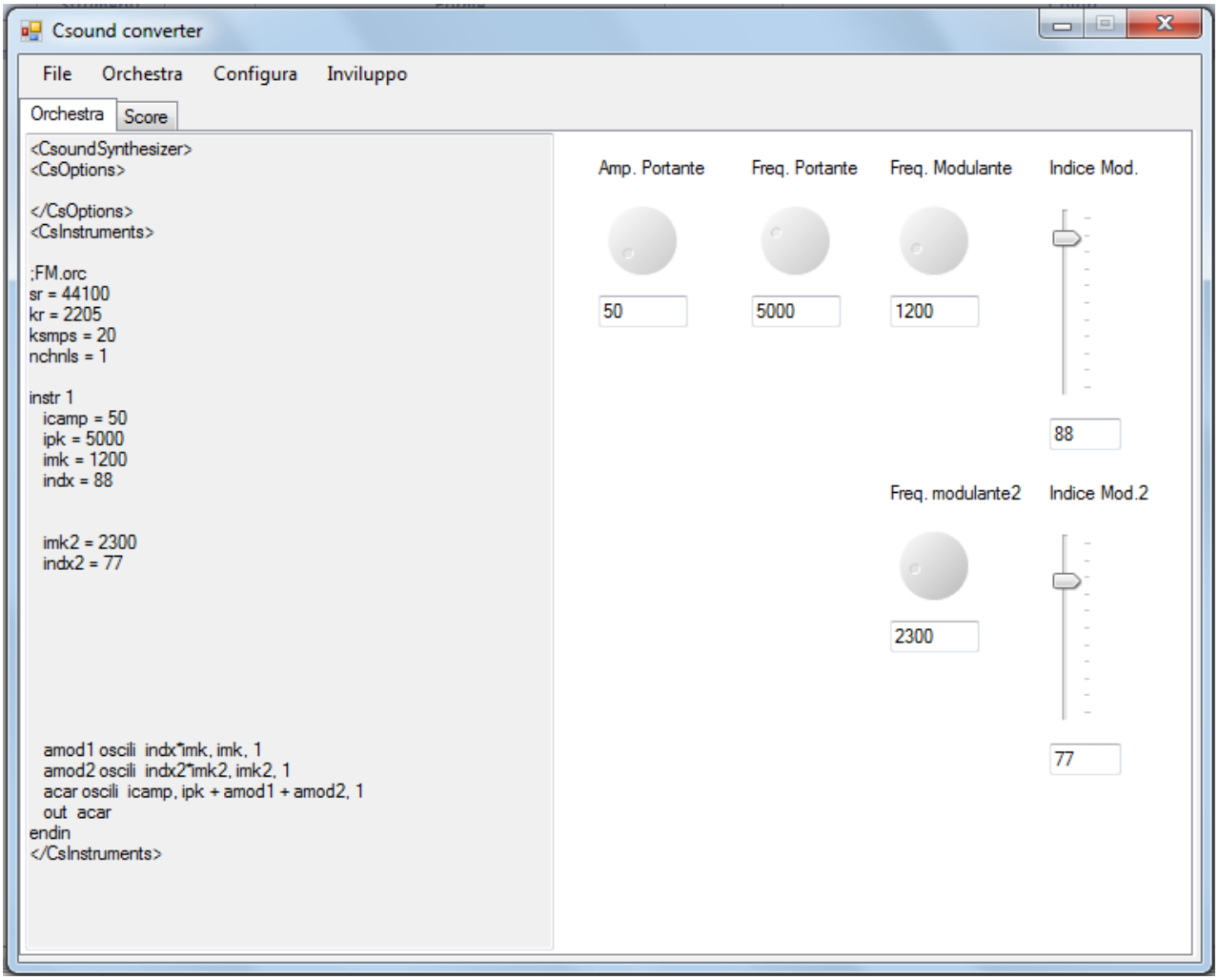


**Figura 14 - tabpage "Orchestra" con configurazione "Parallel Modulators"**

Con il movimento dei controlli viene visualizzato il codice Csound relativo al valore rappresentato dal controllo consentendo all'utente

un riscontro immediato nel codice, e nella textbox sottostante al controllo viene visualizzato il valore relativo del knob o della trackbar mossa.

È possibile inoltre scrivere il valore direttamente nella textbox per avere una regolazione più fine e immediata in caso si sappia già a priori che valore optare.

### 3.4 Inserimento degli inviluppi

Nel menu “Inviluppo” del form principale, la voce “Apri la "Finestra inviluppo" consente di creare un inviluppo dei parametri presenti nella Instruments Section con la visualizzazione di una apposita finestra di dialogo con il medesimo nome, composta da 3 combobox chiamate rispettivamente: parametro di riferimento, tipologia di inviluppo e numero di segmenti.

Nella prima l’utente può selezionare la tipologia di inviluppo scegliendo fra i seguenti parametri:

1. singolo segmento retta;
2. singolo segmenti esponenziale;
3. più segmenti retta
4. più segmenti esponenziale;
5. inviluppo d'ampiezza trapezoidale.

Se vengono selezionate le voci elencate al punto 1 e 2, la seconda combobox, quella relativa al numero di segmenti viene disabilitata e viceversa se vengono selezionate le voci 3 e 4 la combobox viene riabilitata.

A seconda della voce selezionata nella combobox “Tipologia di inviluppo” vengono visualizzati nella parte sottostante della finestra

di dialogo un certo numero di knob con textbox sottostanti che può variare da 2 a 4, ma la condizione necessaria perché vengano visualizzati i controlli è una voce nella combobox “Parametro di riferimento” sia selezionata e nel caso sia abilitata una voce nella combobox “Numero di segmenti”.
Se infatti selezioniamo le seguenti voci nella combobox “Tipologia di inviluppo”:

1. Singolo segmento retta
2. Singolo segmento esponenziale

Verranno visualizzati due knob con annesse textbox, e nella textbox multilinea del form principale nel primo caso verrà visualizzata una stringa contente una variabile chiamata “Kenvcar” con l’opcode Line seguita dai parametri dell’opcode controllati dagli konb presenti e nel secondo caso una stringa contentente la stessa variabile con l’opcode Expon seguita anch’essa dai parametri dell’opcode controllati dagli knob presenti.

Se invece selezioniamo le seguenti voci nella medesima combobox:

1. Più segmenti retta e la voce “2” nella combobox “Numero di segmenti”.
2. Più segmenti esponenziale e la voce “2” nella combobox “Numero di segmenti”.

Anche in questo caso verranno visualizzati tre knob con annesse textbox, e nella textbox multilinea del form principale come in precedenza nel primo caso verrà visualizzata una stringa contente una variabile chiamata “Kenvcar” ma l’opcode presente in questo caso è Linseg nel primo caso e Expseg nel secondo.

E infine se selezioniamo la voce “Inviluppo d’ampiezza trapezoidale” (voce disponibile solo per il parametro “Ampiezza portante”) verranno visualizzati tre knob con annesse textbox e nella textbox multilinea del form principale come in precedenza verrà visualizzata

una stringa contente una variabile chiamata “Kenvcar” contenente l’opcode Linen.

In tutti i casi all’interno delle variabili i parametri di durata che sono parte costituente degli opcode si riferiscono al valore immesso nel controllo numeric-updown (se non cambiato viene preso in considerazione il valore di default) relativo alla durata della nota, che verrà trattato nel capitolo “Tabpage Score”.
Inoltre a seconda della configurazione scelta in precedenza nel form principale vengono aggiunte delle voci nella combobox “Parametro di riferimento”, in particolare se viene scelta la configurazione “Parallel Carriers” vengono aggiunte nella combobox le voci: ”Ampiezza portante2” e “Frequenza portante2”; se viene invece scelta la configurazione “Parallel Modulators” vengono aggiunte le voci: “Frequenza modulante2” e “Indice di modulazione2”.

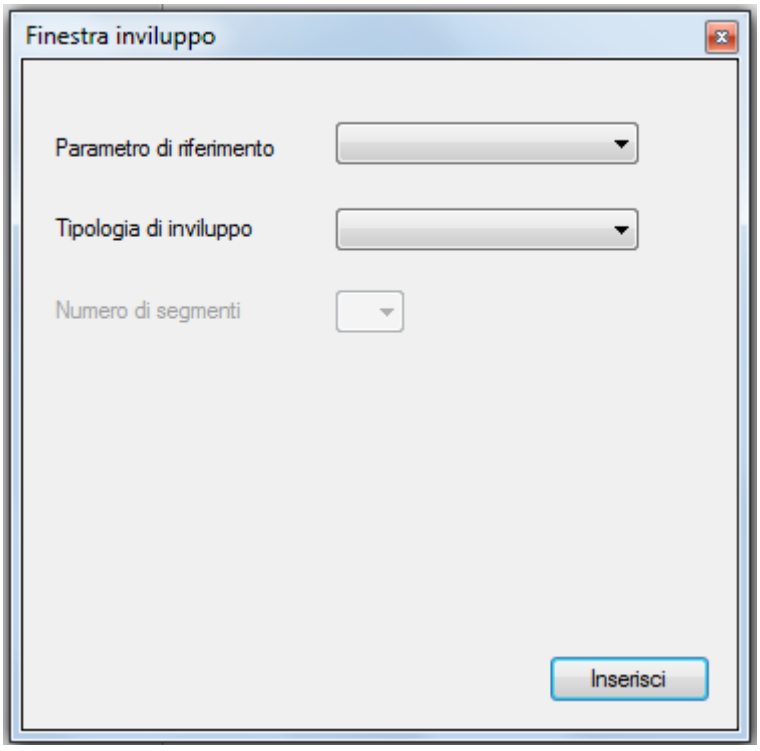


**Figura 75 - Schermata "Finestra inviluppo" senza selezioni**

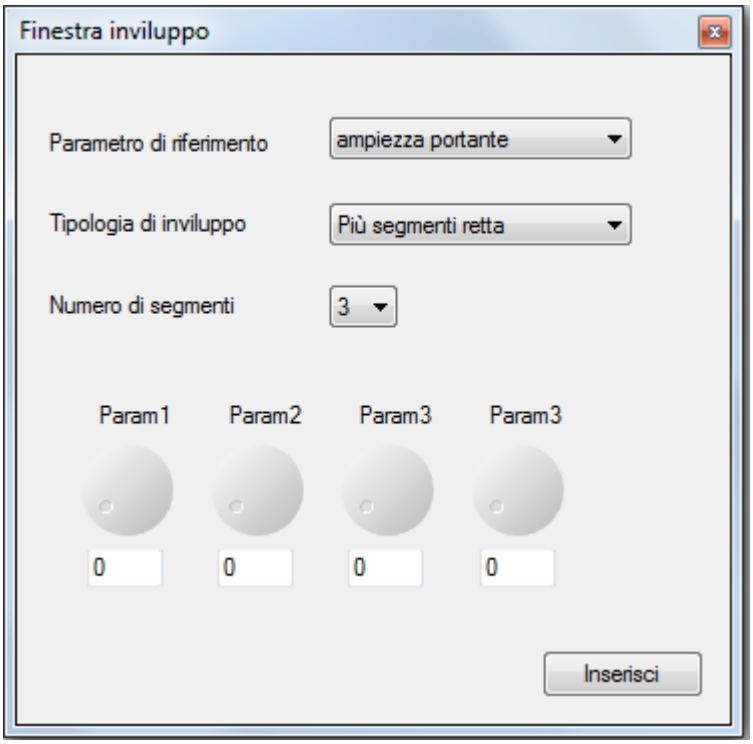


**Figura 86 - Schermata "Finestra inviluppo" con le selezioni**

In totale è possibile creare 8 inviluppi ossia uno per ogni parametro, infatti dopo aver effettuato le regolazioni per un parametro, cambiando voce nella combobox apposita si aggiunge un'altra riga nella textbox multilinea che rappresenta il parametro appena selezionato; per confermare gli inviluppi inseriti occorre cliccare sul tasto “Inserisci” che mostrerà una messagebox che chiede all’utente se è sicuro di voler applicare gli inviluppi appena scritti.
Se la risposta è affermativa, la finestra di dialogo viene chiusa e nella textbox multilinea vengono lasciati scritti gli inviluppi;
se invece la risposta è negativa si ritorna “Finestra inviluppo” per poter effettuare ulteriori regolazioni.

Quando si è ritornati nel form principale ma si vuole eliminare gli inviluppi creati è possibile cliccare nella voce “Rimuovi tutti gli inviluppi” del menu “Inviluppo” che ha come effetto quello di eliminare tutte le righe contenenti gli inviluppi appena scritti.
Tale voce del menu “Inviluppo” è disabilitata quando non ci sono inviluppi scritti e viene abilitata altrimenti; invece la voce “Apri la

finestra inviluppo" è abilitata quando non ci sono inviluppi scritti e diventa disabilitata nel momento della conferma degli inviluppi.

### 3.5 Tabpage "Score"

In questa sezione del form principale vengono visualizzati come affermato in precedenza un'altra textbox multilinea nella parte sinistra della tabpage e nella parte destra alcuni dei principali controlli dello Score: gestione delle armoniche e i tempi di attacco e durata della nota.

Nella textbox multilinea vengono visualizzati i tag relativi allo Score e quelli di chiusura del documento Csound oltre alle due stringhe relative ai parametri di default della funzione e della nota.

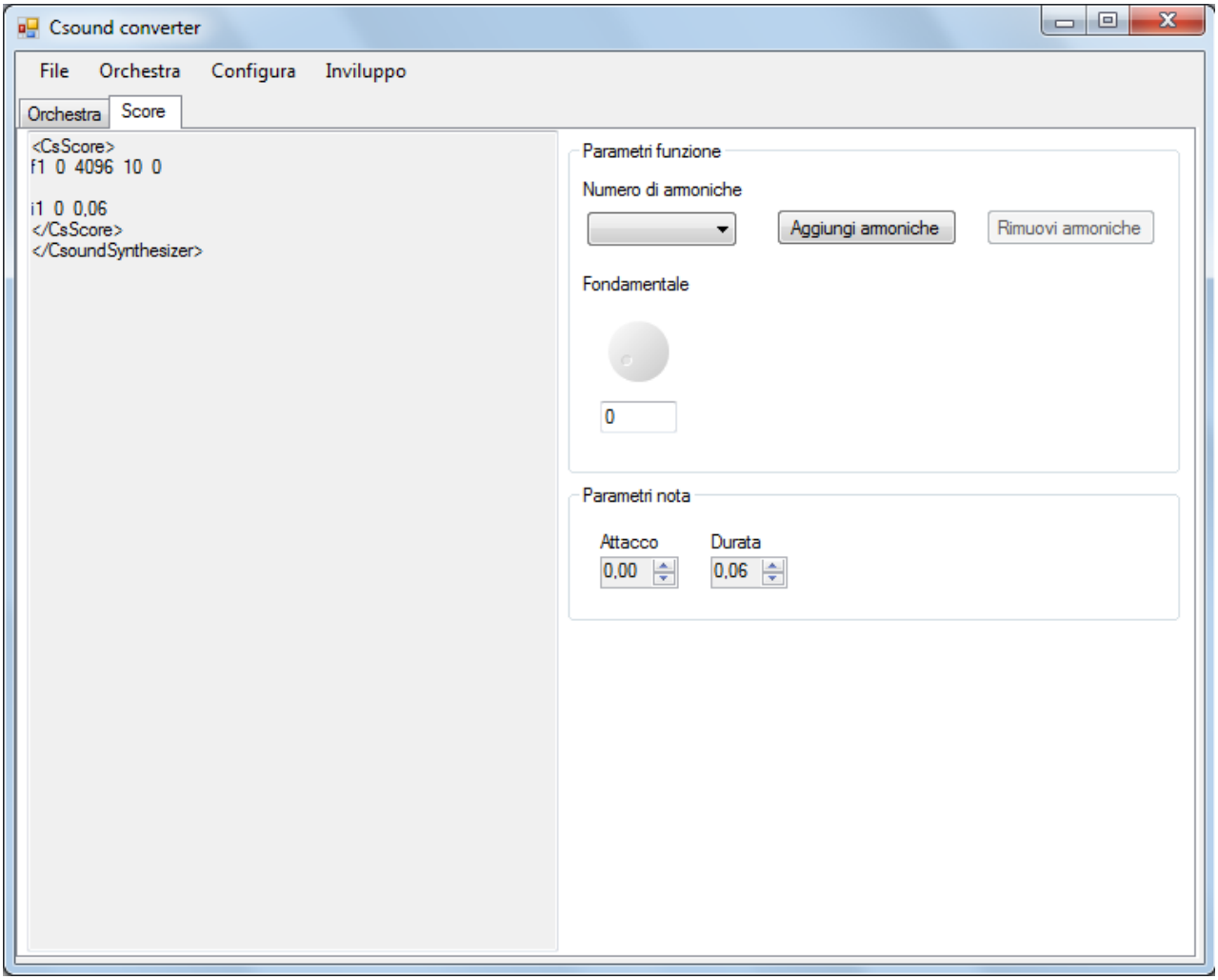


**Figura 9 - Schermata della tabpage "Score"**

Per quanto riguarda la gestione delle armoniche, i controlli a questo preposti a tale scopo sono raggruppati in una groupbox chiamata “Parametri funzione”, di cui di seguito verrà riportata un’immagine.

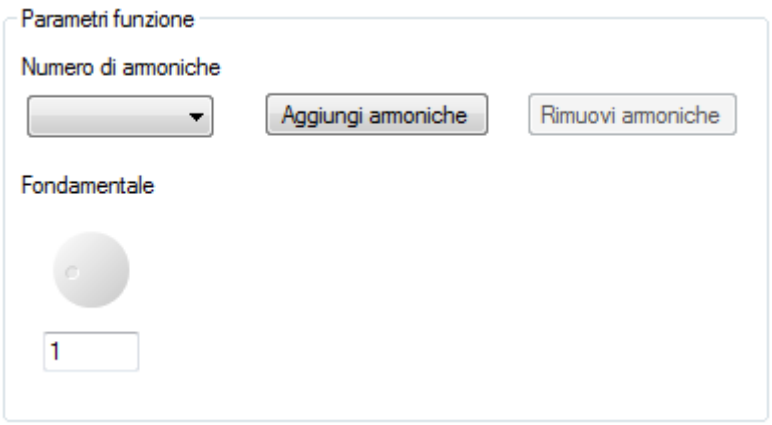


**Figura 108 - groupbox "Parametri Funzione"**

i controlli contenuti in tale container sono una combobox, due button e uno knob con texbox annessa, il cui valore di default è 1.
Il primo permette all’utente di selezionare il numero di armoniche, che nell’elaborato è un numero limitato tra 2 e 5 ma si può implementare lo stesso meccanismo su un numero maggiore di armoniche.
Il secondo chiamato “Aggiungi armoniche”, che nella situazione in cui vi è solo l’armonica fondamentale risulta abilitato , genera a seconda del valore scelto nella combobox i controlli delle armoniche e si disabilita appena viene cliccato; e il terzo chiamato “Rimuovi armoniche” che nella situazione iniziale risulta disabilitato, rimuove tutte le armoniche eccetto la fondamentale consentendo di variare il numero di armoniche abilitando il tasto “Aggiungi armoniche”.

A seconda del numero di armoniche aggiunte vengono visualizzati nella stringa di sintassi della funzione tanti zero dopo l’1 che rappresenta l’ampiezza dell’armonica fondamentale; e come nel caso dei parametri della tabpage “Orchestra” al movimento dei vari

controlli varia sia il valore nella textbox sottostante che il parametro scritto nella textbox multilinea.

Per quanto riguarda i parametri di attacco e durata della nota, essi sono gestiti da due controlli di tipo numeric-updown, contenuti nella groupbox “Parametri nota” di cui vi è un’immagine di seguito.


I due numeric-updown sono di sola lettura, quindi non permettono l’inserimento manuale dei valori ma soltanto con le freccette, e al variare del loro valore variano anche i corrispettivi parametri nella stringa di codice Csound che gestisce la nota.

### 3.6 Esportazione del file.csd

Dopo che l’utente ha concluso le regolazioni di tutti i parametri può effettuare con il comando “Salva” nel menu “File” un riversamento del contenuto delle textbox multilinea nelle pagine Orchestra e Score, ottenendo un documento di testo con l’estensione .csd in modo che sia direttamente importabile da un compilatore Csound (in questo caso csoundqt) per riprodurre il suono.

Come directory di partenza è stato scelto il Desktop, per semplicità di reperimento del documento e come nome file di default è stato scelto: “Documento Csound1”.
E’ anche possibile salvare il file come documento di testo con estensione .txt nel caso si voglia modificare manualmente con un altro programma oppure stamparlo.

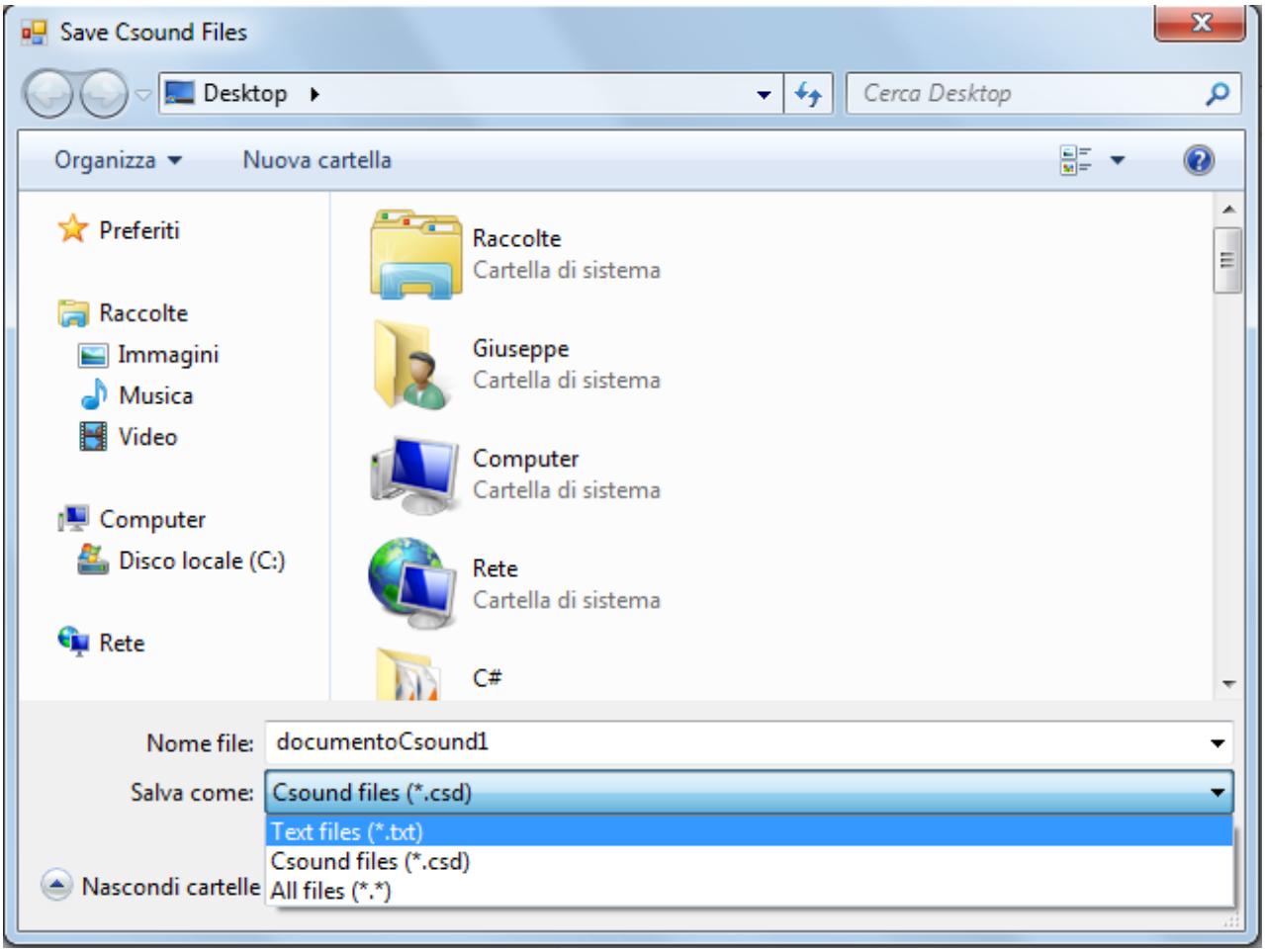


**Figura 18 - Schermata della savefiledialog "Save Csound Files".**

# 4. Descrizione del codice e delle relative funzioni

## 4.1 Dati tecnici

Il programma come ribadito nel capitolo 1, è una Windows Forms Application composta da un form principale, due form secondari e una savefiledialog; essa è sviluppata sia in ambiente Visial Studio che nella piattaforma Monodevelop per garantirne il funzionamento in ambiente Mac.
Nel codice prevale l'approccio Event-Oriented e Object-Oriented per la grande presenza dei Widjet grafici, inoltre è stato importato un file Dll esterno nel progetto in quanto nel Framework principale non era presente una classe per far fronte alla loro creazione.

## 4.2 Inserimento Tag e Header di default

Prima di illustrare come sono stati gestiti l'inserimento dei tag e dell'header di default occorre prima far luce su delle istruzioni fondamentali poste all'inizio del form principale, ossia le seguenti:

```
public string[] newLines = new string[36];
public string[] newLines2 = new string[6];
```

tali istruzioni inizializzano due array che rappresentano le line delle due textbox multilinea (ossia la cui proprietà multiline è settata su true), e in particolare la primo array chiamato "newLines" contiene 37 linee, mentre il secondo ne contiene 7; da notare l'accessibilità "public" di entrambi gli array che devono essere accessibili anche dagli altri form come si vedrà in seguito.

Il metodo usato per l'inserimento dei tag sia nell'Orchestra che nell'Header è il seguente:

```
private void InserisciTag()
{
```

```
newLines[0] = "<CsoundSynthesizer>";
newLines[1] = "<CsOptions>";
newLines[3] = "</CsOptions>";
newLines[4] = "<CsInstruments>";
newLines[34] = "</CsInstruments>";
newLines2[0] = "<CsScore>";
newLines2[4] = "</CsScore>";
newLines2[5] = "</CsoundSynthesizer>";
}
```

Tale metodo non fa altro che scrivere del semplice testo a determinate righe delle due textbox cambiando il numero situato tra le parentesi quadre.
Lo stesso principio viene applicato nel metodo che scrive l'header di default:

```
public void InserisciHeaderDefault()
{
newLines[6] = ";FM.orc";
newLines[7] = "sr = 44100";
newLines[8] = "kr = 2205";
newLines[9] = "ksmps = 20";
newLines[10] = "nchnls = 1";
}
```

### 4.3 Gestione delle configurazioni FM

La visualizzazione dei controlli appartenenti alle tre configurazioni di tipologie di sintesi FM trattate nell'elaborato è gestita mediante tre metodi ma siccome sono ridodanti verranno illustrati i primi due:

```
public void SimpleFM_Conf()
{
ampPort2.Value = ampPort2.Minimum;
frqPort2.Value = frqPort2.Minimum;
frqMod2.Value = frqMod2.Minimum;
IN_MOD2.Value = IN_MOD2.Minimum;
textb2.Text = "" + amp_portante.Value;
textb1.Text = "" + freq_port.Value;
textb3.Text = "" + freq_modulante.Value;
textb4.Text = "" + indice_mod.Value;
tab_Orchestra.Controls.Remove(frqMod2);
tab_Orchestra.Controls.Remove(IN_MOD2);
tab_Orchestra.Controls.Remove(label_IM2);
tab_Orchestra.Controls.Remove(label_frqMod2);
tab_Orchestra.Controls.Remove(txt_frqMod2);
tab_Orchestra.Controls.Remove(txt_IM2);
```

```
tab_Orchestra.Controls.Remove(frqPort2);
tab_Orchestra.Controls.Remove(ampPort2);
tab_Orchestra.Controls.Remove(label_frqPort2);
tab_Orchestra.Controls.Remove(label_ampPort2);
tab_Orchestra.Controls.Remove(txt_frqPort2);
tab_Orchestra.Controls.Remove(txt_ampPort2);
FormC.PR_Combo.Items.Remove("ampiezza portante2");
FormC.PR_Combo.Items.Remove("frequenza portante2");
FormC.PR_Combo.Items.Remove("frequenza modulante2");
FormC.PR_Combo.Items.Remove("indice Modulazione2");
cntrl = 0;
}

public void ParallelCarriers_Conf()
{
ampPort2.Value = 50;
frqPort2.Value = 1300;
frqPort2.Size = new Size(56, 56);
ampPort2.Size = new Size(56, 56);
frqPort2.Location = new Point(443, 244);
ampPort2.Location = new Point(347, 244);
frqPort2.Minimum = 20;
frqPort2.Maximum = 20000;
ampPort2.Minimum = 0;
ampPort2.Maximum = 32767;
frqPort2.ShowLargeScale = false;
ampPort2.ShowLargeScale = false;

this.frqPort2.ValueChanged += new
KnobControl.ValueChangedEventHandler
(this.frqPort2_ValueChanged);

this.ampPort2.ValueChanged += new
KnobControl.ValueChangedEventHandler
(this.ampPort2_ValueChanged);

label_frqPort2.Size = new Size(94, 13);
label_ampPort2.Size = new Size(94, 13);
label_frqPort2.Location = new Point(440, 219);
label_ampPort2.Location = new Point(344, 219);
label_frqPort2.Text = "Freq. Portante2";
label_ampPort2.Text = "Amp. Portante2";
txt_frqPort2.Size = new Size(56, 20);
txt_ampPort2.Size = new Size(56, 20);
txt_frqPort2.Location = new Point(443, 306);
txt_ampPort2.Location = new Point(347, 306);
txt_frqPort2.Text = "" + frqPort2.Value;
txt_ampPort2.Text = "" + ampPort2.Value;

this.txt_frqPort2.TextChanged += new
System.EventHandler(this.txt_frqPort2_TextChanged);

this.txt_frqPort2.KeyPress += new
System.Windows.Forms.KeyPressEventHandler
(this.txt_frqPort2_KeyPress);

this.txt_ampPort2.TextChanged += new
System.EventHandler(this.txt_ampPort2_TextChanged);

this.txt_ampPort2.KeyPress += new
System.Windows.Forms.KeyPressEventHandler
```

```
(this.txt_ampPort2_KeyPress);

this.txt_ampPort2.Leave += new
System.EventHandler(this.txt_ampPort2_Leave);

this.txt_frqPort2.Leave += new
System.EventHandler(this.txt_frqPort2_Leave);

tab_Orchestra.Controls.Add(frqPort2);
tab_Orchestra.Controls.Add(ampPort2);
tab_Orchestra.Controls.Add(label_frqPort2);
tab_Orchestra.Controls.Add(label_ampPort2);
tab_Orchestra.Controls.Add(txt_frqPort2);
tab_Orchestra.Controls.Add(txt_ampPort2);
tab_Orchestra.Controls.Remove(frqMod2);
tab_Orchestra.Controls.Remove(IN_MOD2);
tab_Orchestra.Controls.Remove(label_IM2);
tab_Orchestra.Controls.Remove(label_frqMod2);
tab_Orchestra.Controls.Remove(txt_frqMod2);
tab_Orchestra.Controls.Remove(txt_IM2);
FormC.PR_Combo.Items.Add("ampiezza portante2");
FormC.PR_Combo.Items.Add("frequenza portante2");
FormC.PR_Combo.Items.Remove("frequenza modulante2");
FormC.PR_Combo.Items.Remove("indice Modulazione2");
cntrl = 1;
}
```

Il primo metodo chiamato “Simple_FM_conf” si occupa della visualizzazione dei controlli della configurazione “Simple FM” ossia: uno knob che controlla l’ampiezza della portante, uno che ne controlla la frequenza, uno knob che controlla la frequenza della modulante e una trackbar che controlla l’indice di modulazione.

In particolare il primo metodo nelle prime quattro istruzioni resetta valori dei controlli della configurazione a cui fa riferimento, ponendoli al loro valore minimo, nelle successive 4 permette di visualizzare nelle textbox sottostanti ai valori della “Simple FM” i corrispettivi valori e nelle ultime elimina ogni controllo aggiuntivo non appartenente alla “Simple FM”, che sono knob, textbox e label.

Il secondo metodo invece oltre a rimuovere i controlli che non appartengono alla configurazione a cui fa riferimento ossia la “Parallel carriers”, mediante le istruzioni “Remove”, definisce le caratteristiche dei controlli di tale configurazione, in quanto non sono stati creati nella finestra di progettazione, e inizializza una variabile di tipo int chiamata ctrl a 0, il cui significato verra spiegato in seguito.

Tra queste caratteristiche le più importanti sono le proprietà: location che definisce la posizione del controllo nel form, size che ne definisce la dimensione, minimum e maximum che definiscono i valori massimi e minimi dei controlli knob, value che definisce il valore degli knob che in questo caso sono settati a dei valori di default e altre caratteristiche aggiuntive.
Inoltre vengono sottoscritti gli eventi: value changed, textchanged, key-press e leave ai loro relativi event handler che verranno illustrati successivamente, vengono aggiunti i controlli al form mediante le istruzioni "add" e viene settata la variabile ctrl di tipo int a 1.

### 4.4 Scrittura dei valori degli knob nel documento Csound

La compilazione del documento Csound con i valori immessi dall'utente dei parametri delle varie configurazioni avviene tramite 2 eventi: il movimento degli knob e la scrittura del valore nelle relative textbox.
Nel primo caso viene illustrato di seguito il codice di riferimento:

```
private void amp_portante_ValueChanged(object Sender)
{
textb2.Text = "" + amp_portante.Value;
textBox1.Lines = newLines;
newLines[13] = " " + " " + " " + "icamp = " +
amp_portante.Value;
}
```

In tale frammento viene riportanto la sintassi dell'event handler riferito all'evento "Value changed", il quale fa si' che al cambiamento del valore dello knob, il testo sottostante allo stesso, mostri il valore corrente e nel contempo nella riga 13 della textbox multilinea venga visualizzata oltre alla stringa fissa "icamp = " il valore corrente dello knob.
Nel secondo caso considerando la textbox che controlla il medesimo parametro, vengono coinvolti 3 metodi:

```
private void textb2_TextChanged(object sender, EventArgs e)
{
 if (textb2.Text == "")
  {
  textb2.Text = "";
  }

 else if (int.Parse(textb2.Text) > amp_portante.Maximum)
  {
  MessageBox.Show("Valore non valido", "error",
  MessageBoxButtons.OK, MessageBoxIcon.Error);
  textb2.Text = amp_portante.Minimum.ToString();
  }

 else
  {
  amp_portante.Value = int.Parse(textb2.Text);
  this.amp_portante.ValueChanged += new
  knobControl.ValueChangedEventHandler
 (this.amp_portante_ValueCh  anged);
  }
}


private void textb2_Leave(object sender, EventArgs e)
 {
  if (textb2.Text == "")
  {
  textb2.Text = amp_portante.Minimum.ToString();
  }
 }


private void textb2_KeyPress(object sender, KeyPressEventArgs
e)
 {
 if (!(char.IsDigit(e.KeyChar) || e.KeyChar == '\b'))
    {
    MessageBox.Show("Carattere non valido", "error",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
    e.Handled = true;
    }
 }
```

Il primo metodo è l'event handler dell'evento "Text changed" il quale fa si' che venga scritto il valore del parametro che rappresenta purchè non superi i valori massimi e minimi e purchè ci sia una valore digitato cioè che la textbox non sia vuota.

Se la textbox fosse vuota e si uscisse dal controllo si innescherebbe l'evento "On leave" che tramite il corrispondente l'event handler

ossia il secondo metodo nel frammento di codice, setta il valore della textbox al valore minimo del parametro che controlla.
Il terzo metodo invece è l'event handler dell'evento "Key press" della textbox, ossia l'evento che viene chiamato ogni qual volta si digiti qualcosa all'interno del controllo.
Siccome occorre che l'utente digiti solo caratteri numerici, esso fa si' che ogni qual volta digiti un carattere che non sia o una cifra numerica o il tasto "Backspace" per cancellare il contenuto, visualizzi un messaggio di errore e elimini il carattere digitato dalla controllo.

### 4.5 La "Finestra inviluppo"

La visualizzazione della finestra di dialogo "Finestra inviluppo" viene descritta dal seguente frammento di codice:

```
public Envelop_Form1 FormC = new Envelop_Form1();

private void aggiungiInviluppoToolStripMenuItem_Click(object
sender, EventArgs e)
 {
  FormC.Show();
  FormC.mn1 = this;
  FormC.FormClosed += new
  FormClosedEventHandler(Envelop_Form1_FormClosed);
  this.Enabled = false;
  }
```

Nella prima riga del frammento viene inizializzata una variabile del tipo del form che rappresenta la "Finestra inviluppo" che viene chiamato "Envelop_form1".

Nella seconda riga viene descritto il metodo event handler dell'evento click della voce "Aggiungi inviluppo" nel menustrip del form principale.
In pratica le istruzioni contenute in tale metodo vengono eseguite nel caso l'utente faccia click nella voce "Aggiungi inviluppo" nel menu "Inviluppo" e in ordine di apparizione:

1. La prima istruzione mostra il form “Envelop_form1”.
2. La seconda estende l’ambito degli oggetti contenuti nell’Envelop_form1 al quello del form principale, cioè con questa istruzione è possibile richiamare qualsiasi oggetto (metodo, campo, etc. purchè pubblici) dell’Envelop_form1 nel form principale.
3. La terza sottoscrive l’evento “Form_Closed” con l’event handler corrispondente nel form principale.
4. La quarta disabilita il form principale in modo che l’utente quando visualizza la “Finestra inviluppo”, possa solo regolare i controlli di tale finestra.

Il meccanismo di visualizzazione dei controlli nella “Finestra inviluppo” è lo stesso visto nella gestione delle configurazioni FM con la differenza che anziché essere causata dal click in una voce di menu, è causata dal variare delle voci della combobox, ossia dall’evento “Select value changed”.

Nello specifico verrà esaminato di seguito un estratto relativo all’event hadler dell’evento “Select value changed” della seconda combobox che come visto in precedenza permette di scegliere il tipo di inviluppo:

```
private void EnvType_Combo_SelectedValueChanged(object sender,
 EventArgs e)
 {
  NumSeg_Combo.SelectedIndex = -1;
  if (EnvType_Combo.SelectedIndex == 0 &&
 (string)PR_Combo.SelectedItem != null) // caso singolo
 segmento retta
  {
   label2.Enabled = false;
   NumSeg_Combo.Enabled = false;
   KnobSSR1.Location = new Point(25, 187);
   KnobSSR2.Location = new Point(90, 187);
   KnobSSR1.Size = new Size(59, 59);
   KnobSSR2.Size = new Size(59, 59);
   KnobSSR1.Value = KnobSSR1.Minimum;
   KnobSSR2.Value = KnobSSR2.Minimum;
   KnobSSR1.ShowLargeScale = false;
   KnobSSR2.ShowLargeScale = false;
   this.KnobSSR1.ValueChanged += new
   knobControl.ValueChangedEventHandler
  (this.KnobSSR1_ValueChanged);
```

```
this.KnobSSR2.ValueChanged += new
KnobControl.ValueChangedEventHandler(this.KnobSSR2_ValueChange
d);
LabEnv1.Size = new Size(65, 13);
LabEnv2.Size = new Size(65, 13);
LabEnv1.Location = new Point(37, 172);
LabEnv2.Location = new Point(102, 172);
LabEnv1.Text = "Param1";
LabEnv2.Text = "Param2";
TxtEnv1.Size = new Size(38, 20);
TxtEnv2.Size = new Size(38, 20);
TxtEnv1.Location = new Point(34, 245);
TxtEnv2.Location = new Point(99, 245);
TxtEnv1.Text = KnobSSR1.Minimum.ToString();
TxtEnv2.Text = KnobSSR2.Minimum.ToString();
this.TxtEnv1.TextChanged += new
System.EventHandler(this.TxtEnv1_TextChanged);
this.TxtEnv1.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.TxtEnv1_KeyPres
s);
this.TxtEnv1.Leave += new
System.EventHandler(this.TxtEnv1_Leave);
this.TxtEnv2.TextChanged += new
System.EventHandler(this.TxtEnv2_TextChanged);
this.TxtEnv2.KeyPress += new
System.Windows.Forms.KeyPressEventHandler(this.TxtEnv2_KeyPres
s);
this.TxtEnv2.Leave += new
System.EventHandler(this.TxtEnv2_Leave);
this.Controls.Add(KnobSSR1);
this.Controls.Add(KnobSSR2);
this.Controls.Add(LabEnv1);
this.Controls.Add(TxtEnv1);
this.Controls.Add(LabEnv2);
this.Controls.Add(TxtEnv2);
this.Controls.Remove(TxtEnv3);
this.Controls.Remove(TxtEnv4);
this.Controls.Remove(LabEnv3);
this.Controls.Remove(LabEnv4);
this.Controls.Remove(KnobPSR1);
this.Controls.Remove(KnobPSR2);
this.Controls.Remove(KnobPSR3);
this.Controls.Remove(KnobPSR4);
this.Controls.Remove(KnobSSE1);
this.Controls.Remove(KnobSSE2);
this.Controls.Remove(KnobPSE1);
this.Controls.Remove(KnobPSE2);
this.Controls.Remove(KnobPSE3);
this.Controls.Remove(KnobPSE4);
this.Controls.Remove(KnobIAT1);
this.Controls.Remove(KnobIAT2);
this.Controls.Remove(KnobIAT3);
}
```

La prima istruzione del metodo consente di resettare I valori scelti nella combobox contenente i numeri dei segmenti, e successivamente tramite delle istruzioni If che oltre a visualizzare i controlli con le

relative proprietà a seconda della scelta operata nella combobox in questione, verifica che sia stata effettuata una scelta nella combobox indicante quale parametro sottoporre a inviluppo.
Infatti non effettuando nessuna scelta in tale combobx non verrà visualizzato alcun controllo; questa scelta è stata fatta affinchè l'utente effettui le sue scelte dalla combobox più in alto a quella più in basso.
Le tipologie di controlli inseriti sono: knob,textbox, e label; e gli eventi sottoscritti in questo metodo sono gli stessi visti per l'inserimento dei controlli nella tabpage "Orchestra" quindi: value changed, text changed, on leave e key press.

L'event hadler dell'evento "Select value changed" della prima combobox invece non ha come effetto la visualizzazione dei controlli ma permette di :

1. Resettare i valori di ogni knob in modo da essere posti al minimo nelle selezioni successive.
2. Determinare tramite alla chiamata al metodo ValuesRange(), quali siano valori minimi e massimi di ogni knob.
3. E infine permettere la scelta del tipo di inviluppo chiamato "Inviluppo d'ampiezza trapezioidale" limitatamente ai parametri "Ampiezza portante" e "Ampiezza portante2", in quanto non applicabile agli altri parametri[4].

Per quanto riguarda invece la scrittura dei valori, la logica del codice che verrà riportato in seguito, è che l'utente può scegliere di ciascuno parametro solo un tipo di inviluppo, infatti è logico pensare che non è possibile avere in contemporanea due tipologie diverse di inviluppi dello stesso parametro; l'utente può sottoporre a inviluppo solo alcuni parametri, tutti i parametri o anche nessun parametro dato che non è indispensabile per la definizione del timbro finale;
e infine l'utente non può eliminare la scrittura di un singolo inviluppo nella "Finestra inviluppo" ma può solo cambiarne la tipologia.

Tenendo conto di tali considerazioni il seguente è un estratto di uno dei 5 metodi usati per la visualizzazione dei valori degli inviluppi nella textbox multilinea della tabpage “Orchestra”:

```
private void valoriSSR()
{
TxtEnv1.Text = "" + KnobSSR1.Value;
TxtEnv2.Text = "" + KnobSSR2.Value;
dividi();
if ((string)PR_Combo.SelectedItem == "ampiezza portante")
   {
   contatore1 = "1";
   if (!(PR_Combo.Items.Contains("ampiezza portante2")
     ||R_Combo.Items.Contains("indice Modulazione2")))
   {
   mn1.textBox1.Lines = mn1.newLines;
   mn1.newLines[21] = " " + " " + " " + "kenvcar  line " +
KnobSSR1.Value + ", " + durata + ", " + KnobSSR2.Value;
   mn1.newLines[29] = " " + " " + " " + "aout foscili  " +
param1 + ", 1, " + param2 + ", " + param3 + ", " + param4 + ",
1";
   param4 = "indx";
   param2 = "ipk";
   param1 = "kenvcar";
   param3 = "imk";

   if (contatore2 != null)
    {
    param2 = "kglis";
    }
    if (contatore3 != null)
    {
    param3 = "kglis2";
    }
    if (contatore4 != null)
    {
    param4 = "kindxenv";
    }
  }
 if(PR_Combo.Items.Contains("ampiezza portante2"))
   {
   mn1.textBox1.Lines = mn1.newLines;
   mn1.newLines[21] = " " + " " + " " + "kenvcar  line " +
KnobSSR1.Value + ", " + durata + ", " + KnobSSR2.Value;
   mn1.newLines[29] = " " + " " + " " + "acar foscili  " +
param1 + ", 1, " + param2 + ", " + param3 + ", " + param4 + ",
1";
   mn1.newLines[30] = " " + " " + " " + "acar2 foscili  " +
param5 + ", 1, " + param6 + ", " + param3 + ", " + param4 + ",
1";
   param4 = "indx";
   param2 = "ipk";
   param1 = "kenvcar";
   param3 = "imk";
   param5 = "icamp2";
   param6 = "ipk2";
   if (contatore2 != null)
```

```
    {
    param2 = "kglis";
    }
   if (contatore3 != null)
    {
    param3 = "kglis2";
    }

   if (contatore4 != null)
   {
    param4 = "kindxenv";
    }
   if (contatore5 != null)
    {
    param5 = "kenvcar2";
    }
   if (contatore6 != null)
    {
    param6 = "kglis3";
    }
   }
 if(PR_Combo.Items.Contains("indice Modulazione2"))
   {
    mn1.textBox1.Lines = mn1.newLines;
    mn1.newLines[21] = " " + " " + " " + "kenvc  line " +
KnobSSR1.Value + ", " + durata + ", " + KnobSSR2.Value;
    mn1.newLines[29] = " " + " " + " " + "amod oscili  " +
param1 + "*" + param2 + ", " + param2 + ", 1";
    mn1.newLines[30] = " " + " " + " " + "amod2 oscili  " +
param8 + "*" + param7 + ", " + param7 + ", 1";
    mn1.newLines[31] = " " + " " + " " + "acar oscili  " +
param3 + ", " + param4 + " + amod1 + amod2, 1";
    param1 = "indx";
    param4 = "ipk";
    param3 = "kenvc";
    param2 = "imk";
    param7 = "imk2";
    param8 = "indx2";
   if (contatore2 != null)
    {
    param4 = "kglis";
    }
   if (contatore3 != null)
    {
    param2 = "kglis2";
    }
   if (contatore4 != null)
    {
    param1 = "kindxenv";
    }
   if (contatore7 != null)
    {
    param7 = "kglis4";
    }
   if (contatore8 != null)
    {
    param8 = "kindxenv2";
    }
  }
}
```

Nel seguente frammento, oltre alle istruzioni già viste come le prime due che visualizzano il valore degli knob nelle textbox, ci sono tre istruzioni "If" annidate nel quale a seconda del grado di annidamento:

1. Nella più esterna viene controllato il parametro selezionato nella combobox "PR_combo".
2. Nell'intermedia viene controllato che tipo di configurazione tramite il metodo "Contains", infatti viene dedotta la configurazione controllando se la "PR_combo" ha come oggetto "Ampiezza portante2" oppure "Indice di modulazione2" o nessuno dei due.
3. E nella più interna se sono stati selezionati già altri parametri, in quanto si deve tenere conto che la selezione che effettua l'utente non è necessariamente la prima.

Nelle istruzioni "NewLines" compare oltre ai valori degli knob, compare una variabile chiamata "Durata", dichiarata all'esterno del metodo e inizializzata nel metodo Dividi() invocato all'inizio di questo metodo (che verrà illustrato di seguito), che rappresenta la durata della nota immessa nel form principale, nella tabpage "Score".

Il metodo Dividi() invocato all'inizio del metodo valoriSSR(), è formato in tale modo:

```
string valDiviso2;
string valDiviso3;
string durata;

public void dividi()
 {
  decimal valDiv2 = Decimal.Divide(mn1.textDuration.Value, 2);
  decimal valDiviso2Rounded = decimal.Round(valDiv2, 2);
  valDiviso2 =
  valDiviso2Rounded.ToString(System.Globalization.Culture
  Info.InvariantCulture);

  Decimal valDiv3 = Decimal.Divide(mn1.textDuration.Value, 3);
  decimal valDiviso3Rounded = decimal.Round(valDiv3, 2);
  valDiviso3 =
  valDiviso3Rounded.ToString(System.Globalization.Culture
  Info.InvariantCulture);

  durata =
```

```
 mn1.textDuration.Value.ToString(System.Globalization.Culture
 Info.InvariantCulture);
}
```

Nel presente frammento sono dichiarate tre variabili di tipo string, chiamate “Durata”, “ValDiviso2” e “Valdiviso3” che rappresentano rispettivamente: la durata della nota nello score, il valore della durata diviso 2 e il valore della durata diviso 3.
Tali variabili vengono inizializzate all’interno del metodo con i risultati della divisione decimale per due e per tre della durata e la durata stessa, arrotondati a due cifre dopo la virgola e convertite in string in cui tramite il metodo string(System.Globalization.CultureInfo.InvariantCulture) con cui viene anche cambiato il separatore decimale dalla virgola al punto in quanto in Csound i valori decimali vengono scritti con il separatore decimale punto.

Nel caso che l’utente voglia chiudere il form “Finestra inviluppo” mediante la “X” in alto a destra, viene scattato l’evento “Form Closed” il cui event handler definito nel form principale:

1. Elimina il contenuto di tutte le righe scritte con la “Finestra inviluppo” mediante il metodo EliminaInviluppi.
2. Nasconde il form attuale e riattiva il form principale.
3. A seconda del valore inizializzato alla variabile intera ctrl il cui valore cambia a seconda della configurazione scelta, le due istruzioni Csound finali vengono riscritte.

Nel caso invece che l’utente decida di inserire i gli inviluppi appena regolati cliccando il pulsante inserisci, viene visualizzata una messagebox che chiede se l’utente è sicuro di inserire i presenti inviluppi; in caso risponda si questi vengono applicati e divengono definitivi, in caso risponda negativamente si ritorna alla “Finestra inviluppo” dando la possibilità di effettuare ulteriori regolazioni.

In caso l'utente clicchi prematuramente nel tasto inserisci senza effettuare alcuna regolazione, viene visualizzata una messagebox invitando l'utente a selezionare almeno un inviluppo da applicare.
Questo avviene in virtù del seguente codice contenuto all'interno dell'event handler buttonInsInv_Click:

```
mn1.textBox1.Lines = mn1.newLines;
  if (mn1.newLines[21] == null && mn1.newLines[22] == null &&
mn1.newLines[23] == null && mn1.newLines[24] == null &&
mn1.newLines[25] == null && mn1.newLines[26] == null &&
mn1.newLines[27] == null && mn1.newLines[28] == null)
  {
MessageBox.Show("Devi selezionare almeno un inviluppo da
applicare", "Csound Converter", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
  }
```

Infatti in tale metodo mediante un'istruzione If viene visualizzata la messagebox sopracitata se e solo se tutte le righe della textbox multilinea adibite alla scrittura degli inviluppi sono contemporaneamente nulle, ossia non contengono alcun carattere.

### 4.6 Gestione parametri dello Score

I parametri dello score gestiti nella apposita tabpage sono come già accennato:l'ampiezza della fondamentale e delle eventuali armoniche per quanto rigurda la funzione e il tempo di attacco e di durata per quanto riguarda la nota.

Di seguito viene riportato il metodo atto alla scrittura delle armoniche nella textbox multilinea della tabpage score:

```
private void ScriviArmoniche()
 {
  if (ComboArm.SelectedIndex == 0)
    {

newLines2[1] = "f1  0  4096  10  " + knobFond.Value + "  " +
Arm2.Value;
textBox2.Lines = newLines2;

    }
  if (ComboArm.SelectedIndex == 1)
    {

newLines2[1] = "f1  0  4096  10  " + knobFond.Value + "  " +
Arm2.Value + "  " + Arm3.Value;
textBox2.Lines = newLines2;
    }

    if (ComboArm.SelectedIndex == 2)
      {
      newLines2[1] = "f1  0  4096  10  " + knobFond.Value + "
" + Arm2.Value + "  " + Arm3.Value + "  " + Arm4.Value;
      textBox2.Lines = newLines2;
      }
    if (ComboArm.SelectedIndex == 3)
      {

newLines2[1] = "f1  0  4096  10  " + knobFond.Value + "  " +
Arm2.Value + "  " + Arm3.Value + "  " + Arm4.Value + "  " +
Arm5.Value;

      textBox2.Lines = newLines2;
      }
 }
```

Come già affermato nel capitolo “Descrizione dell’applicazione” le armoniche inseribili oltre alla fondamentale possono variare da 1 a 4 e questo metodo a seconda del valore della combobox in cui viene selezionato il numero di armoniche, viene aggiunta la stringa corrispondente contenente i valori di default di tali armoniche.

I parametri di attacco e di durata della nota vengono come già asserito in precedenza, vengono gestiti da due semplici controlli di tipo numeric-updown, in cui le cifre decimali sono limitate a 2, e per scelta tecnica vengono impostate in sola lettura.
La logica di immissione dei valori nella textbox multilinea è la stessa per gli altri parametri spiegati in precendenza.

### 4.7 Finestra di salvataggio e reset dell'applicazione

La visualizzazione della finestra di salvataggio è gestita dal presente metodo:

```
private void SaveMenu_Click(object sender, EventArgs e)
 {
  SaveFileDialog saveFileDialog1 = new SaveFileDialog();
  saveFileDialog1.InitialDirectory = @"Desktop";
  saveFileDialog1.Title = "Save Csound Files";
  saveFileDialog1.CheckFileExists = false;
  saveFileDialog1.CheckPathExists = true;
  saveFileDialog1.AddExtension = true;
  saveFileDialog1.DefaultExt = ".csd";
  saveFileDialog1.Filter = "Text files (*.txt)|*.txt|Csound
files (*.csd)|*.csd|All files (*.*)|*.*";
  saveFileDialog1.FilterIndex = 2;
  saveFileDialog1.RestoreDirectory = true;
  saveFileDialog1.FileName = "documentoCsound1";

  if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {

System.IO.File.WriteAllText(saveFileDialog1.FileName,
textBox1.Text + textBox2.Text);
     }
 } // save file dialog
```

Nel presente metodo viene dichiarato un oggetto di tipo SaveFIleDialog chiamato "saveFileDialog1" e successivamente vengono definite alcune proprietà di questo oggetto che saranno applicate alla finestra di salvataggio.

Tali proprietà sono:

- Initialdirectory che inidica la directory di partenza dove viene salvato il file, che nel presente elaborato è il Desktop.
- Title che indica il titolo in alto a sinistra raffigurato nella finestra.
- checkFileExist che controlla se il file esiste prima di chiudere la presente finestra di dialogo.
- checkPathExist che controlla se il percorso dove viene salvato il file esiste prima di chiudere la finestra di dialogo.
- AddExtesions che determina se le estensioni vengono automaticamente aggiunte ai nomi dei file.

- DefaultExt che definisce l'estensione accordata ai nomi dei file digitati dall'utente per i quali non sia stata specificata alcuna estensione.
- Filter che definisce i filtri da visualizzare nella finestra di dialogo.
- Restore che determina se viene ripristinata la directory corrente prima della chiusura della finestra di dialogo.

L'ultima istruzione del metodo è un'istruzione If fa si' che se l'utente cliccasse su ok nella finestra di dialogo, viene salvato il contenuto delle due textbox multilinea nel file salvato.

Per quanto riguarda il reset dell'applicazione viene invocato tale event handler relativo all'evento click dell'omonima voce nel menu file:

```
private void nuovoFileToolStripMenuItem_Click(object sender,
EventArgs e)
 {
  freq_port.Value = 5000;
  amp_portante.Value = 50;
  freq_modulante.Value = 1200;
  indice_mod.Value = 88;
  textFond.Text = knobFond.Minimum.ToString();
  InserisciTag();
  EliminaInviluppi();
  SimpleFM_Conf();
  InserisciStrumentoSimpleFMDefault();
  InserisciHeaderDefault();
  simpleFMToolStripMenuItem.Enabled = false;
  parallelCarriersToolStripMenuItem.Enabled = true;
  parallelModulatorsToolStripMenuItem.Enabled = true;
  rimuoviTuttiGliInviluppiToolStripMenuItem.Enabled = false;
  buttonRemovArm.Enabled = false;
  textFond.Text = knobFond.Minimum.ToString();
  knobFond.Value = knobFond.Minimum;
  newLines2[1] = "f1  0  4096  10  " + knobFond.Value;
  newLines2[3] = "i1  " +
textAttack.Value.ToString(System.Globalization.CultureInfo.Inv
ariantCulture) + "  " +
textDuration.Value.ToString(System.Globalization.CultureInfo.I
nvariantCulture);
  textBox2.Lines = newLines2;
 }
```

Tale event handler si occupa di resettare tutti i parametric immessi dall'utente in modo da ritornare alla situazione iniziale e programmare un nuovo timbro.
Questo è stato reso possibile risettando ai valori di default i parametri della configurazione "Simple FM", eliminando gli inviluppi e risettando ai valori minimi o di default gli altri parametri come il valore della armonica fondamentale, il valore della durata etc.

### 4.8 Range dei valori

Per i parametri riguardanti gli operatori modulanti e portanti il rage di valori degli knob va dai 20 ai 20000 in quanto devono appartenere al range udibile per essere percepibili singolarmente[5].
Per quanto riguarda l'ampiezza della portante in Csound il range va da 0 a 32767[4] e per quanto riguarda invece l'ampiezza delle armoniche il range è stato scelto da 0 a 10 in assenza di disposizioni precise nella documentazione in atto, ma il valore può essere maggiorato in caso di riscontro con altre fonti.
Stesso discorso va applicato all'indice di modulazione in cui non vi sono disposizioni precise sia nella documentazione ufficiale di J.M. Chowning o nei manuali Csound, quindi è stato scelto un range che va da 0 il quale indica che non vi è nessuna modulazione a 100 come modulazione massima.

# 5. Sviluppi futuri

All'applicazione ottenuta possono essere aggiungere diverse features riguardanti la sintesi FM.

Ad esempio può essere aumentato il numero di strumenti anziché limitarlo a uno, combinandoli tra loro e creando timbri più interessanti e inoltre si potrebbero aumentare il numero di funzioni e di note tenendo conto che si dovrebbe cambiare anche il numero di funzione nell'orchestra alla fine degli opcode e che i valori delle note devono giacere in maniera perfetta nell'asse temporale tramite i tempi di attacco e di durata e quindi, in relazione ai valori della nota precedente, l'applicazione deve poter consentire all'utente la scelta di soli determinati valori.

Altresi' potrebbero diventare editabili altri parametri della funzione come: l'action time, il numero di punti e il metodo di generazione che nel presente elaborato sono lasciati fissi.

Per quanto riguarda la gestione delle note si potrebbe, in caso si decida di ampliare il numero delle note generabili, passare alcuni o tutti gli argomenti degli opcode dell'orchestra nello score mediante il procedimento visto nel paragrafo 2.5.

Inoltre lo schema del programma può adeguarsi agli altri tipi di sintesi sonora come ad esempio la sintesi additiva, sottrattiva e granulare.

Per quanto riguarda gli Schemi di modulazione trattati nel paragrafo 1.7, l'applicazione può essere arricchita anzitutto rendendo disponibile la configurazione "Cascade" che, come ribadito, non viene presa in considerazione nel presente elaborato e successivamente offrendo all'utente la possibilità di creare combinazioni complesse fra gli operatori tramite configurazioni "ibride" derivanti dall'uso contemporaneo di due o più configurazioni base.

Infine riguardo la tipologia del programma, si potrebbe creare una versione plug-in del software direttamente eseguibile all'interno del compilatore Csoundqt e mediante lo stesso linguaggio C# creare la

stessa applicazione mobile per android dato che esiste già da poco tempo anche un compilatore Csound per dispositivi android ed è in fase di sviluppo la versione per dispositivi IOS e per dispositivi Windows Mobile.

# 6. Conclusioni

Il software è funzionante e stabile, sia in ambiente Windows, che in ambiente Mac fornisce all'utente la possibilità di accedere all'utilizzo di Csound, limitatamente alla generazione di suoni con la sintesi FM, pur privo delle necessarie conoscenze sintattiche e questo è un vantaggio non indifferente in quanto la conoscenza della sintassi ha costituito un handicap alla fruizione di questo potente sintetizzatore virtuale che proprio per questo motivo avrebbe potuto avere un mercato più vasto.

Ciò è possibile in quanto, in sostituzione della stesura del codice Csound, all'utente sono forniti strumenti di interfaccia grafica come ad esempio, manopole e sliders, più intuitivi e di immediato utilizzo.

Inoltre questo software è utilizzabile a scopo didattico per approfondire i concetti della sintesi FM.

# 7. Bibliografia


1. **Autori**: Edd Dumbill e Neil M. Bornstein
   **Titolo**: Mono: A Developer's Notebook
   **Editore**: O' Reilly Media
   **Edizione:** Luglio 2004

2. **Autore**: Erick Brown
   **Titolo**: Windows Forms Programm With C#
   **Editore**: Manning
   **Edizione**: 2002 - Greenwich

3. **Autore**: John Sharp
   **Titolo**: Microsoft Visual C# 2008 Passo per passo
   **Editore**: Mondadori Informatica
   **Edizione**: 2008 – Redmond,Washington

4. **Autori**: Riccardo Bianchini – Alessandro Cipriani
   **Titolo**: Il Suono Virtuale
   **Editore**: ConTempoNet
   **Edizione**: 2011 – Roma

5. **Autore:** J.M. Chowning
   **Titolo:** The Synthesis of Complex Audio Spectra by Means of Frequency Modulations
   **Editore**: J Audio Eng. Soc.
   **Edizione**: Stanford (California) 21.07.1973.