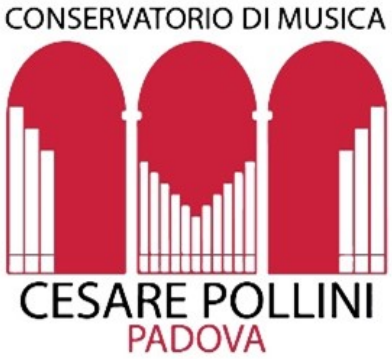

Tesi di diploma accademico di I livello in Musica Elettronica indirizzo tecnico di sala di registrazione

Distorsione: implementazione di un processore di segnale audio digitale per la ricerca e lo studio del fenomeno

DCPL: 34

Diplomando

Gabriel Bizzo

Matricola

12424

Relatore

Prof. Julian Scordato

Anno Accademico 2020-2021

Gabriel Bizzo: Matricola 12424

Distorsione: implementazione di un processore di segnale audio digitale per la ricerca e lo studio del fenomeno, Tesi di diploma accademico di I livello, © Novembre 2021.

Sommario

Il presente documento descrive il lavoro svolto per la tesi da parte del laureando Gabriel Bizzo. Il lavoro prevede l'approfondimento della distorsione di un segnale audio come effetto desiderato, utilizzabile sia a scopo creativo che tecnico per la produzione musicale.
Lo studio in questione prevede un'iniziale ricerca a livello teorico sui principi, sulle diverse tipologie di distorsione possibili e l'analisi di alcune implementazioni digitali esistenti di questo fenomeno.
Successivamente nel documento viene descritta la realizzazione del processore di segnale audio digitale denominato *Biztortion*, il quale implementa diversi algoritmi di distorsione del segnale, sia strettamente legati al mondo digitale che come emulazione di fenomeni analogici, come singoli moduli concatenabili a piacere in un'unica semplice interfaccia grafica.
Infine viene presentata una panoramica sulle licenze software, identificando quelle utilizzate dalle librerie esterne come condizione per il loro utilizzo e definendo la licenza utilizzata per il rilascio del software sopraccitato.

Ringraziamenti

Innanzitutto desidero ringraziare con affetto i miei genitori e tutta la mia famiglia per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante questi difficili anni di studio. Ci tengo in particolare a dedicare questo traguardo a mia nonna Teresa, con la speranza che possa essere fiera del mio percorso.

Vorrei esprimere inoltre la mia gratitudine al Prof. Julian Scordato, relatore della mia tesi, per la disponibilità e l'aiuto fornitomi nello svolgimento del lavoro descritto nel presente documento.

Ho desiderio di ringraziare infine tutti i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, Novembre 2021 Gabriel Bizzo

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Motivazione ed obiettivi

La *distorsione* del segnale audio in ambito creativo, dettata da una continua ricerca di sonorità particolari e ricche di armoniche, è per me sempre stata fonte inesauribile di curiosità e divertimento. Questa continua sperimentazione sonora mi ha portato negli anni ad appassionarmi molto alla musica elettronica dance e, in particolare, al sottogenere denominato *Hardstyle*, tipico del nord Europa e dalla sonorità molto aggressiva e distorta. Un ulteriore fattore, fondamentale per la mia scelta di approfondire la distorsione come argomento per la tesi di laurea, è sicuramente la passione per lo *sviluppo software*, alimentata dal mio percorso universitario nel corso di laurea in scienze informatiche da poco conclusosi.

Dopo una lunga riflessione, accomunando le mie passioni e le mie competenze apprese nel mio percorso di studi, ho deciso di approfondire la tematica della distorsione attraverso l'implementazione di un processore di segnale audio digitale, in modo da poter alimentare la mia ambizione di diventare uno sviluppatore di software per le applicazioni multimediali.

Oltre a dire quanto per me questo percorso di ricerca sia risultato incredibilmente appagante, vorrei infine manifestare la mia contentezza nel poter finalmente contribuire nelle community di produttori di musica elettronica e di sviluppatori software per l'audio offrendo diversi algoritmi di distorsione in un unico strumento open-source.

1.2 Strumenti utilizzati

Microsoft Visual Studio

Microsoft Visual Studio è un ambiente di sviluppo integrato creato e manutenuto da Microsoft. Lo strumento in questione è multi-linguaggio e attualmente supporta la creazione di progetti per varie piattaforme, tra cui anche Mobile e Console. È possibile creare ed utilizzare estensioni e componenti aggiuntivi.

Xcode

Xcode è un ambiente di sviluppo integrato completamente sviluppato e mantenuto da Apple, contenente una suite di strumenti utili allo sviluppo di software per i sistemi macOS, iOS, iPadOS, watchOS e tvOS.

Git

Git è uno strumento per il controllo di versione distribuito utilizzabile da interfaccia a riga di comando. E' possibile utilizzare il software in questione per collaborare con più membri di un team e per controllare la versione del codice prodotto così da poter ritornare ad una versione stabile in caso di problemi.

Adobe Illustrator / Photoshop

I due programmi in questione sono due software proprietari prodotti da Adobe e sono specializzati nell'elaborazione di immagini digitali. Questi strumenti sono stati utilizzati per la realizzazione di alcune immagini utilizzate nell'interfaccia grafica del software per rendere più piacevole l'esperienza dell'utente.

Draw.io

Draw.io è una piattaforma online gratuita per la creazione di varie tipologie di diagrammi, esportabili come file in diversi formati (tra i quali PDF o JPEG). Tra le diverse possibilità offerte dal software sono presenti i diagrammi di flusso, di processo,

UML, entità-relazione e di rete. Questa piattaforma è stata utilizzata per creare i diagrammi dei casi d'uso, utilizzati per l'analisi dei requisiti illustrata nel capitolo 3.1.

Balsamiq Mockups

Balsamiq Mockups è uno strumento di progettazione dell'interfaccia utente per la creazione di mockup (ovvero dei prototipi a bassa fedeltà). E' possibile utilizzare questo strumento per generare schizzi digitali di varie idee di prodotto per facilitare la discussione e la comprensione prima che venga scritto qualsiasi codice.

Overleaf

Overleaf è un editor LaTeX collaborativo basato su cloud e viene utilizzato per scrivere, modificare e pubblicare varie tipologie di documenti. Questo software è stato utilizzato per la scrittura del presente documento.

Inno Setup / Packages

I due programmi in questione sono dei sistemi di installazione guidati da script. Questi software sono stati utilizzati per la creazione dei pacchetti di installazione guidata rispettivamente per Windows e MacOS.

1.3 Organizzazione del testo

1.3.1 Struttura del documento

Il documento, suddiviso in cinque capitoli, è strutturato nella seguente modalità:

Il primo capitolo effettua una breve introduzione al lavoro e agli strumenti utilizzati per la realizzazione della tesi di laurea.

Il secondo capitolo approfondisce il fenomeno della distorsione del segnale audio descrivendo i principi teorici e le diverse tipologie di algoritmi esistenti, oltre ad analizzare alcune implementazioni digitali presenti sul mercato.

Il terzo capitolo descrive le diverse fasi che hanno permesso la realizzazione del processore di segnale audio digitale denominato Biztortion.

Il quarto capitolo effettua una panoramica sulle licenze software, analizzando quelle utilizzate dalle librerie terze necessarie all'implementazione del software Biztortion e quindi quella utilizzata per il rilascio del suddetto software.

Il quinto capitolo , infine, contiene un'analisi del lavoro svolto e le conclusioni tratte.

1.3.2 Convenzioni tipografiche

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

* gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
* per la prima occorrenza dei termini riportati nel glossario viene utilizzata la nomenclatura "*parola(abbreviazione)*", mentre per ogni successiva occorrenza verrà utilizzata solamente l'abbreviazione di tale termine;
* i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Distorsione

In questo capitolo viene approfondito il fenomeno della distorsione del segnale audio descrivendo alcuni principi teorici fondamentali, analizzando diverse tecniche esistenti e alcune implementazioni digitali presenti sul mercato

2.1 Introduzione al fenomeno

Per *distorsione* di un segnale si intende l'*alterazione della forma d'onda* originale, causando la modifica dell'informazione che tale segnale rappresenta. Il fenomeno in questione può risultare indesiderato, come nel caso delle telecomunicazioni in cui se viene prodotto un disturbo nella ricezione del segnale viene alterata l'informazione originariamente trasmessa, oppure desiderato, come nel caso dell'ambito musicale in cui la distorsione viene utilizzata come strumento per la ricerca e il sound design.
Il termine distorsione, seppur a livello teorico assuma questo significato, nel contesto musicale viene usato principalmente per riferirsi alla distorsione non lineare e in particolare all'introduzione di nuove componenti frequenziali mediante non linearità[1].
La *distorsione lineare* consiste in qualsiasi tipo di alterazione della forma d'onda che non genera nuove frequenze rispetto al segnale originale. Alcuni esempi di questa categoria di distorsione sono l'alterazione dell'ampiezza del segnale (comunemente chiamata regolazione del volume), dipendente dalla frequenza, di un amplificatore e

[1] Gary J. Louie Glenn D. White. *The Audio Dictionary*. 3ª ed. University of Washington Press, 2005. ISBN: 0-295-98498-8.

la quantità di attenuazione, sempre dipendente dalla frequenza, da parte di un filtro elettrico[2].

Sempre nel contesto musicale esistono diverse fonti di *distorsione non lineare*, alcune delle quali vengono approfondite nella successiva sezione, e la più comune consiste nel *clipping*, ovvero la saturazione di un sistema per la regolazione dell'ampiezza del segnale audio, ottenibile utilizzando circuiti elettrici analogici (per esempio un amplificatore di segnale quando gli viene richiesto di produrre un livello che eccede i suoi limiti di progettazione[3]) oppure attraverso delle emulazioni digitali.

2.2 Tecniche di distorsione

In questa sezione vengono analizzate alcune tecniche di distorsione *non lineari* del segnale audio, tutte implementate nel software Biztortion attraverso appositi algoritmi descritti nel terzo capitolo del presente documento.

2.2.1 Clipping

Il *clipping* consiste in un processo non lineare che produce frequenze non originariamente presenti nel segnale audio. Queste frequenze possono essere armoniche, nel senso che sono multipli interi della frequenza fondamentale del segnale originale, o disarmoniche, ovvero slegate da qualsiasi rapporto con le frequenze del segnale originario. La produzione di armoniche nel segnale è un fenomeno chiamato distorsione armonica, mentre le frequenze spurie possono venire introdotte nel segnale a causa dei comportamenti non lineari nell'elaborazione del segnale utilizzando sia apparecchiature analogiche che algoritmi digitali, a causa del fenomeno della DIM. Infatti, ad esempio, suonare un power chord con la chitarra elettrica attraverso un clipper provoca, oltre all'arricchimento dello spettro sonoro con l'aggiunta delle armoniche del segnale originario, anche un'intermodulazione che produce nuove subarmoniche. Di seguito vengono descritte due diverse modalità di clipping del segnale:

[2] M. Kleiner. *Acoustics and Audio Technology*. 3ª ed. J. Ross Publishing, 2011. ISBN: 9781604270525, 1604270527.

[3] R. Jones G. Davis. *The Sound Reinforcement Handbook*. 2ª ed. Yamaha, 1989. ISBN: 9780881889000.

* **Soft clipping**:

  - effettua un *appiattimento graduale* dei picchi di un segnale, creando un numero di armoniche più alte che condividono una relazione con il timbro originale;

  - il soft clipping è ottenibile utilizzando strumentazione analogica ed avviene quando si aumenta un segnale ad una tensione di picco più elevata rispetto a quella che un determinato dispositivo riesce a gestire. Un esempio può essere la distorsione attraverso la saturazione di un amplificatore a *valvole*. Essenzialmente l'aumento del voltaggio spinge le valvole oltre la propria regione di funzionamento lineare, ossia quella regione in cui ad un aumento del voltaggio segue un aumento proporzionale dell'ampiezza del segnale;

  - questo fenomeno, molto utilizzato per aggiungere carattere al suono delle chitarre elettriche, è ottenibile anche nel mondo digitale implementando un algoritmo che emuli il comportamento della strumentazione analogica.

* **Hard clipping**:

  - effettua un *appiattimento brusco* dei picchi di un segnale, con conseguente maggiore potenza nelle armoniche prodotte dal segnale originario. All'aumentare del clipping, il segnale in ingresso inizia progressivamente ad assomigliare a un'onda quadra con armoniche dispari. Questa viene generalmente descritta come una sonorità più aspra;

  - l'hard clipping è ottenibile facilmente nel mondo digitale in quanto si verifica quando un segnale viene amplificato oltre 0 dBFS in qualsiasi supporto digitale, come il tipico convertitore analogico/digitale o digitale/analogico. In questi casi, 0 dBFS è il valore più alto in assoluto che il computer può gestire e al di sopra di questo livello le informazioni vengono scartate con il risultato di tagliare la parte superiore della forma d'onda. Ovviamente risulta possibile inoltre implementare un algoritmo che tagli automaticamente ogni informazione del segnale sopra ad una soglia impostata dall'utente;

– questo fenomeno risulta ottenibile non solo in digitale ma anche utilizzando strumentazione analogica. Un esempio consiste nell'utilizzo degli *amplificatori a stato solido* che incorporano transistor e/o amplificatori operazionali. Dal punto di vista elettronico, ciò si ottiene solitamente amplificando il segnale fino a un punto in cui viene tagliato dalla limitazione della tensione del binario di alimentazione o clippando il segnale utilizzando i diodi.

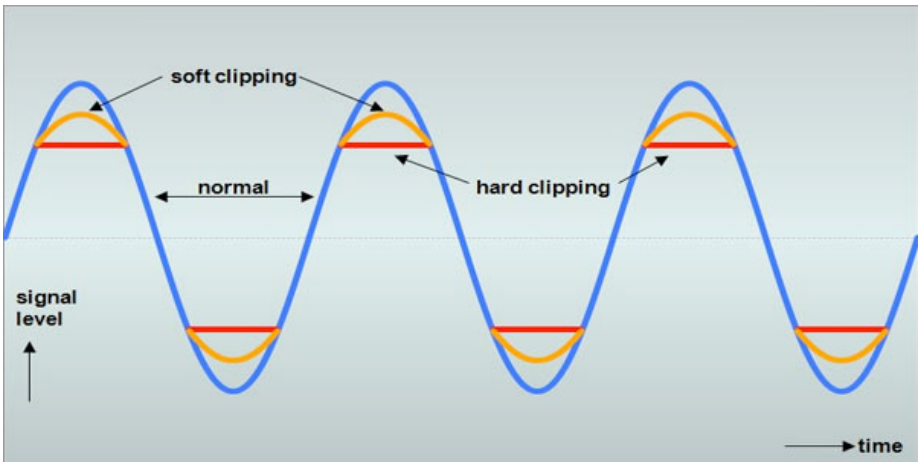

Figura 2.1: Il fenomeno del Clipping

2.2.2 Waveshaping

La tecnica di distorsione del segnale *Waveshaping* viene descritta in modo preciso e puntuale nel libro *The Theory and Technique of Electronic Music*[4] come un passaggio del segnale originale attraverso una funzione non lineare opportunamente scelta. La funzione f(), chiamata *funzione di trasferimento*, distorce la forma d'onda in ingresso in una forma diversa. La nuova forma dipende dalla forma dell'onda in input, dalla funzione di trasferimento e anche, in modo cruciale, dall'ampiezza del segnale in ingresso. Poiché l'ampiezza della forma d'onda in input influenza direttamente la forma della forma d'onda in output (e quindi il timbro), questo dà la possibilità di creare una famiglia di timbri che varia continuamente, semplicemente variando il livello di ingresso della trasformazione. Per questo motivo, è consuetudine includere un controllo di ampiezza iniziale come parte dell'operazione di waveshaping.

[4]Miller Puckette. *The Theory and Technique of Electronic Music*. World Scientific Publishing Co. Pte. Ltd., 2006.

L'ampiezza della forma d'onda in ingresso è chiamata *indice* della forma d'onda. In molte situazioni un indice piccolo porta a una distorsione relativamente piccola (in modo che l'output assomigli molto all'input) e uno più grande dà un timbro più distorto e più ricco.

La figura 2.2 mostra un esempio di waveshaping in cui f() equivale a una funzione di clipping. Questo esempio mostra chiaramente come l'indice può influenzare la forma d'onda di output. La funzione di clipping **(b)** passa il suo input direttamente all'output senza variarlo fintanto che rimane nell'intervallo tra - 0.3 e +0.3. Quindi, quando l'input non supera 0,3 in valore assoluto, l'output è uguale all'input. Tuttavia quando l'input cresce oltre i limiti, l'output viene fatto rimanere entro essi e, all'aumentare dell'ampiezza del segnale, l'effetto di questa azione di clipping diventa progressivamente più importante. Nella figura 2.2 il segnale di input è una sinusoide smorzata **(a)** e l'output **(c)** evolve nell'asse x da una forma d'onda quasi quadra all'inizio ad una sinusoide pura alla fine.

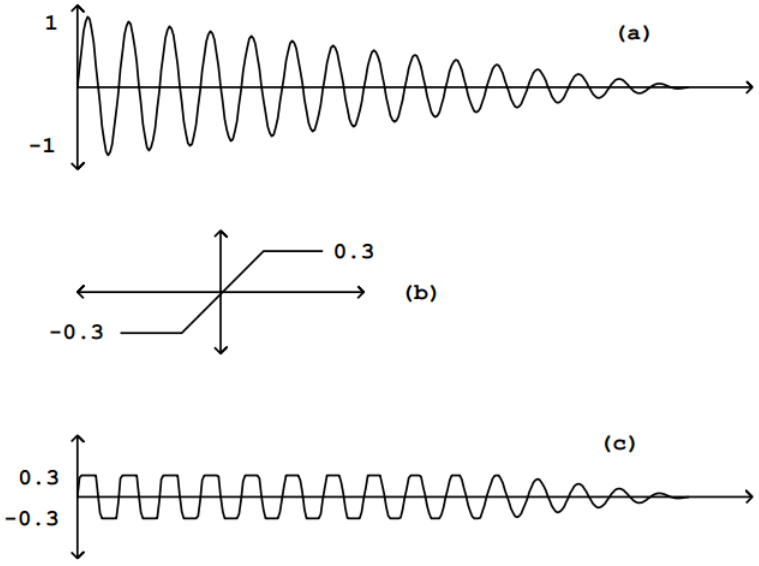

Figura 2.2: Componenti del Waveshaper

2.2.3 Bitcrushing

Il *Bitcrusher* consiste in un processore di segnale che produce distorsione *riducendo la risoluzione dei dati audio digitali.* Gli artefatti introdotti utilizzando questa

tecnica possono produrre una sonorità più calda o aspra, a seconda della quantità di degradazione del segnale, andando a simulare il suono prodotto dai vecchi campionatori disponibili all'inizio dell'era digitale.
Il bitcrusher utilizza principalmente due metodi per ridurre la fedeltà audio: la riduzione della frequenza di campionamento e la riduzione della profondità di bit.

* **Frequenza di campionamento**: secondo la teoria del campionamento[5], l'audio digitale è composto da una rapida serie di campioni numerici, acquisiti in modo periodico da un apposito dispositivo chiamato convertitore analogico/digitale, i quali codificano l'ampiezza variabile di una forma d'onda audio. Per una rappresentazione accurata di un segnale audio si necessità di una *frequenza di campionamento* elevata, in quanto maggiore è la frequenza di campionamento, più accurata risulta la forma d'onda. Infatti nel caso in cui il campionamento avvenga ad una frequenza troppo bassa le componenti frequenziali più elevate vengono codificate con troppe poche informazioni e la loro ricostruzione produce degli artefatti. Questo fenomeno, chiamato *aliasing*, avviene nel caso in cui la frequenza di campionamento risulta minore del doppio della frequenza massima del segnale (Teorema del campionamento di Nyquist-Shannon). Sfruttando questo fenomeno il bitcrusher effettua una riduzione graduale, attraverso un apposito parametro, della frequenza di campionamento, introducendo quindi frequenze non armonicamente legate al segnale originale;

* **Profondità di bit**: i singoli campioni nell'audio digitale vengono solitamente registrati come numeri in *virgola mobile* e memorizzati nella memoria digitale utilizzando la *codifica binaria*. In questo modo maggiore è il numero di bit disponibili per rappresentare il livello di ampiezza dei singoli campioni, più accuratamente le forme d'onda vengono campionate. La riduzione della risoluzione effettuata dal bitcrusher riduce intenzionalmente il numero di bit utilizzati per i campioni audio. Man mano che la profondità di bit diminuisce, le forme d'onda diventano più rumorose e si perdono sottili variazioni di volume, riducendo quindi la gamma dinamica. Alla riduzione estrema dei bit a disposizione, le

[5] D. Repetto P. Burk L. Polansky. *Music and Computers*. Key College Puslishing, 2005. ISBN: 1-930190-95-6.

forme d'onda vengono ridotte a delle onde quadre o addirittura a singoli click, aggiungendo quindi molte alte frequenze nello spettro sonoro.

Figura 2.3: Riduzione della risoluzione del segnale in un Bitcrusher

2.2.4 Slew Limiting

Lo *Slew Limiter* consiste in un processore di segnale audio, solitamente presente come singolo modulo nei sintetizzatori modulari analogici, che permette di livellare un segnale in ingresso in modo che la variazione del livello di tensione non possa superare un certo numero di volt al secondo. Per questo motivo il processore in questione viene solitamente utilizzato come controller di portamento e può essere utilizzato anche come un semplice generatore di inviluppi Attack-Release.

Una emulazione digitale di questo particolare processore è implementata nel software Biztortion ed è ispirata al modulo Eurorack *Slew Limiter*[6] dell'azienda Befaco, del quale esiste un'implementazione digitale per il software VCV Rack distribuita con licenza GPL-3.0. In particolare in questo algoritmo risulta possibile impostare la soglia di velocità massima sopra alla quale essa stessa viene limitata nel tempo sul segnale in uscita, sia nella fase in cui l'onda si muove verso la zona positiva (*Rise*), sia durante la fase in cui l'onda procede nella direzione opposta (*Fall*), andando in questo modo a distorcere la forma d'onda dell'audio.

[6] *Befaco Slew Limiter*. URL: https://library.vcvrack.com/Befaco/SlewLimiter.

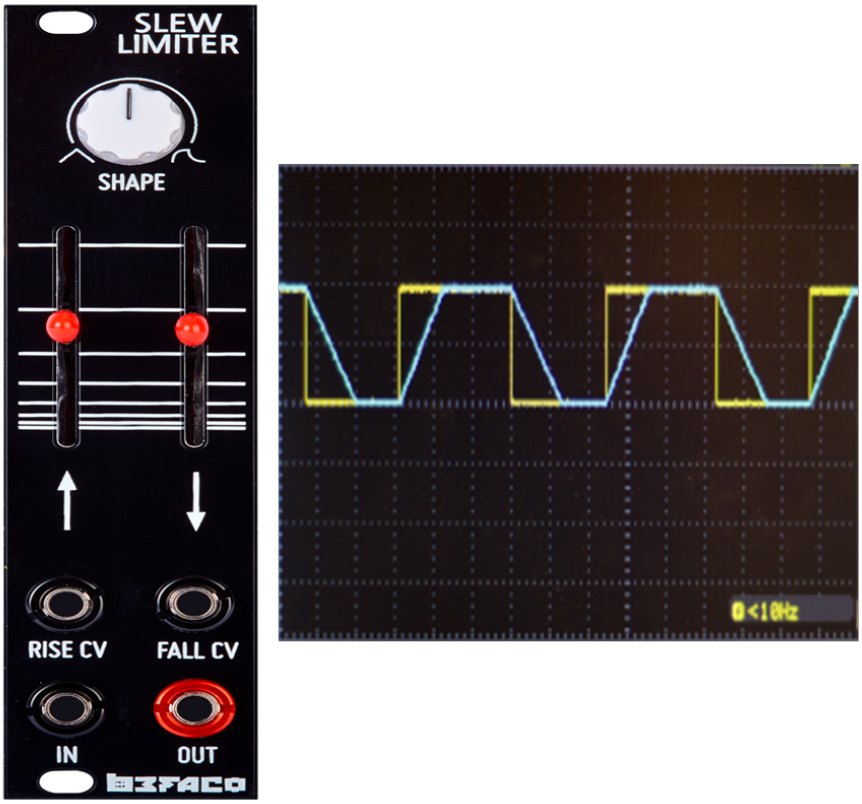

Figura 2.4: Funzionamento del modulo Slew Limiter dell'azienda Befaco: i segnali in ingresso e uscita sono rispettivamente giallo ed azzurro

2.3 Studio dello stato dell'arte

In questa sezione viene effettuata un'analisi di alcuni prodotti software presenti sul mercato che offrono diverse tipologie di algoritmi e strumenti utili per la distorsione del segnale audio.

Questi prodotti sono stati utilizzati come fonte d'ispirazione per la creazione dell'interfaccia grafica del software *Biztortion*, utili prevalentemente per identificare il layout più funzionale per massimizzare la user experience. Questa ricerca si è rivelata utile inoltre per effettuare un'analisi dei pregi e difetti dei processori di segnale in questione, in modo da poter ricreare nel modo più fedele possibile le funzionalità utili e allo stesso tempo effettuare degli aggiustamenti o aggiunte ad alcuni algoritmi nel software sopraccitato.

Izotope Trash 2

Izotope Trash 2[7] consiste in un processore di segnale audio digitale, disponibile come plugin audio in diversi formati, che consente una trasformazione sonora complessa grazie alla presenza di un elevato numero di moduli che offrono diverse funzionalità.

Figura 2.5: Izotope Trash 2

I moduli offerti dal software sono i seguenti:

* **Pre/Post-filtering**: i moduli per il filtraggio del segnale consentono di modellare lo spettro sonoro utilizzando fino a 6 filtri, i quali possono essere configurati con oltre 20 modelli differenti. Sono inoltre utilizzabili degli LFO, inviluppi e il sidechain per la modulazione dei parametri;

* **Trash**: il modulo in questione consente la distorsione del segnale utilizzando oltre 60 funzioni di trasferimento per effettuare waveshaping in due stadi, con la possibilità di applicare l'effetto a tutto lo spettro o fino a 4 bande di frequenze differenti;

* **Convolve**: attraverso il modulo per la convoluzione del segnale Trash 2 consente una simulazione realistica di diversi altoparlanti ed acustiche, permettendo di posizionare completamente l'audio in un altro luogo o oggetto;

[7] *Izotope Trash 2*. URL: https://www.izotope.com/en/products/trash.html.

* **Dynamics**: consente la compressione multibanda del segnale utilizzando fino a 4 bande di frequenze differenti;

* **Delay**: consente l'aggiunta dell'effetto Echo offrendo fino a 6 algoritmi diversi.

Kilohearts Toolbox

Kilohearts Toolbox[8] consiste in un pacchetto, del quale esistono diverse versioni in base al budget che si vuole investire nel suo acquisto, che offre vari strumenti che fanno parte dell'ecosistema Kilohearts. In questo pacchetto sono compresi diversi plugin audio, utilizzabili singolarmente all'interno della DAW desiderata, e l'host di effetti modulari gratuito, Snap Heap, per poter combinare i singoli plugin in rack di effetti seriali e/o paralleli.

Figura 2.6: Kilohearts Snap Heap

[8] *Kilohearts Toolbox*. URL: https://kilohearts.com/products/kilohearts_toolbox.

Tra l'elevato numero di effetti presenti nel pacchetto in questione sono compresi i seguenti plugin che effettuano la distorsione del segnale:

* **Distortion**: consiste in un processore di segnale in cui è possibile utilizzare 5 diversi algoritmi, i quali utilizzano diverse combinazioni di waveshaping e filtri, per la distorsione dell'audio in ingresso. Risulta inoltre possibile aggiungere un DC offset al segnale prima del processing e, utilizzando un apposito potenziometro, differenziare l'offset stesso tra i due canali sinistro e destro, in modo da creare dei sottili ampliamenti della stereofonia;

* **Bitcrush**: permette di simulare l'audio come se fosse riprodotto utilizzando un campionatore di bassa qualità, con frequenza di campionamento e profondità di bit limitate. Questo processore di segnale consente inoltre l'aggiunta di rumore in modo effettuare del dithering per ridurre il rumore causato dalla quantizzazione, oltre all'utilizzo di appositi filtri per la diminuzione dell'aliasing causato dalla riduzione della qualità del campionamento (sia ad alte che basse frequenze);

* **Phase Distortion**: consiste in un processore audio che consente al segnale di modulare la fase di se stesso, risultando sostanzialmente in qualcosa di simile al feedback FM. Questo effetto infatti non modifica l'ampiezza del segnale, come effettuano invece i distorsori tradizionali, ma trasforma la fase di singole armoniche usando l'input stesso come modulante. Poichè l'input influenza direttamente l'algoritmo per la produzione del segnale in uscita, l'effetto risultante è sempre originale e non può suonare uguale in due moduli diversi.

Fabfilter Saturn 2

Fabfilter Saturn 2[9] consiste in un processore di segnale audio digitale, disponibile come plugin audio in diversi formati, che offre una vasta gamma di algoritmi di distorsione ispirati al mondo analogico vintage degli amplificatori a valvole e dei nastri magnetici. Sono inoltre presenti diversi algoritmi basati su differenti tecniche di distorsione come il bitcrushing o la granularizzazione del segnale.

[9] *Fabfilter Saturn 2*. URL: https://www.fabfilter.com/products/saturn-2-multiband-distortion-saturation-plug-in.

Figura 2.7: Fabfilter Saturn 2

Le funzionalità salienti del software sono le seguenti:

* **Display multibanda interattivo**: il display interattivo consente di creare e selezionare direttamente le bande di frequenza e, allo stesso tempo, è un analizzatore di frequenza in tempo reale, permettendo di visualizzare il segnale in uscita e rendendo facile decidere dove impostare le frequenze di crossover della banda;

* **Controlli delle bande**: i controlli delle bande controllano le impostazioni delle bande selezionate nel display. Per ogni banda, è possibile regolare separatamente il tipo di distorsione, il drive, le impostazioni di feedback, le dinamiche, il tono, il livello e le impostazioni di mix;

* **Sezione di modulazione**: la sezione di modulazione, presente nella parte inferiore dell'interfaccia mostra tutte le sorgenti disponibili per la modulazione dei parametri, ovvero LFO, envelope generator (EG), envelope follower (EF), controller MIDI e XY;

* **Fase lineare**: quando la modalità "Linear Phase" è abilitata, sia il filtro crossover multibanda che l'oversampling interno, attivabile attraverso la modalità "High Quality", vengono eseguiti utilizzando un filtraggio a fase lineare.

Capitolo 3

Biztortion

In questo capitolo viene descritta la realizzazione del processore di segnale audio digitale denominato Biztortion, il quale implementa i diversi algoritmi di distorsione del segnale descritti nel capitolo precedente. Nel capitolo in questione verranno trattate in particolare l'analisi dei requisiti, la progettazione e la codifica del plugin audio sopraccitato, in modo da coprire tutte le fasi del lavoro svolto e ottenere un prodotto software di qualità.

3.1 Analisi dei requisiti

3.1.1 Contesto di utilizzo

Il software Biztortion, implementando diverse tipologie di algoritmi per la distorsione del segnale audio, viene sviluppato e distribuito come plugin audio utilizzabile da qualsiasi utente all'interno della propria DAW preferita.
Il processore di segnale in questione è pensato quindi per effettuare della **ricerca sonora** nel modo più creativo possibile, fornendo vari strumenti molto utili principalmente per la produzione di musica elettronica dance (e per questo è stata scelta un'elaborazione di **segnale audio stereofonico**), senza però escludere un possibile utilizzo più tecnico e legato alla post-produzione.

3.1.2 Attori

Il sistema ha come attore principale un **utente generico**, sul quale non vengono effettuate particolari supposizioni in quanto l'utente, una volta installato l'applicativo sul proprio computer e caricato sulla DAW desiderata, ha la possibilità di accedere a tutte le funzionalità che offre il sistema stesso. Inoltre è utile sottolineare come il prodotto software, essendo open-source, non necessita di effettuare particolari controlli e/o autenticazioni agli utenti che lo utilizzano.

3.1.3 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi di tipo UML per descrivere i casi d'uso[1] (*UC*) del software Biztortion.
I casi d'uso consistono in degli scenari di utilizzo del prodotto software e sono dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso.

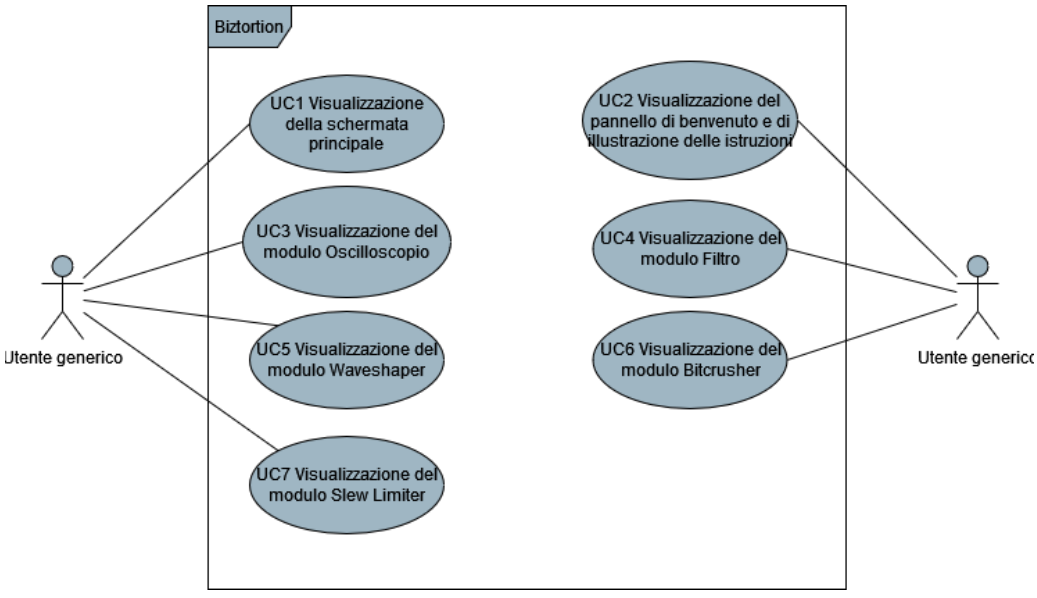

Figura 3.1: Diagramma UML contenente i casi d'uso che riguardano la visualizzazione dei componenti dell'interfaccia

[1] *Casi D'uso*. URL: https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20Use%20Case_4x4.pdf.

UC1: Visualizzazione della schermata principale

Attori Principali: Utente generico.

Precondizioni: L'utente generico ha caricato con successo il plugin nella propria DAW.

Descrizione: L'utente generico visualizza i seguenti componenti nell'interfaccia principale del plugin:

* la *Module Section*[**UC1.1**], ovvero una sezione dell'interfaccia riservata alla visualizzazione dei moduli per l'audio processing o al pannello di benvenuto[**UC2**];
* la *Chain Section*[**UC1.2**], ovvero una sezione dell'interfaccia riservata alla visualizzazione della catena di *ChainPosition*, i quali possono rappresentare un modulo già istanziato [**UC1.2.1**] o non ancora instanziato [**UC1.2.2**];
* la *Input Section*[**UC1.3**], ovvero una sezione dell'interfaccia riservata al controllo del guadagno e il monitoraggio del segnale in input;
* la *Output Section*[**UC1.4**], ovvero una sezione dell'interfaccia riservata al controllo del guadagno e il monitoraggio del segnale in output;
* il nome del plugin come titolo sulla parte superiore dell'interfaccia;
* un link che indirizza l'utente al profilo Github di Gabriel Bizzo per visualizzare il codice sorgente del software e varie informazioni extra sul progetto;
* un pulsante che genera un pannello di benvenuto all'utente[**UC2**], contenente inoltre le istruzioni basilari per l'utilizzo del plugin, sulla *Module Section*.

Postcondizioni: L'utente generico visualizza la schermata principale del software.

UC1.1: Visualizzazione della Module Section

Attori Principali: Utente generico.

Precondizioni: L'utente generico sta visualizzando la schermata principale del software.

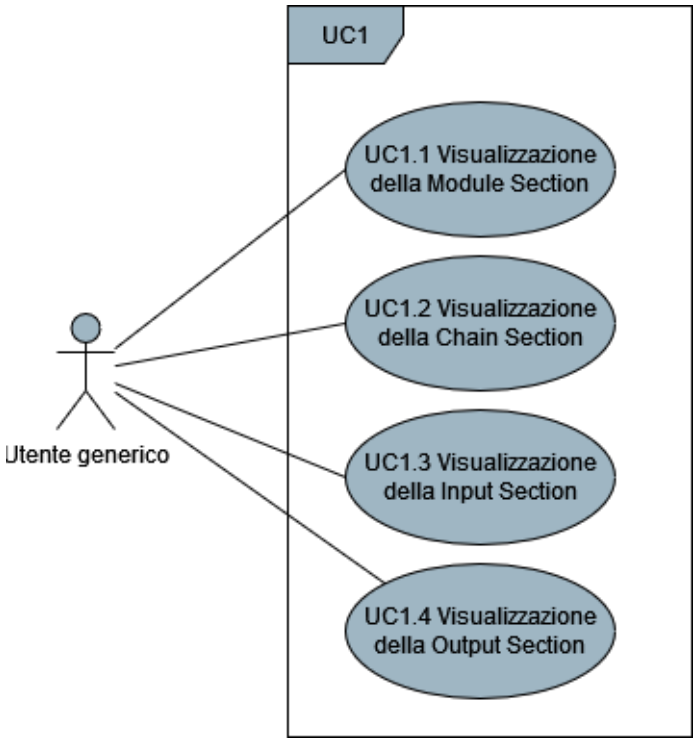

Figura 3.2: Use Case - UC1: Visualizzazione della schermata principale

Descrizione: L'utente generico nella schermata denominata *Module Section* visualizza due possibili contenuti:

* un *modulo per il processing del segnale audio* [**UC3-7**];
* un *pannello di benvenuto* all'utente [**UC2**].

Postcondizioni: L'utente generico visualizza il contenuto della Module Section.

UC1.2: Visualizzazione della Chain Section

Attori Principali: Utente generico.

Precondizioni: L'utente generico sta visualizzando la schermata principale del software.

Descrizione: L'utente generico nella schermata denominata *Chain Section* visualizza un insieme di otto Chain Position, posizionati in ordine uno di fianco all'altro. Questi elementi contenuti in questa sezione possono essere di due tipologie:

* una Chain Position che rappresenta un modulo *istanziato* e presente nella catena di elaborazione dell'audio [**UC1.2.1**];
* una Chain Position che rappresenta un modulo *NON istanziato*, e quindi non effettua alcuna operazione sul segnale audio [**UC1.2.2**].

Postcondizioni: L'utente generico visualizza il contenuto della Chain Section.

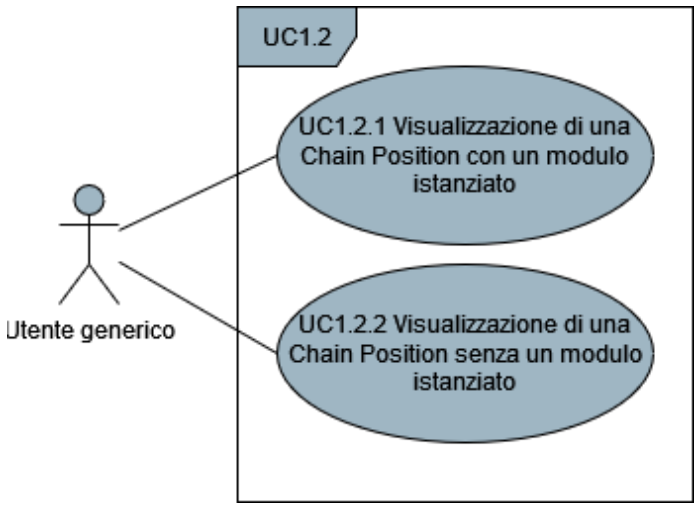

Figura 3.3: Use Case - UC1.2: Visualizzazione della Chain Section

UC1.2.1: Visualizzazione di una Chain Position con un modulo istanziato

Attori Principali: Utente generico.

Precondizioni: L'utente generico sta visualizzando una Chain Position che rappresenta un modulo attualmente istanziato nella catena di elaborazione del segnale.

Descrizione: L'utente generico visualizza i seguenti componenti contenuti nella sezione in questione:

* il numero di posizione nella catena di elaborazione del segnale;
* il pulsante per l'eliminazione del modulo attualmente istanziato dalla catena di elaborazione dell'audio [**UC11**];
* il pulsante per l'accesso al modulo attualmente istanziato [**UC12**].

Postcondizioni: L'utente generico visualizza il contenuto della Chain Position rappresentante un modulo attualmente istanziato nella catena di elaborazione del segnale.

UC1.2.2: Visualizzazione di una Chain Position senza un modulo istanziato

Attori Principali: Utente generico.

Precondizioni: L'utente generico sta visualizzando una Chain Position che non rappresenta nessun modulo istanziato nella catena di elaborazione del segnale.

Descrizione: L'utente generico visualizza i seguenti componenti contenuti nella sezione in questione:

* il numero di posizione nella catena di elaborazione del segnale;
* il pulsante per l'istanziazione di un nuovo modulo nella catena di elaborazione dell'audio alla posizione indicata dal numero del punto precedente [**UC10**].

Postcondizioni: L'utente generico visualizza il contenuto della Chain Position non rappresentante alcun modulo, corrispondente dunque ad una posizione libera nella catena di elaborazione dell'audio.

UC1.3: Visualizzazione della Input Section

Attori Principali: Utente generico.

Precondizioni: L'utente generico sta visualizzando la schermata principale del software.

Descrizione: L'utente generico nella schermata denominata *Input Section* visualizza i seguenti componenti:

* il titolo della sezione in questione;
* un misuratore di livello del segnale in ingresso al software;

* un potenziometro per la modifica del guadagno del segnale in ingresso al software.

Postcondizioni: L'utente generico visualizza i componenti costitutivi della Input Section.

UC1.4: Visualizzazione della Output Section

Attori Principali: Utente generico.

Precondizioni: L'utente generico sta visualizzando la schermata principale del software.

Descrizione: L'utente generico nella schermata denominata *Output Section* visualizza i seguenti componenti:

* il titolo della sezione in questione;
* un misuratore di livello del segnale in uscita al software;
* un potenziometro per la modifica del guadagno del segnale in uscita al software.

Postcondizioni: L'utente generico visualizza i componenti costitutivi della Output Section.

UC2: Visualizzazione del pannello di benvenuto e di illustrazione delle istruzioni

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza la schermata principale del software e si verifica una delle seguenti condizioni:

* l'utente effettua l'istanziazione del software nella propria DAW;
* l'utente elimina un modulo dalla catena di elaborazione del segnale [**UC11**];
* non sono presenti moduli istanziati nella catena di elaborazione del segnale;

* l'utente clicca sul pulsante, visualizzabile nella schermata principale [**UC1**], per visualizzare le istruzioni del plugin in questione.

.

Descrizione: L'utente generico visualizza i seguenti componenti nella *Module Section*:

* il nome e il logo del software;
* un messaggio di benvenuto;
* un breve elenco di istruzioni utili all'utente per capire il funzionamento del plugin audio.

Postcondizioni: L'utente generico visualizza una sezione contenente un messaggio di benvenuto ed un elenco di istruzioni utili per il corretto utilizzo del software.

UC3: Visualizzazione del modulo Oscilloscopio

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza la schermata principale del software e si verifica una delle seguenti condizioni:

* l'utente effettua l'istanziazione del modulo Oscilloscopio in una specifica Chain Position [**UC10**];
* l'utente accede al modulo Oscilloscopio, attualmente istanziato in una specifica Chain Position, attraverso l'apposito pulsante [**UC12**].

.

Descrizione: L'utente generico visualizza i seguenti componenti, corrispondenti al contenuto del modulo Oscilloscopio, nella *Module Section*:

* il titolo del modulo;
* un *oscilloscopio* per visualizzare la forma d'onda del segnale in ingresso;
* i parametri utilizzabili dall'utente per la modifica della visualizzazione del segnale audio[**UC14-15**], e in particolare:

  - il pulsante *bypass*;
  - un pulsante *freeze*;
  - un potenziometro *zoom verticale*;
  - un potenziometro *zoom orizzontale*.

Postcondizioni: L'utente generico visualizza tutte le informazioni, parametri e componenti grafici necessari al corretto utilizzo del modulo Oscilloscopio.

UC4: Visualizzazione del modulo Filtro

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza la schermata principale del software e si verifica una delle seguenti condizioni:

* l'utente effettua l'istanziazione del modulo Filtro in una specifica Chain Position [**UC10**];
* l'utente accede al modulo Filtro, attualmente istanziato in una specifica Chain Position, attraverso l'apposito pulsante [**UC12**].

.

Descrizione: L'utente generico visualizza i seguenti componenti, corrispondenti al contenuto del modulo Filtro, nella *Module Section*:

* il titolo del modulo;
* un *analizzatore di spettro* per visualizzare lo spettro frequenziale del segnale;
* un pulsante *spectrum analyzer*;
* un componente grafico per visualizzare la *curva di risposta del filtro*, la quale comprende i filtri Passa Basso, Passa Banda e Passa Alto;
* i parametri utilizzabili dall'utente per la modifica del segnale audio[**UC16**], e in particolare:

- il pulsante *bypass*;
- un potenziometro *Low Pass frequency*;
- un potenziometro *Low Pass order*;
- un potenziometro *Peak frequency*;
- un potenziometro *Peak gain*;
- un potenziometro *Peak quality*;
- un potenziometro *High Pass frequency*;
- un potenziometro *High Pass order*.

Postcondizioni: L'utente generico visualizza tutte le informazioni, parametri e componenti grafici necessari al corretto utilizzo del modulo Filtro.

UC5: Visualizzazione del modulo Waveshaper

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza la schermata principale del software e si verifica una delle seguenti condizioni:

* l'utente effettua l'istanziazione del modulo Waveshaper in una specifica Chain Position [**UC10**];
* l'utente accede al modulo Waveshaper, attualmente istanziato in una specifica Chain Position, attraverso l'apposito pulsante [**UC12**].

.

Descrizione: L'utente generico visualizza i seguenti componenti, corrispondenti al contenuto del modulo Waveshaper, nella *Module Section*:

* il titolo del modulo;
* un grafico che rappresenta la *funzione di trasferimento* applicata al segnale in ingresso;

* i parametri utilizzabili dall'utente per la modifica del segnale audio[**UC17**], e in particolare:
  - il pulsante *bypass*;
  - un potenziometro *drive*;
  - un potenziometro *mix*;
  - un potenziometro *symmetry*;
  - un potenziometro *bias*;
  - un potenziometro *tanh amplitude*;
  - un potenziometro *tanh slope*;
  - un potenziometro *sine amplitude*;
  - un potenziometro *sine frequency.*

Postcondizioni: L'utente generico visualizza tutte le informazioni, parametri e componenti grafici necessari al corretto utilizzo del modulo Waveshaper.

UC6: Visualizzazione del modulo Bitcrusher

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza la schermata principale del software e si verifica una delle seguenti condizioni:

* l'utente effettua l'istanziazione del modulo Bitcrusher in una specifica Chain Position [**UC10**];
* l'utente accede al modulo Bitcrusher, attualmente istanziato in una specifica Chain Position, attraverso l'apposito pulsante [**UC12**].

.

Descrizione: L'utente generico visualizza i seguenti componenti, corrispondenti al contenuto del modulo Bitcrusher, nella *Module Section*:

* il titolo del modulo;

* i parametri utilizzabili dall'utente per la modifica del segnale audio [**UC18**], e in particolare:
  - il pulsante *bypass*;
  - un potenziometro *drive*;
  - un potenziometro *mix*;
  - un potenziometro *symmetry*;
  - un potenziometro *bias*;
  - un potenziometro *rate reduction*;
  - un potenziometro *bit reduction*;
  - un potenziometro *dithering*.

Postcondizioni: L'utente generico visualizza tutte le informazioni, parametri e componenti grafici necessari al corretto utilizzo del modulo Bitcrusher.

UC7: Visualizzazione del modulo Slew Limiter

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza la schermata principale del software e si verifica una delle seguenti condizioni:

* l'utente effettua l'istanziazione del modulo Slew Limiter in una specifica Chain Position [**UC10**];
* l'utente accede al modulo Slew Limiter, attualmente istanziato in una specifica Chain Position, attraverso l'apposito pulsante [**UC12**].

.

Descrizione: L'utente generico visualizza i seguenti componenti, corrispondenti al contenuto del modulo Slew Limiter, nella *Module Section*:

* il titolo del modulo;

- i parametri utilizzabili dall'utente per la modifica del segnale audio [**UC19**], e in particolare:
  - il pulsante *bypass*;
  - un potenziometro *drive*;
  - un potenziometro *mix*;
  - un potenziometro *symmetry*;
  - un potenziometro *bias*;
  - un potenziometro *rise*;
  - un potenziometro *fall*;
  - un pulsante a doppio stato *DC filter*.

Postcondizioni: L'utente generico visualizza tutte le informazioni, parametri e componenti grafici necessari al corretto utilizzo del modulo Slew Limiter.

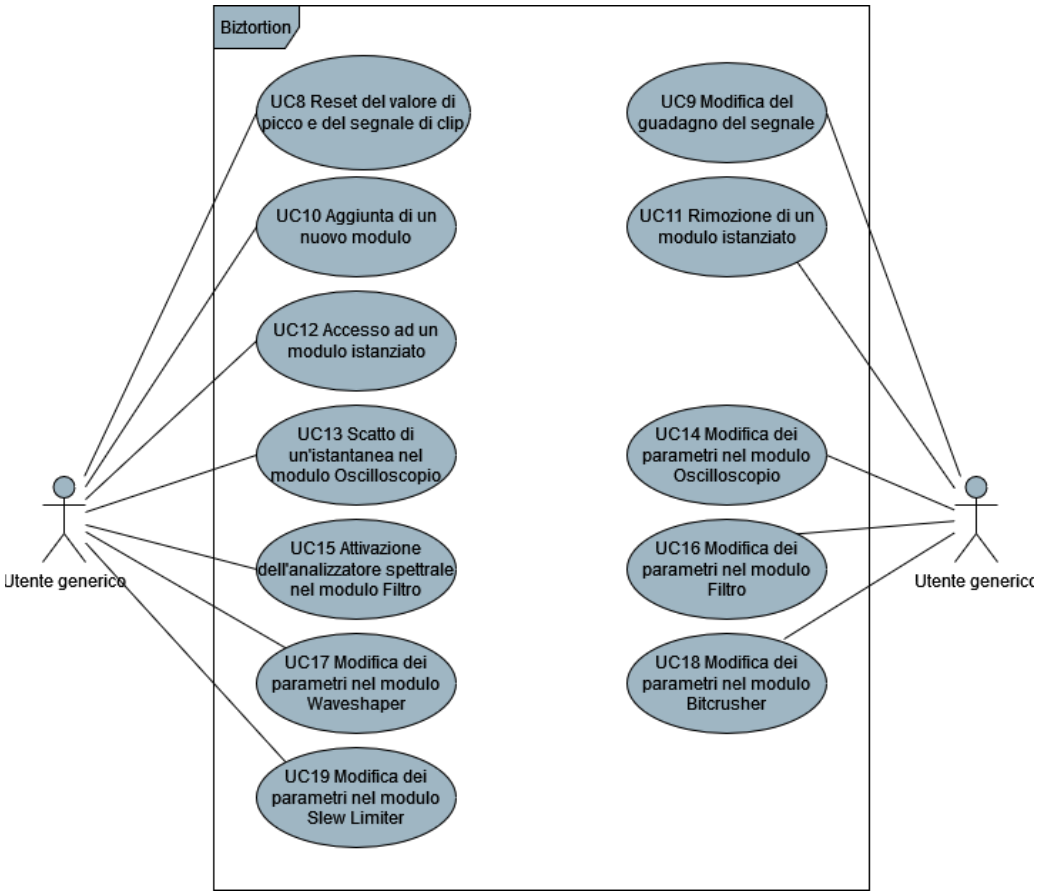

Figura 3.4: Diagramma UML contenente i casi d'uso che riguardano le funzionalità offerte dai moduli istanziabili nel sistema

UC8: Reset del valore di picco e del segnale di clip

Attori Principali: Utente generico.

Precondizioni: L'utente generico sta visualizzando la *Input Section* o la *Output Section*.

Descrizione: L'utente generico, visualizzando il misuratore di livello, ha la possibilità di resettare il massimo valore di picco rilevato, andando inoltre a disabilitare il segnale di clip (rosso nel caso in cui il segnale supera i 0 dBFS).

Postcondizioni: L'utente generico effettua il reset del valore di picco e del segnale di clip nel misuratore di livello presente nella *Input Section* o nella *Output Section*.

UC9: Modifica del guadagno del segnale

Attori Principali: Utente generico.

Precondizioni: L'utente generico sta visualizzando la *Input Section* o la *Output Section*.

Descrizione: L'utente generico, visualizzando il potenziometro relativo al guadagno del segnale, ha la possibilità di modificare il guadagno del segnale in ingresso nella sezione in questione.

Postcondizioni: L'utente generico effettua la modifica del guadagno del segnale in ingresso nella *Input Section* o nella *Output Section*.

UC10: Aggiunta di un nuovo modulo

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza la *Chain Section*[**UC1.2**] ed ha selezionato una Chain Position senza un modulo istanziato[**UC1.2.2**].

Descrizione: L'utente generico, con l'obiettivo di istanziare un nuovo modulo nella *Chain Position* desiderata, deve eseguire le seguenti operazioni:

1. cliccare sul pulsante, presente nella *Chain Position* in questione, per istanziare un nuovo modulo nella catena di elaborazione del segnale;
2. selezionare il modulo desiderato dal menù a tendina comparso nella *Module Section*.

Dopo aver completato la procedura appena descritta il modulo risulterà istanziato nella catena di elaborazione del segnale nella posizione descritta dal numero della *Chain Position* in questione e il modulo risulterà visibile ed interagibile nella *Module Section*.

Postcondizioni: L'utente generico istanzia un nuovo modulo nella catena di elaborazione del segnale, il quale diventa visibile ed interagibile attraverso l'interfaccia grafica del software.

UC11: Rimozione di un modulo istanziato

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza la *Chain Section*[**UC1.2**] ed ha selezionato una Chain Position con un modulo attualmente istanziato[**UC1.2.1**].

Descrizione: L'utente generico, con l'obiettivo di eliminare il modulo attualmente istanziato nella *Chain Position* desiderata, deve eseguire le seguenti operazioni:

1.

2. cliccare sul pulsante, presente nella *Chain Position* in questione, per eliminare il modulo attualmente istanziato dalla catena di elaborazione del segnale.

Dopo aver completato la procedura appena descritta il modulo risulterà rimosso dalla catena di elaborazione del segnale e dalla *Module Section*.

Postcondizioni: L'utente generico rimuove un modulo precedentemente istanziato nella catena di elaborazione del segnale.

UC12: Accesso ad un modulo istanziato

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza la *Chain Section*[**UC1.2**] ed ha selezionato una Chain Position con un modulo attualmente istanziato[**UC1.2.1**].

Descrizione: L'utente generico, con l'obiettivo di accedere all'interfaccia grafica per interagire con il modulo attualmente istanziato nella *Chain Position* desiderata, deve eseguire le seguenti operazioni:

1.

2. cliccare sul pulsante, presente nella *Chain Position* in questione, per accedere al modulo istanziato dalla catena di elaborazione del segnale.

Dopo aver completato la procedura appena descritta il modulo risulterà visibile ed interagibile nella *Module Section*.

Postcondizioni: L'utente generico accede ad un modulo istanziato nella catena di elaborazione del segnale.

UC13: Scatto di un'istantanea nel modulo Oscilloscopio

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza il modulo Oscilloscopio [**UC3**], il quale

risulta istanziato in una determinata *Chain Position* e quindi presente nella catena di elaborazione del segnale.

Descrizione: L'utente generico ha la possibilità di effettuare un'istantanea sul componente grafico corrispondente all'oscilloscopio cliccando sull'apposito pulsante *freeze*.

Postcondizioni: L'utente generico effettua un'istantanea sul componente grafico corrispondente all'oscilloscopio.

UC14: Modifica dei parametri nel modulo Oscilloscopio

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza il modulo Oscilloscopio[**UC3**], il quale risulta istanziato in una determinata *Chain Position* e quindi presente nella catena di elaborazione del segnale.

Descrizione: L'utente generico ha la possibilità influenzare l'azione del modulo sulla catena di elaborazione interagendo con i suoi parametri. In particolare le modifiche effettuabili dall'utente sono le seguenti:

* il pulsante *bypass* permette di controllare l'attivazione del modulo sulla catena di elaborazione del segnale;
* il potenziometro *zoom verticale* permette l'aggiustamento della visualizzazione verticale della forma d'onda;
* *zoom orizzontale* permette l'aggiustamento della visualizzazione orizzontale della forma d'onda.

Postcondizioni: L'utente generico, interagendo con i parametri del modulo Oscilloscopio, modifica il segnale nella modalità descritta precedentemente.

UC15: Attivazione dell'analizzatore spettrale nel modulo Filtro

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza il modulo Filtro [**UC4**], il quale risulta istanziato in una determinata *Chain Position* e quindi presente nella catena di elaborazione del segnale.

Descrizione: L'utente generico ha la possibilità di attivare l'analizzatore dello spettro frequenziale del segnale cliccando sull'apposito pulsante *spectrum analyzer*.

Postcondizioni: L'utente generico attiva l'analizzatore dello spettro frequenziale.

UC16: Modifica dei parametri nel modulo Filtro

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza il modulo Filtro[**UC4**], il quale risulta istanziato in una determinata *Chain Position* e quindi presente nella catena di elaborazione del segnale.

Descrizione: L'utente generico ha la possibilità influenzare l'azione del modulo sulla catena di elaborazione interagendo con i suoi parametri. In particolare le modifiche effettuabili dall'utente sono le seguenti:

* il pulsante *bypass* permette di controllare l'attivazione del modulo sulla catena di elaborazione del segnale;
* il potenziometro *Low Pass frequency* permette l'aggiustamento della frequenza del filtro IIR Passa Basso;
* il potenziometro *Low Pass order* permette l'aggiustamento dell'ordine del filtro IIR Passa Basso;
* il potenziometro *Peak frequency* permette l'aggiustamento della frequenza del filtro IIR Passa Banda;
* il potenziometro *Peak gain* permette l'aggiustamento del guadagno del filtro IIR Passa Banda;
* il potenziometro *Peak quality* permette l'aggiustamento della qualità del filtro IIR Passa Banda;
* il potenziometro *High Pass frequency* permette l'aggiustamento della frequenza del filtro IIR Passa Alto;

* il potenziometro *High Pass order* permette l'aggiustamento dell'ordine del filtro IIR Passa Alto.

Postcondizioni: L'utente generico, interagendo con i parametri del modulo Filtro, modifica il segnale nella modalità descritta precedentemente.

UC17: Modifica dei parametri nel modulo Waveshaper

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza il modulo Waveshaper[**UC5**], il quale risulta istanziato in una determinata *Chain Position* e quindi presente nella catena di elaborazione del segnale.

Descrizione: L'utente generico ha la possibilità influenzare l'azione del modulo sulla catena di elaborazione interagendo con i suoi parametri. In particolare le modifiche effettuabili dall'utente sono le seguenti:

* il pulsante *bypass* permette di controllare l'attivazione del modulo sulla catena di elaborazione del segnale;
* il potenziometro *drive* permette di gestire il guadagno del segnale in ingresso;
* il potenziometro *mix* permette il dosaggio tra il segnale originale e quello modificato;
* il potenziometro *symmetry* permette il dosaggio dell' effetto dato dalla distorsione nelle due diverse zone dell'onda sonora, ovvero nella fase positiva e negativa;
* il potenziometro *bias* permette di determinare la separazione tra la zona positiva e quella negativa dell'onda sonora in punti diversi rispetto all'origine;
* il potenziometro *tanh amplitude* permette di determinare l'ampiezza della funzione arcotangente utilizzata nella funzione di trasferimento;
* il potenziometro *tanh slope* permette di determinare l'inclinazione della funzione arcotangente utilizzata nella funzione di trasferimento;

* il potenziometro *sine amplitude* permette di determinare l'ampiezza della funzione sinusoidale utilizzata nella funzione di trasferimento;
* il potenziometro *sine frequency* permette di determinare la frequenza della funzione sinusoidale utilizzata nella funzione di trasferimento.

Postcondizioni: L'utente generico, interagendo con i parametri del modulo Waveshaper, modifica il segnale nella modalità descritta precedentemente.

UC18: Modifica dei parametri nel modulo Bitcrusher

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza il modulo Bitcrusher[**UC6**], il quale risulta istanziato in una determinata *Chain Position* e quindi presente nella catena di elaborazione del segnale.

Descrizione: L'utente generico ha la possibilità influenzare l'azione del modulo sulla catena di elaborazione interagendo con i suoi parametri. In particolare le modifiche effettuabili dall'utente sono le seguenti:

* il pulsante *bypass* permette di controllare l'attivazione del modulo sulla catena di elaborazione del segnale;
* il potenziometro *drive* permette di gestire il guadagno del segnale in ingresso;
* il potenziometro *mix* permette il dosaggio tra il segnale originale e quello modificato;
* il potenziometro *symmetry* permette il dosaggio dell' effetto dato dalla distorsione nelle due diverse zone dell'onda sonora, ovvero nella fase positiva e negativa;
* il potenziometro *bias* permette di determinare la separazione tra la zona positiva e quella negativa dell'onda sonora in punti diversi rispetto all'origine;
* il potenziometro *rate reduction* permette di determinare la riduzione della frequenza di campionamento utilizzata per la degradazione del campionamento del segnale;

* il potenziometro *bit reduction* permette di determinare la riduzione della profondità di bit utilizzata per la degradazione della quantizzazione del segnale;

* il potenziometro *dithering* permette di determinare la percentuale di rumore bianco da introdurre nel segnale per distorcere ulteriormente il segnale oppure per effettuare del dithering, ammorbidendo quindi gli artefatti introdotti con la degradazione del segnale.

Postcondizioni: L'utente generico, interagendo con i parametri del modulo Bitcrusher, modifica il segnale nella modalità descritta precedentemente.

UC19: Modifica dei parametri nel modulo Slew Limiter

Attori Principali: Utente generico.

Precondizioni: L'utente generico visualizza il modulo Slew Limiter[**UC7**], il quale risulta istanziato in una determinata *Chain Position* e quindi presente nella catena di elaborazione del segnale.

Descrizione: L'utente generico ha la possibilità influenzare l'azione del modulo sulla catena di elaborazione interagendo con i suoi parametri. In particolare le modifiche effettuabili dall'utente sono le seguenti:

* il pulsante *bypass* permette di controllare l'attivazione del modulo sulla catena di elaborazione del segnale;

* il potenziometro *drive* permette di gestire il guadagno del segnale in ingresso;

* il potenziometro *mix* permette il dosaggio tra il segnale originale e quello modificato;

* il potenziometro *symmetry* permette il dosaggio dell' effetto dato dalla distorsione nelle due diverse zone dell'onda sonora, ovvero nella fase positiva e negativa;

* il potenziometro *bias* permette di determinare la separazione tra la zona positiva e quella negativa dell'onda sonora in punti diversi rispetto all'origine;

* il potenziometro *rise* permette di scegliere la soglia di velocità massima sopra alla quale essa stessa viene limitata nel tempo sul segnale in uscita durante la fase di salita dell'onda sonora;

* il potenziometro *fall* permette di scegliere la soglia di velocità massima sopra alla quale essa stessa viene limitata nel tempo sul segnale in uscita durante la fase di discesa dell'onda sonora;

* il pulsante a doppio stato *DC filter* permette di determinare l'inserimento di un filtro Passa Alto a bassa frequenza, il quale consente di eliminare ogni tipo di DC offset sul segnale in ingresso.

Postcondizioni: L'utente generico, interagendo con i parametri del modulo Slew Limiter, modifica il segnale nella modalità descritta precedentemente.

3.2 Progettazione e codifica

3.2.1 Tecnologie utilizzate

C++

Il C++ è un linguaggio di programmazione general purpose, ovvero è caratterizzato da una certa versatilità e quindi adatto a molti impieghi. Il linguaggio in questione è stato sviluppato in origine da Bjarne Stroustrup nei Bell Labs nel 1983 come evoluzione del linguaggio C inserendo la programmazione orientata agli oggetti; col tempo ha avuto notevoli evoluzioni, come l'introduzione dell'astrazione.
Poiché uno degli obiettivi del lavoro descritto in questo documento risulta la realizzazione di un processore di segnale digitale, si è resa necessaria la scelta di un linguaggio di programmazione adeguato alla sfida dell'elaborazione audio in tempo reale.
La scelta di questo linguaggio di programmazione per la realizzazione del software *Biztortion* è stata effettuata per i suoi molti punti a favore[2], alcuni dei quali sono i seguenti:

[2] *C++ features*. URL: https://www.cplusplus.com/info/description/.

Figura 3.5: C++ Logo

* è un linguaggio *standardizzato ISO*, ovvero manutenuto in una modalità ben definita e strutturata da una apposita commissione della International Organization for Standardization;

* è un linguaggio che *compila* direttamente nel codice nativo di una macchina, permettendogli di essere uno dei linguaggi più veloci al mondo se ottimizzato;

* è un linguaggio *portatile*, in quanto essendo uno dei linguaggi più utilizzati al mondo e un linguaggio aperto, il C++ ha una vasta gamma di compilatori che funzionano su molte piattaforme diverse che lo supportano. Il codice che utilizza esclusivamente la libreria standard di C++ verrà eseguito su molte piattaforme con poche o nessuna modifica;

* ha un incredibile *supporto per le librerie* da parte della community, in quanto una ricerca di "libreria" sul popolare sito Web di gestione dei progetti SourceForge produrrà oltre 3000 risultati per le librerie C++.

JUCE

In seguito alla scelta del linguaggio di programmazione si è resa necessaria l'individuazione di una libreria C++ che permettesse la creazione di applicazioni per il processing audio e l'esportazione in un formato compatibile con i plugin utilizzabili

all'interno delle DAW più comuni.

Con questi obiettivi in mente è stato selezionato il framework JUCE.

JUCE[3] è un framework applicativo C++ multipiattaforma e parzialmente open-source, utilizzato per lo sviluppo di applicazioni desktop e mobili e risulta molto diffuso per il suo ampio insieme di funzionalità audio e di creazione di user interface. L'obiettivo di JUCE è consentire la scrittura del software in modo che lo stesso codice sorgente venga compilato ed eseguito in modo identico su piattaforme Windows, macOS e Linux. Il framework in questione supporta inoltre vari ambienti di sviluppo e compilatori.

Figura 3.6: JUCE Logo

Infine è utile notare che JUCE fornisce classi wrapper per la creazione di plugin audio e browser. Quando si crea un plugin audio, viene prodotto un singolo file binario che supporta più formati di plugin (VST e VST3, RTAS, AAX, Audio Units). Poiché tutto il codice specifico della piattaforma e del formato è contenuto nel wrapper, un utente può creare VST/VST3/RTAS/AAX/AU sia per MacOS che per Windows da un'unica codebase.

Virtual Studio Technology

Virtual Studio Technology (VST)[4] consiste in un'interfaccia software per plugin audio che integra sintetizzatori software e unità di effetti nelle workstation audio digitali. VST e tecnologie simili utilizzano l'elaborazione del segnale digitale per simulare l'hardware di uno studio di registrazione tradizionale nel software. Esistono migliaia di plugin, sia commerciali che gratuiti, e molte applicazioni audio supportano VST su licenza della sua azienda creatrice, ovvero Steinberg.

I plugin VST generalmente vengono eseguiti all'interno di una DAW per fornire fun-

[3] *JUCE website*. URL: https://juce.com/.

[4] *Steinberg VST website*. URL: https://www.steinberg.net/technology/.

zionalità aggiuntive, sebbene esistano alcuni host di plugin autonomi che supportano VST. La maggior parte dei plugin VST sono strumenti (VSTi) o effetti (VSTfx), sebbene esistano altre categorie, ad esempio analizzatori di spettro e misuratori di livello. I plugin VST di solito forniscono un'interfaccia utente grafica personalizzata che visualizza controlli simili a interruttori e manopole fisici sull'hardware audio. Risulta infine utile notare che gli strumenti VST ricevono note come informazioni digitali tramite MIDI ed emettono audio digitale, mentre gli effetti VST ricevono l'audio digitale, lo elaborano ed alla fine lo emettono attraverso le loro uscite.

Audio Units

Gli *Audio Units* (AU) sono un'architettura per i plugin a livello di sistema fornita da Core Audio nei sistemi operativi macOS e iOS di Apple. Gli Audio Units sono un insieme di servizi Application Program Interface forniti dal sistema operativo per generare, elaborare, ricevere o manipolare flussi di audio in tempo reale con una latenza minima. Può essere pensato come l'equivalente architettonico di Apple del formato per plugin VST appena descritto.

3.2.2 Progettazione della maschera

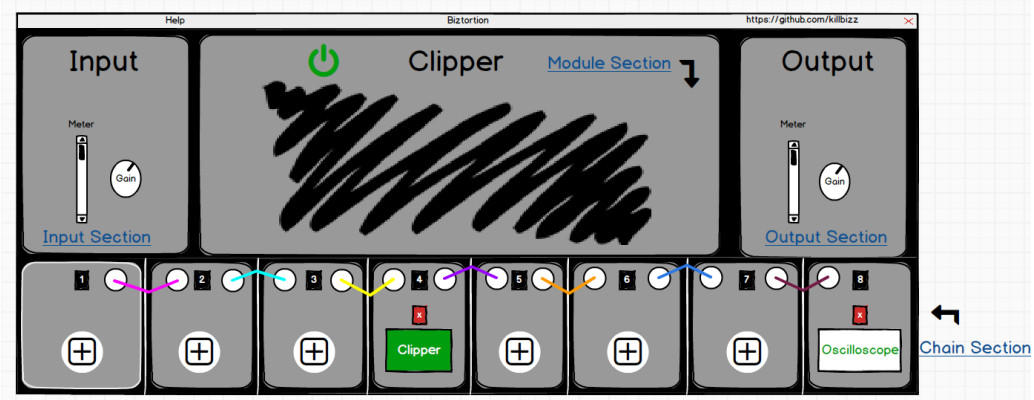

Figura 3.7: Mockup della maschera principale del software Biztortion

Prima di procedere con la codifica del software è stata effettuata la progettazione della maschera principale, in modo da avere un'idea di base da cui partire e successivamente ampliare per la realizzazione dell'interfaccia grafica in base alle necessità. Questo processo ha aiutato a definire in modo più accurato le componenti necessarie

per il corretto funzionamento del software, identificando le componenti denominate *Module Section*, *Chain Section*, *Chain Position*, *Input Section* e *Output Section*: le componenti in questione sono state quindi delineate e inserite nell'analisi dei requisiti. Infine risulta utile notare che il mockup ottenuto, raffigurato nella figura 3.7, è stato realizzato con lo strumento *Balsamiq Mockups*, descritto nella sezione 1.2.

3.2.3 Architettura

Il software in questione consiste in un plugin audio realizzato utilizzando il framework **JUCE** sia per la parte logica di DSP che per la parte di vista, ovvero la user interface. Il progetto di base, creato con lo strumento Projucer, ha consentito lo sviluppo del software con un unico linguaggio di programmazione, ovvero il **C++**, grazie alla possibilità di creare diverse build nelle piattaforme Windows, MacOS e Linux. Nello specifico il software è stato esportato in formato **VST3** su **Windows** utilizzando lo strumento Microsoft Visual Studio, e in formato **AU** su **MacOS** attraverso lo strumento Xcode.

I concetti fondamentali per l'implementazione di un plugin audio utilizzando il framework JUCE sono le classi *juce::AudioProcessor* e *juce::AudioProcessorEditor*. Queste due classi corrispondono relativamente al processore di segnale audio (DSP) e all'editor grafico (GUI) e, implementandole attraverso due classi concrete, rispettivamente *BiztortionAudioProcessor* e *BiztortionAudioProcessorEditor*, si ha la possibilità di definire i comportamenti del software una volta caricato nella DAW desiderata.

E' possibile notare nella figura 3.8 il **diagramma dei package** del software Biztortion, il quale permette di avere una chiara visione della struttura del progetto e delle strutture di supporto utilizzate all'interno delle classi *BiztortionAudioProcessor* e *BiztortionAudioProcessorEditor* per definire il comportamento del software.
Gli elementi fondamentali dell'architettura sono i seguenti:

* **Shared**: contiene tutte le strutture e classi di supporto utilizzate nei diversi moduli e component. Al suo interno sono presenti tutte le definizioni ed implementazioni dei componenti di utilità personalizzati per comporre la user

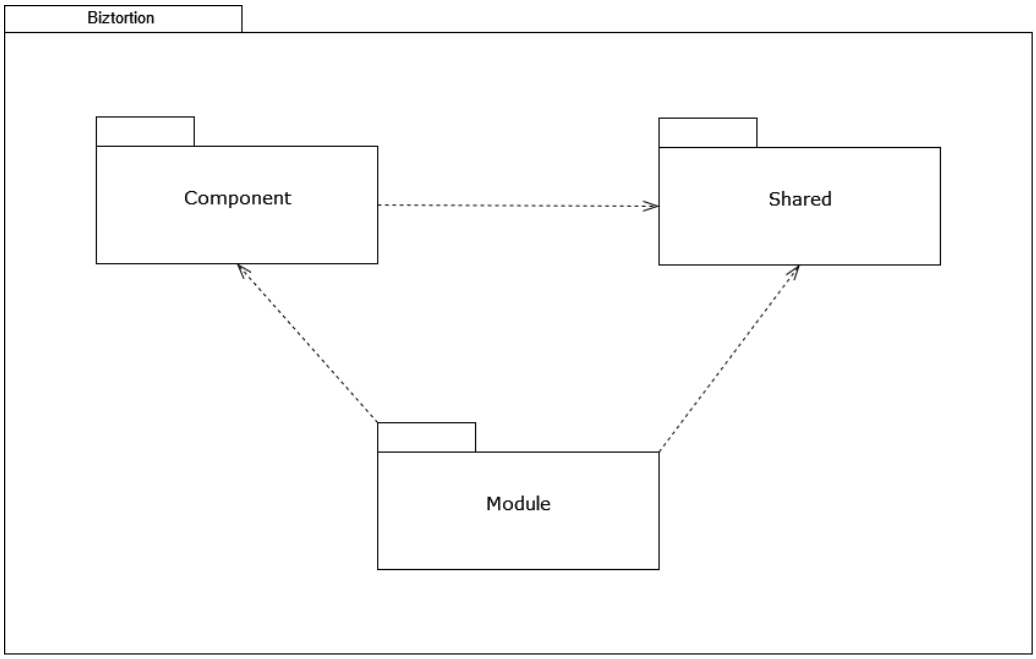

Figura 3.8: Diagramma dei package del software Biztortion

interface (potenziometri e bottoni) oltre che ai componenti logici necessari per l'analisi della FFT;

* **Component**: contiene tutti i componenti grafici personalizzati, i quali implementano la classe juce::Component e vengono utilizzati per illustrare informazioni in tempo reale all'interno di diversi moduli. In particolare sono presenti il FFTAnalyzerComponent, ResponseCurveComponent e TransferFunctionComponent;

* **Module**: contiene tutti i moduli istanziabili nella *Chain Section*. Ogni modulo implementa due classi che permettono di standardizzare sia la parte di DSP (*DSPmodule*) sia la parte relativa all'interfaccia grafica (*GUImodule*). Entrambe le classi, DSPmodule e GUImodule, vengono utilizzate rispettivamente da BiztortionAudioProcessor e BiztortionAudioProcessorEditor sfruttando il concetto di polimorfismo, tipico della programmazione ad oggetti e disponibile col linguaggio di programmazione c++.

3.2.4 Funzionalità implementate

Il codice prodotto ha permesso di soddisfare la totalità dei requisiti, ovvero delle funzionalità necessarie evidenziate dai casi d'uso riscontrati durante l'analisi dei requisiti. Con questo risultato si è arrivato ad avere un funzionamento completo del software *Biztortion*, che verrà brevemente descritto in questa sezione.

Schermata principale

La schermata principale del software è costituita dalle seguenti sezioni fondamentali, definite nel primo caso d'uso:

* **Module Section**: riservata alla visualizzazione dei moduli per l'audio processing o al pannello di benvenuto;
* **Chain Section**: riservata alla visualizzazione della catena di elaborazione del segnale, la quale può contenere moduli istanziati in un ordine arbitrario;
* **Input Section**: riservata al controllo del guadagno e il monitoraggio del segnale in ingresso;
* **Output Section**: riservata al controllo del guadagno e il monitoraggio del segnale in uscita;

Moduli

Tutti i moduli presenti nel software (*Oscilloscope*, *Filter*, *Waveshaper*, *Bitcrusher*, *Slew Limiter*) sono istanziabili dall'utente nella *Chain Section* in un ordine arbitrario e implementano digitalmente le tecniche di distorsione del segnale analizzate nel precedente capitolo:

* **Clipping**: è possibile effettuare il clipping del segnale in ingresso al plugin utilizzando il modulo *Waveshaper* per simulare l'effetto ottenibile con la strumentazione analogica. Grazie a questo modulo infatti risulta possibile utilizzare la funzione arcotangente con il potenziometro "tanh slope" al minimo e, regolando il potenziometro "drive", si può osservare un *Soft Clipping*; per ottenere invece un *Hard Clipping* è sufficiente aumentare in modo considerevole il potenziometro

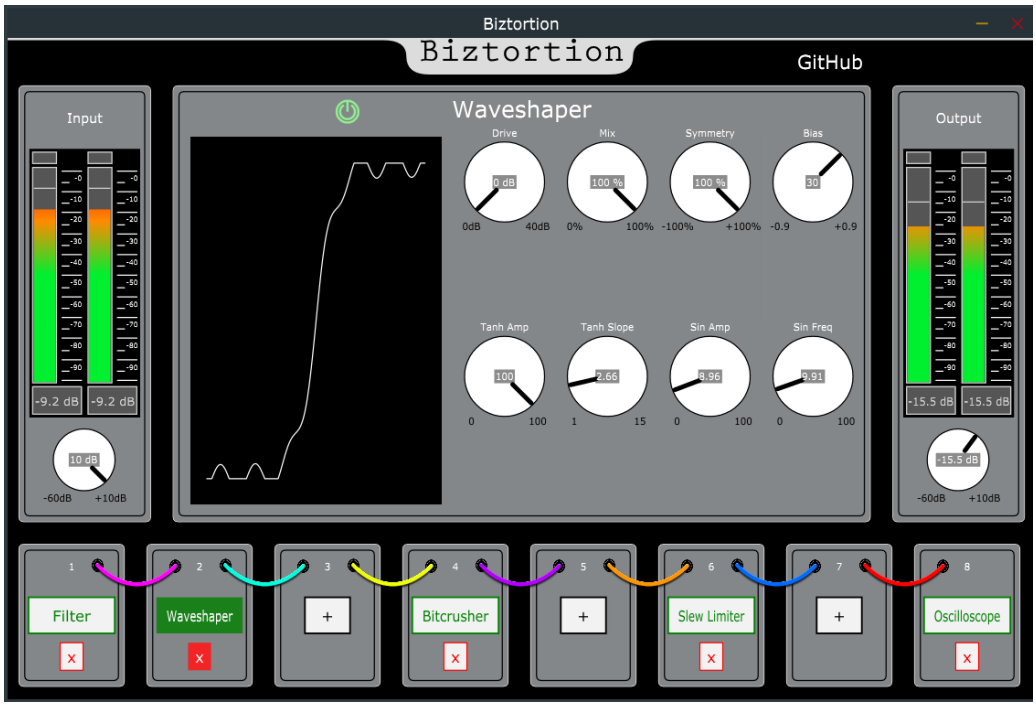

Figura 3.9: Biztortion: schermata principale e visualizzazione del modulo Waveshaper

"drive" in quanto il modulo in questione, utilizzando anche la funzione seno che può causare l'uscita del segnale dai limiti -1/+1, ha un limitatore integrato che provvede a tagliare bruscamente il segnale nel caso superi il limite superiore o inferiore;

* **Waveshaping**: è possibile distorcere il segnale utilizzando la funzione di trasferimento composta presente nel modulo *Waveshaper*, la quale utilizza le funzioni seno ed arcotangente, per creare varie forme d'onda molto complesse;

* **Bitcrushing**: è possibile ridurre la risoluzione del segnale utilizzando il modulo *Bitcrusher* per intervenire sulla frequenza di campionamento e profondità di bit ed introdurre gradualmente gli artefatti digitali tipici dei vecchi campionatori;

* **Slew Limiting**: utilizzando il modulo *Slew Limiter* è possibile impostare la soglia di velocità massima sopra alla quale essa stessa viene limitata nel tempo sul segnale in uscita, sia nella fase in cui l'onda si muove verso la zona positiva, sia durante la fase in cui l'onda procede nella direzione opposta, andando in questo modo a livellare il segnale in ingresso.

Inoltre tutti i moduli che effettuano la distorsione del segnale (*Waveshaper*, *Bitcrusher*, *Slew Limiter*) possono applicare un algoritmo che gli consente di applicare l'effetto a tutta la forma d'onda o ad una sezione di essa, permettendo all'utente di decidere se e quanto effetto applicare in modo simmetrico o asimmetrico.

L'utilizzo combinato dei moduli nella catena di elaborazione permette non solo di creare sonorità molto complesse, ma anche di ricreare effetti già collaudati e frequentemente utilizzati, come per esempio alcuni pedali per la distorsione del suono della chitarra elettrica attraverso le seguenti combinazioni:

* **Overdrive**: utilizzando due moduli Filtro, posizionati prima e dopo un modulo Waveshaper, ed effettuando un'operazione di pre-enfasi *attenuando* le basse frequenze nel primo filtro è possibile ottenere un suono molto simile al tipico effetto overdrive;

* **Fuzz**: utilizzando due moduli Filtro, posizionati prima e dopo un modulo Waveshaper, ed effettuando un'operazione di pre-enfasi *enfatizzando* le basse frequenze nel primo è possibile ottenere un suono molto simile al tipico effetto fuzz.

Risulta utile infine citare che il software *Biztortion* è stato testato manualmente su un sistema Windows 10 Home nelle seguenti DAW:

* Ableton Live Suite 11 (v. 11.0)

* Cycling'74 Max (v. 8.0.2 - 64 bit)

* Fruity Loops Studio (v. 12.1.2 - 64 bit)

* Reaper (v.6.38)

Capitolo 4

Licenze software

Il capitolo in questione effettua una panoramica sulle licenze per il software libero, analizzando quelle utilizzate dalle librerie terze necessarie all'implementazione del software Biztortion e quindi quella utilizzata per il rilascio del suddetto software

4.1 Software libero

Il software Biztortion, corrispondente a parte del lavoro effettuato per la tesi di laurea, è stato sviluppato con l'obiettivo di approfondire il fenomeno della distorsione del segnale audio e di aumentare il mio bagaglio di esperienza da programmatore. Essendo inoltre la tesi di laurea un lavoro di studio e ricerca, risuta logicamente ed eticamente corretto il rilascio e la diffusione del software in questione come *software libero*.

Il "Software libero"[1] rispetta la libertà degli utenti e la comunità. In breve, significa che gli utenti hanno la libertà di eseguire, copiare, distribuire, studiare, modificare e migliorare il software. Tramite queste libertà gli utenti (individualmente o nel loro complesso) controllano il programma e le sue funzioni. Quando non sono gli utenti a controllare il programma, allora il programma (che in quel caso è denominato "non libero" o "proprietario") controlla gli utenti.
Un programma è software libero se gli utenti del programma godono delle quattro libertà fondamentali:

[1] *Definizione di software libero.* URL: https://www.gnu.org/philosophy/free-sw.html.

1. Libertà di **eseguire** il programma come si desidera, per qualsiasi scopo;

2. Libertà di **studiare** come funziona il programma e di **modificarlo** in modo da adattarlo alle proprie necessità. L'accesso al codice sorgente ne è un prerequisito;

3. Libertà di **ridistribuire** copie in modo da aiutare gli altri;

4. Libertà di **migliorare** il programma e distribuirne pubblicamente i miglioramenti apportati (e le versioni modificate in genere), in modo tale che tutta la comunità ne tragga beneficio. Anche qui l'accesso al codice sorgente ne è un prerequisito.

E' necessario inoltre evidenziare il fatto che software libero non vuol dire non commerciale. Al contrario, con un programma libero deve essere possibile anche l'**uso commerciale**, lo sviluppo commerciale, e la distribuzione commerciale. Questa politica è di importanza fondamentale, infatti senza questa il software libero non potrebbe raggiungere i suoi obiettivi.
Infine certi tipi di regole sul come distribuire il software libero sono accettabili quando non entrano in conflitto con le libertà principali. Per esempio, il **copyleft**, noto anche impropriamente come "permesso d'autore", è in poche parole la regola per cui, quando il programma è ridistribuito, non è possibile aggiungere restrizioni per negare ad altre persone le libertà principali. Questa regola non entra in conflitto con le libertà principali, anzi le protegge.

4.2 Licenze per il software libero

Una licenza di software libero è una licenza libera, un testo legale caratterizzato da un aspetto contrattuale o para-contrattuale, che si applica ad un software per garantirne la libertà d'utilizzo, di studio, di modifica e di condivisione, ovvero per renderlo software libero.
La nascita del concetto di licenza applicata ad un software per renderlo libero combacia in parte con la nascita di *GNU*, il primo sistema operativo completamente libero ideato da Richard Stallman nel 1983. Tutt'oggi il progetto GNU e la Free Software Foundation patrocinano attivamente il software distribuito sotto licenze libere e, in

generale, la libertà digitale degli utenti.

Di seguito vengono descritte alcune tra le licenze per il software libero utilizzate dalle librerie esterne necessarie allo sviluppo del software *Biztortion*.

4.2.1 GPL

La *GNU General Public License*[2] (comunemente indicata con l'acronimo GNU GPL o semplicemente GPL) è una licenza *fortemente copyleft* per software libero, originariamente stesa nel 1989 da Richard Stallman per patrocinare i programmi creati per il sistema operativo GNU.
La Free Software Foundation (FSF) detiene i diritti di copyright sul testo della GNU GPL, ma non detiene alcun diritto sul software da essa coperto. La GNU GPL non è liberamente modificabile: solo la copia e la distribuzione sono permesse. Per questo motivo solo la FSF può pubblicare nuove revisioni o versioni, l'ultima delle quali fu pubblicata il 29 giugno del 2007 sotto il nome di **GNU GPL v3**.

4.2.2 MIT

La *Licenza MIT*[3] è una licenza di software libero creata dal Massachusetts Institute of Technology (MIT). La licenza in questione è *senza copyleft* e risulta molto permissiva, in quanto a differenza delle licenze software copyleft, la licenza MIT consente anche il riutilizzo all'interno di software proprietario, a condizione che tutte le copie del software o delle sue parti sostanziali includano una copia dei termini della licenza MIT e anche un avviso di copyright. La licenza MIT È anche una licenza *GPL-compatibile*, cioè la GPL permette di combinare e ridistribuire tale software con altro che usa la licenza MIT.

4.2.3 BSD

Le *licenze BSD* sono una famiglia di licenze permissive, *senza copyleft*, per software libero. Il loro nome deriva dal fatto che la licenza BSD originale (detta anche licenza

[2] *Licenza GPL e GPLv3*. URL: https://www.gnu.org/licenses/quick-guide-gplv3.html.
[3] *Licenza MIT*. URL: https://choosealicense.com/licenses/mit/.

BSD con 4 clausole) fu usata originariamente per distribuire il sistema operativo Unix Berkeley Software Distribution (BSD), una revisione libera di UNIX sviluppata presso l'Università di Berkeley.
La versione originale è stata successivamente rivista e le sue discendenti sono più propriamente definite licenze BSD modificate. Due varianti della licenza, la Nuova Licenza BSD (o Licenza BSD Modificata)[4] e la Licenza semplificata BSD (o FreeBSD), sono state verificate come licenze di software libero *GPL-compatible* dalla Free Software Foundation.

4.3 Librerie utilizzate

Per la realizzazione del software *Biztortion* sono state utilizzate delle librerie sviluppate da terzi in modo da semplificare lo sviluppo di alcuni componenti. Di seguito vengono elencate e descritte tutte le librerie in questione, le quali utilizzano tutte delle licenze per software libero che sono *GPL-compatible* rendendo più semplice la scelta della licenza di rilascio e distribuzione del software sopraccitato:

* **JUCE**: il framework C++ in questione, necessario per la creazione dell'applicazione audio, utilizza termini di licenza a vari livelli, con termini diversi per ogni licenza disponibile: *JUCE Personal* (per sviluppatori o start-up con entrate inferiori al limite di entrate di 50K USD; gratuito), *JUCE Indie* (per le piccole imprese con limite di entrate inferiore a 500K USD; $ 40/mese), *JUCE Pro* (nessun limite di entrate; $ 130/mese) e *JUCE Educational* (nessun limite di entrate; gratuito per istituzioni educative in buona fede). Risulta infine possibile rilasciare un'applicazione sviluppata con JUCE sotto la **GNU GPLv.3**, rendendola software libero in modo che ne possa beneficiare tutta la comunità;

* **dRowAudio**: consiste in un modulo JUCE di terze parti progettato per lo sviluppo rapido di applicazioni audio; la libreria, sviluppata da *David Rowland*, contiene diverse classi per l'elaborazione audio e vari elementi utili per la

[4] *Licenza BSD modificata.* URL: https://www.gnu.org/licenses/license-list.html#ModifiedBSD.

creazione dell'interfaccia grafica. Il modulo in questione è distribuito secondo i termini della **Licenza MIT** per il software libero;

* **ff_meters**: consiste in un modulo JUCE di terze parti che offre un componente facile da usare per visualizzare una lettura di livello del segnale utilizzando un *juce::AudioBuffer* in ingresso. La libreria, sviluppata da *Daniel Walz* (Foleys Finest Audio Ltd.), deve essere utilizzata con il framework JUCE. Il modulo in questione è distribuito secondo i termini della **Licenza BSD a 3 clausole** (conosciuta anche come "Nuova" o "Modificata").

4.4 Software Biztortion

Il software *Biztortion*, poiché è stato sviluppato con le librerie citate nella sezione precedente che utilizzano licenze per il software libero *GPL-compatible*, viene distribuito secondo i termini stabiliti dalla licenza GNU GPLv.3, ovvero la più recente delle GPL. In sostanza la licenza in questione, oltre ad affermare le quattro libertà fondamentali del software libero, garantisce che tutte le versioni migliorate a partire da quella originale che saranno distribuite dovranno essere libere a loro volta.

Il software in questione viene rilasciato, insieme al presente documento per la discussione della tesi di laurea, alla versione *1.0* ed è possibile trovare tutti i file corrispondenti al prodotto software in questo repository remoto.

Capitolo 5

Conclusioni

5.1 Analisi del lavoro svolto

Il lavoro necessario alla realizzazione della tesi di laurea è risultato molto interessante e stimolante. La prima parte di esso, prevedendo uno studio sul fenomeno della distorsione del segnale audio, mi ha permesso di approfondire un concetto creativamente molto importante, in modo tale da poterlo applicare con criterio nell'implementazione degli algoritmi presenti nel software *Biztortion*.
La realizzazione del software in questione è risultata la parte più entusiasmante in quanto ha permesso di acquisire a livello pratico nuove abilità e competenze, oltre a consolidare quelle già possedute.
Il mio percorso di studi è risultato fondamentale per un corretto sviluppo del plugin audio; infatti il mio background accademico mi ha fornito le conoscenze di base legate all'*informatica musicale* (come per esempio l'audio digitale, il DSP e i filtri digitali) necessarie per un corretto approccio all'audio programming. Inoltre la mia carriera universitaria, legata alle scienze informatiche, ha reso possibile una buona analisi del problema e progettazione del software ancora prima di iniziare la fase di codifica, oltre che ad un rapido apprendimento autonomo delle tecnologie necessarie allo sviluppo del plugin stesso. In particolare ho potuto apprezzare una certa rapidità di scrittura del codice grazie all'esperienza con la programmazione ad oggetti e con il linguaggio di programmazione C++, maturata durante il corso universitario di *Programmazione ad Oggetti*. Un ulteriore merito va dato inoltre al corso universitario di *Ingegneria*

del Software, il quale mi ha dato le competenze necessarie ad una gestione efficiente ed efficace di un progetto software, oltre alla conoscenza degli strumenti legati alla stesura del presente documento.

5.2 Conoscenze acquisite

Lo sviluppo del software *Biztortion* mi ha permesso di interfacciarmi per la prima volta con il mondo della programmazione per l'audio, consentendomi di imparare in modo rapido e molto concreto alcune *tecnologie* ad asso legate come i formati VST/AU e il framework JUCE, il quale risulta molto popolare e utile anche per iniziare una carriera nel relativo ambito lavorativo.
Inoltre entrando nell'ottica dell'audio programming ho avuto la possibilità di *comprendere meglio le problematiche* legate a questa professione: in particolare ho verificato in prima persona l'impatto che le performance fornite sia dall'hardware che dal software hanno con la produzione del suono, rendendole fondamentali sia per l'elaborazione del segnale audio in tempo reale che per semplicemente evitare che le strumentazioni analogiche si possano danneggiare a causa di una produzione di click nell'audio in uscita.
Infine è stato molto interessante l'approfondimento effettuato sulle *licenze software* e sulle tecnologie open-source, risultate come uno strumento molto potente e di fondamentale importanza per tutti gli sviluppatori in modo tale che possano dare valore aggiunto all'intera community.

5.3 Valutazione personale

In conclusione ritengo che questo periodo, dedicato al lavoro per la tesi di laurea, mi abbia permesso di conoscere meglio e migliorare le mie capacità informatiche e di gestione delle tempistiche in modo autonomo.
Il lavoro svolto è stato molto interessante, stimolante ed appagante in quanto mi ha permesso un primo assaggio della parte tecnica del lavoro di un *audio programmer*, motivandomi ulteriormente nel cercare di cominciare una carriera in questo ambito lavorativo.

Glossario

API in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 41, 59

DAW Una *Workstation Audio Digitale*, a cui ci si riferisce spesso anche con il nome inglese di Digital Audio Workstation o con il suo acronimo DAW, è un sistema elettronico progettato per la registrazione, il montaggio e la riproduzione dell'audio digitale. 14, 17–19, 23, 40, 42, 46, 59

dBFS I *Decibel Full Scale* (dBFS o dB FS) sono un'unità di misura per i livelli di ampiezza nei sistemi digitali che hanno un livello di picco massimo definito. Il livello di 0dBFS è assegnato al massimo livello digitale possibile. Ad esempio, un segnale che raggiunge il 50% del livello massimo ha un livello di -6dBFS, che è 6dB al di sotto del massimo della scala. 7, 30, 59

DIM *La distorsione di intermodulazione* consiste nella modulazione di ampiezza di segnali contenenti due o più frequenze diverse, causata principalmente dalla non linearità di un sistema. L'intermodulazione tra i componenti di frequenza forma componenti aggiuntivi a frequenze che non sono solo alle frequenze armoniche (multipli interi) di entrambi, come la distorsione armonica, ma anche alle frequenze somma e differenza delle frequenze originali e alle somme e differenze di multipli di quelle frequenze. 6, 59

Dithering Il *dithering*, nella elaborazione numerica di segnali, è una forma di rumore con una opportuna distribuzione, che viene volontariamente aggiunto ai campioni con l'obiettivo di minimizzare la distorsione introdotta dal troncamento nel caso in cui si riquantizzino i campioni stessi. Il dithering viene usato abitualmente nell'elaborazione di segnali video e audio campionati e quantizzati. 15, 37

DSP Digital Signal Processing. 42, 43, 53

FFT In matematica, la *trasformata di Fourier veloce*, spesso abbreviata con FFT (dall'inglese Fast Fourier Transform), è un algoritmo ottimizzato per calcolare la trasformata discreta di Fourier (DFT) o la sua inversa. La FFT è utilizzata in una grande varietà di applicazioni, dall'elaborazione di segnali digitali alla soluzione di equazioni differenziali alle derivate parziali agli algoritmi per moltiplicare numeri interi di grandi dimensioni grazie al basso costo computazionale. 43, 59

IIR In teoria dei segnali, un sistema dinamico *Infinite Impulse Response* (in italiano risposta all'impulso infinita e spesso abbreviato in IIR) è un sistema dinamico causale la cui risposta impulsiva non è nulla al tendere all'infinito del tempo. I sistemi la cui risposta si annulla ad un tempo finito sono invece detti finite impulse response (FIR). Sebbene la definizione si adatti a sistemi tempo-continui, solitamente si ha a che fare con sistemi numerici, spesso i filtri digitali. 34, 35, 59

LFO L'*oscillatore a bassa frequenza* o LFO (sigla di Low Frequency Oscillator) è un generatore di forme d'onda a frequenza infrasonica, con funzione di modulatore di effetti, negli strumenti musicali elettronici. Per potersi considerare tale deve stare al di sotto di 20 Hz proprio perché l'orecchio umano può udire i suoni nell'intervallo dai 20 Hz ai 20 kHz e serve a modulare altri segnali. 13, 16, 59

Mockup Un *mockup*, o mock-up, è una realizzazione a scopo illustrativo o meramente espositivo di un oggetto o un sistema, senza le complete funzioni dell'originale; un mockup può rappresentare la totalità o solo una parte dell'originale di

riferimento (già esistente o in fase di progetto), essere in scala reale oppure variata. 3, 42

Open-source Con *open-source* (in italiano sorgente aperto), in informatica, si indica un tipo di software o il suo modello di sviluppo o distribuzione. Un software open source è reso tale per mezzo di una licenza attraverso cui i detentori dei diritti favoriscono la modifica, lo studio, l'utilizzo e la redistribuzione del codice sorgente. 1, 18, 40, 54

Plugin Un *plugin* in campo informatico corrisponde ad un programma non autonomo che interagisce con un altro programma per ampliarne o estenderne le funzionalità originarie (ad es. un plugin per un software di grafica permette l'utilizzo di nuove funzioni non presenti nel software principale): possono essere utilizzati non solo su software, ma anche su qualunque cosa che possa essere visitata da chiunque, quindi pubblica (ad es. i videogiochi online). 13–15, 17, 19, 24, 39–42, 44, 53

Polimorfismo In informatica, il termine *polimorfismo* viene usato in senso generico per riferirsi a espressioni che possono rappresentare valori di diversi tipi (dette espressioni polimorfiche). In un linguaggio non tipizzato, tutte le espressioni sono intrinsecamente polimorfiche. Il termine nel contesto della programmazione orientata agli oggetti si riferisce al fatto che un'espressione il cui tipo sia descritto da una classe A può assumere valori di un qualunque tipo descritto da una classe B sottoclasse di A (polimorfismo per inclusione). 43

Power Chord Un *power chord* (in inglese, letteralmente, "accordo potente"), anche noto come accordo di quinta o quinta vuota, è un bicordo i cui suoni vengono di solito eseguiti simultaneamente. 6

Pre-enfasi La *Pre-Enfasi* di un segnale audio consiste in un processo in cui è previsto l'utilizzo di due shelf filter. Entrambi i filtri possono essere o high shelf o low shelf e sono posti uno prima e uno dopo un ulteriore blocco di elaborazione audio (spesso consiste in un effetto di distorsione). L'operazione di pre-enfasi prevede l'applicazione di un'attenuazione o enfatizzazione della frequenza del

primo filtro, in modo tale da effettuare l'operazione inversa nel secondo filtro. 46

Projucer *Projucer* è uno strumento per la creazione e la gestione di progetti JUCE. Una volta specificati i file e le impostazioni per un progetto JUCE, Projucer genera automaticamente una raccolta di file di progetto di terze parti per consentire la compilazione nativa del progetto su ciascuna piattaforma di destinazione. Attualmente può generare progetti Xcode, Visual Studio, Linux Makefile, CodeBlocks e build Android Ant. Oltre a fornire un modo per gestire i file e le impostazioni di un progetto, ha anche un editor di codice, un editor GUI integrato, procedure guidate per la creazione di nuovi progetti/file e un motore di codifica live utile per la progettazione dell'interfaccia utente. 42

Sidechain Il *sidechain* è una tecnica di compressione che, applicato a una traccia, attiva la compressione di un processore di dinamica a partire da un altro segnale audio esterno a tale traccia. Di norma i compressori vengono cablati in insert e il loro funzionamento dipende dal materiale audio che passa per quella data traccia. Se a un compressore posto in insert viene attivato il controllo sidechain, questo smetterà di funzionare, fin quando ad esso verrà indicato quale segnale dovrà fare da trigger (da azionatore). 13

UML in ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 3, 18, 59

Acronimi

API Application Program Interface. 55

DAW Digital Audio Workstation. 55

dBFS Decibel Full Scale. 55

DIM Distorsione di intermodulazione. 55

FFT Fast Fourier Transform. 56

IIR Infinite Impulse Response. 56

LFO Low Frequency Oscillator. 56

UML Unified Modeling Language. 58

Bibliografia

Riferimenti bibliografici

G. Davis, R. Jones. *The Sound Reinforcement Handbook*. 2ª ed. Yamaha, 1989. ISBN: 9780881889000 (cit. a p. 6).

Glenn D. White, Gary J. Louie. *The Audio Dictionary*. 3ª ed. University of Washington Press, 2005. ISBN: 0-295-98498-8 (cit. a p. 5).

Kleiner, M. *Acoustics and Audio Technology*. 3ª ed. J. Ross Publishing, 2011. ISBN: 9781604270525, 1604270527 (cit. a p. 6).

P. Burk L. Polansky, D. Repetto. *Music and Computers*. Key College Puslishing, 2005. ISBN: 1-930190-95-6 (cit. a p. 10).

Puckette, Miller. *The Theory and Technique of Electronic Music*. World Scientific Publishing Co. Pte. Ltd., 2006 (cit. a p. 8).

Siti web consultati

Befaco Slew Limiter. URL: https://library.vcvrack.com/Befaco/SlewLimiter (cit. a p. 11).

C++ features. URL: https://www.cplusplus.com/info/description/ (cit. a p. 38).

Casi D'uso. URL: https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20Use%20Case_4x4.pdf (cit. a p. 18).

Definizione di software libero. URL: https://www.gnu.org/philosophy/free-sw.html (cit. a p. 47).

Fabfilter Saturn 2. URL: https://www.fabfilter.com/products/saturn-2-multiband-distortion-saturation-plug-in (cit. a p. 15).

Izotope Trash 2. URL: https://www.izotope.com/en/products/trash.html (cit. a p. 13).

JUCE website. URL: https://juce.com/ (cit. a p. 40).

Kilohearts Toolbox. URL: https://kilohearts.com/products/kilohearts_toolbox (cit. a p. 14).

Licenza BSD modificata. URL: https://www.gnu.org/licenses/license-list.html#ModifiedBSD (cit. a p. 50).

Licenza GPL e GPLv3. URL: https://www.gnu.org/licenses/quick-guide-gplv3.html (cit. a p. 49).

Licenza MIT. URL: https://choosealicense.com/licenses/mit/ (cit. a p. 49).

Steinberg VST website. URL: https://www.steinberg.net/technology/ (cit. a p. 40).