ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«ΨΗΦΙΑΚΗ ΑΝΤΗΧΗΣΗ: ΑΠΟ ΤΗΝ ΘΕΩΡΙΑ ΣΤΗΝ ΥΛΟΠΟΙΗΣΗ»

Του φοιτητή
Βασίλη Κοντσέ
Αρ. Μητρώου: 154608

Επιβλέπων
Κωτσάκης Ρήγας
Επίκουρος Καθηγητής

ΜΑΙΟΣ 2024

Ψηφιακή Αντήχηση: Από την θεωρία στην υλοποίηση
23129
Βασίλης Κοντσές
Κωτσάκης Ρήγας
8 Μαρτίου 2023
20 Μαΐου 2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Η επεξεργασία του ήχου και η παραγωγή μουσικής είναι τομείς με συνεχώς αυξανόμενη ζήτηση και οι εφαρμογές αντήχησης αποτελούν βασικό εργαλείο σε αυτές τις διαδικασίες. Η παρόν πτυχιακή εργασία προσέφερε μια ιδανική ευκαιρία να συνδυαστεί η θεωρητική γνώση, με την πρακτική εφαρμογή. Διερευνώνται, τόσο οι φυσικές και μαθηματικές αρχές πίσω από τον ήχο και την αντήχηση, ενώ παράλληλα αναπτύσσεται ένα λογισμικό που μπορεί να χρησιμοποιηθεί από ερασιτέχνες μουσικούς μέχρι και επαγγελματίες.

Επιπλέον, η ανάπτυξη μιας εφαρμογής ψηφιακής αντήχησης απαιτεί γνώσεις προγραμματισμού, ψηφιακής επεξεργασίας σήματος (DSP), και σχεδιασμού διεπαφής του χρήστη (GUI), προσφέροντας μια ολιστική προσέγγιση σε τεχνικές δεξιότητες. Αυτό μου επίτρεψε να αποκτήσω εμπειρία σε διάφορους τομείς της μηχανικής σε ρεαλιστικό επίπεδο ώστε να μπορώ να ανοίξω πόρτες για μελλοντικές επαγγελματικές ευκαιρίες με κατεύθυνση προς τις εταιρείες λογισμικού ή στον χώρο της μουσικής βιομηχανίας.

«Ψηφιακή Αντήχηση: Από την θεωρία στην υλοποίηση»

«Βασίλης Κοντσές»

Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην αναλυτική μελέτη και υλοποίηση ενός plugin αντήχησης, καλύπτοντας θεωρητικά και πρακτικά ζητήματα της ψηφιακής αντήχησης. Αρχικά, διερευνώνται οι βασικές αρχές επεξεργασίας του ψηφιακού ηχητικού σήματος, αναλύοντας από την δειγματοληψία του και την ανακατασκευή του, μέχρι, τους αλγόριθμους επεξεργασίας του. Ακολούθως, εξετάζεται η θεωρία της ψηφιακής αντήχησης, περιγράφοντας τα φυσικά και μαθηματικά μοντέλα που χρησιμοποιούνται για την αναπαραγωγή της. Αναλύονται οι αισθητικοί παράμετροι που διαφοροποιούν το θεμιτό αποτέλεσμα και πρέπει να λάβουμε υπόψιν για την κατανόηση της εφαρμογής. Η εργασία προχωρά με την ανάλυση της δομής των audio plugins, εξηγώντας τα βασικά συστατικά τη λειτουργία τους. Τέλος, παρουσιάζεται η υλοποίηση ενός reverb plugin χρησιμοποιώντας το Juce Framework για την δημιουργία της διεπαφής του χρήστη, παρέχοντας λεπτομέρειες για την ανάπτυξη του λογισμικού και επεξήγηση του κώδικα που χρησιμοποιήθηκε.

Digital Reverb: From theory to implementation

Vassilis Kontses

Abstract

This project focuses on the detailed study and implementation of a reverb plugin, covering theoretical and practical issues of digital reverberation. Initially, the basic principles of digital audio signal processing are explored, analyzing from its sampling and reconstruction, to, its processing algorithms. Next, the theory of digital reverberation is examined, describing the physical and mathematical models used to reproduce it. The aesthetic parameters that differentiate the legitimate effect and must be considered to understand the application are analyzed. The study proceeds with an analysis of the structure of audio plugins, explaining the basic components of their function. Finally, the implementation of a reverb plugin using the Juce Framework to create the user interface is presented, providing details of the software development and an explanation of the code used.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

ADC	Analog to Digital Converter
API	Application Programming Interface
AU	Audio Unit
BPM	Beats per minute
CD	Compact Disc
DAC	Digital to Analog Converter
DC	Direct Current
DB	Decibel
DSP	Digital Signal Processing
DXi	Direct X instrument
FFT	Fast Fourier Transform
GMPI	Generalized Musical Plugin Interface
GUID	Globally Unique ID
HPF	High-Pass Filter
Hz	Hertz
I/O	Input/Output
IEEE	Institute of Electrical and Electronics Engineers
IRQ	Interrupt Request Line
ISR	Interrupt Service Routine
LADSPA	Linux Audio Developer's Simple Plugin API
LPF	Low-Pass Filter
MAS	MOTU Audio System
Max	Maximum
MIDI	Musical Instrument Digital Interface
Min	Minimum
Ms	Millisecond
RTAS	Real-Time Audio Suite
UI	User Interface
VST	Virtual Studio Technology
WAV	Waveform Audio File Format

Κεφάλαιο 1ο: Αρχές επεξεργασίας ψηφιακού ηχητικού σήματος

1.1 Εισαγωγή

Οι πρώτες προσιτές συσκευές αναπαραγωγής ψηφιακού ήχου άρχισαν να εμφανίζονται στα μέσα της δεκαετίας του 1980. Η επεξεργασία του ψηφιακού σήματος (DSP) από την μαθηματική σκοπιά, προϋπήρχε από τη δεκαετία του 1960 και τα εμπορικά μέσα ψηφιακής ηχογράφησης εμφανίστηκαν για πρώτη φορά στις αρχές της δεκαετίας του 1970, αλλά η τεχνολογία δεν έγινε διαθέσιμη για ευρεία διανομή μέχρι περίπου 15 χρόνια αργότερα, όταν εμφανίστηκε ο συμπαγής δίσκος (Compact Disc ή η ευρέος πιο γνωστή συντομογραφία CD) όπου εγκαινίασε την εποχή του ψηφιακού ήχου. Η ψηφιακή δειγματοληψία αναφέρεται στην απόκτηση σημείων δεδομένων από ένα συνεχές αναλογικό σήμα. Τα σημεία δεδομένων δειγματοληψίας υπολογίζονται σε τακτά χρονικά διαστήματα. Διαστήματα γνωστά ως περίοδος δειγματοληψίας ή διάστημα δειγματοληψίας. Το αντίστροφο της περιόδου δειγματοληψίας είναι η συχνότητα δειγματοληψίας. Ένας συμπαγής δίσκος χρησιμοποιεί συχνότητα δειγματοληψίας 44.100 Hz, παράγοντας 44.100 διακριτά δείγματα ανά κανάλι κάθε δευτερόλεπτο, με διάστημα δειγματοληψίας περίπου 22,7 μικροδευτερολέπτων (μS). Ενώ η ψηφιακή δειγματοληψία εφαρμόζεται σε πολλά διαφορετικά συστήματα, η παρούσα μελέτη επικεντρώνεται σε μία μόνο από αυτές τις εφαρμογές: τον ήχο. [1]

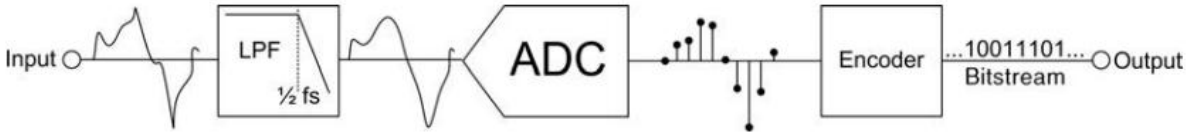
Κατά τη διάρκεια αυτής της πτυχιακής εργασίας, θα μάθετε τόσο τη θεωρία όσο και τις εφαρμογές του DSP στον ήχο. Αυτό επιτυγχάνεται καλύτερα με τον πειραματισμό των δικών σας αλγορίθμους DSP ταυτόχρονα με την εκμάθηση της θεωρίας, χρησιμοποιώντας το παρεχόμενο του JUCE framework. Ο στόχος είναι να κατανοήσουμε πώς οι αλγόριθμοι DSP μεταφράζονται σε C++ κώδικα. Τα παραγόμενα plug-ins μπορούν να χρησιμοποιηθούν για την επεξεργασία ήχου σε πραγματικό χρόνο, σαν το αντίστοιχο που υλοποιήθηκε, όπως θα δούμε παρακάτω. Επειδή τα plug-ins ξεκίνησαν να σχεδιάζονται σαν ψηφιακές εκδοχές λογισμικού σε σχέδια ήδη υπάρχον υλικού εξοπλισμού που χρησιμοποιείται σε επαγγελματικά στούντιο, αξίζει να εξετάσουμε τον τρόπο λειτουργίας αυτών των υλικών συστημάτων, τις τυπικές μορφές κωδικοποίησης ήχου και τα δομικά στοιχεία των αλγορίθμων για να κατανοήσουμε καλύτερα την λειτουργία των plug-ins.

1.2 Δειγματοληψία

Το θεώρημα δειγματοληψίας δηλώνει ότι ένα συνεχές αναλογικό σήμα μπορεί να δειγματοληπτηθεί σε διακριτά σημεία δεδομένων και στη συνέχεια να ανακατασκευαστεί στο αρχικό αναλογικό σήμα χωρίς καμία απώλεια πληροφορίας - συμπεριλαμβανομένων των διακυμάνσεων μεταξύ των δειγμάτων - εάν και μόνο εάν το σήμα εισόδου έχει περιοριστεί σε ζώνες, έτσι ώστε να μην περιέχει συχνότητες υψηλότερες από το μισό του ρυθμού δειγματοληψίας, επίσης γνωστό ως συχνότητα Nyquist ή ρυθμός Nyquist. Ο περιορισμένη ζωνών σημαίνει χαμηλοπερατό φίλτρο (Low-Pass Filter ή LPF). Ο περιορισμός της ζώνης εισόδου σήματος πριν από τη δειγματοληψία είναι επίσης γνωστός ως τήρηση των κριτηρίων Nyquist.

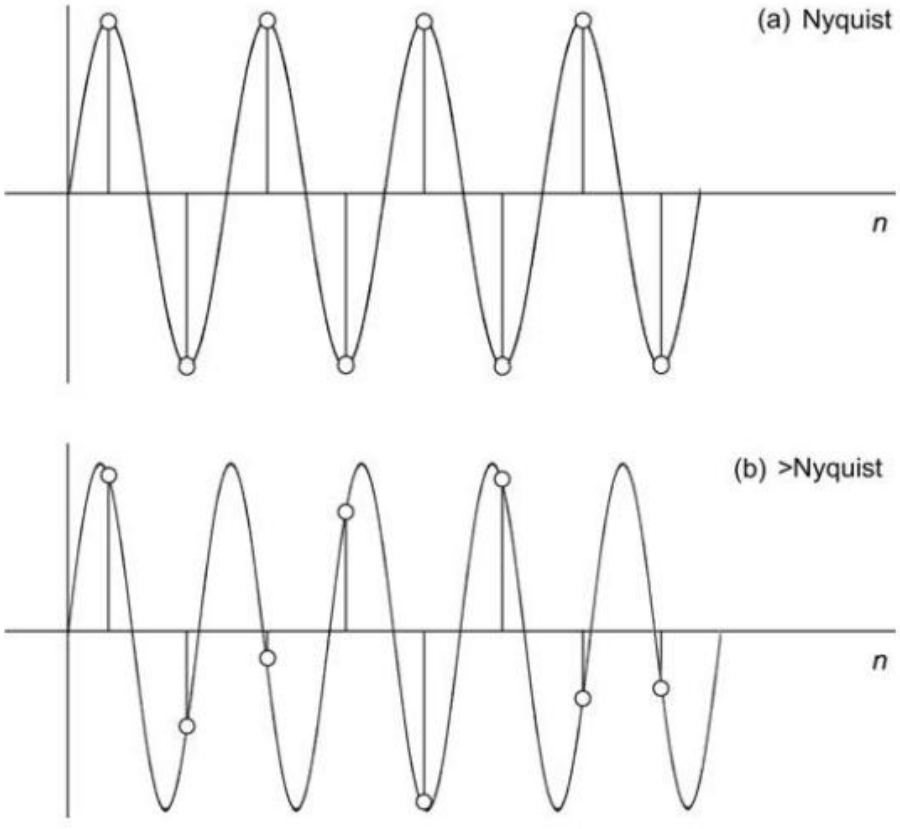
Το εισερχόμενο αναλογικό σήμα ήχου δειγματοληπτείται με έναν μετατροπέα αναλογικού σε ψηφιακό σήμα (ADC ή A/D). Ο A/D μετατροπέας πρέπει να μετατρέπει με ακρίβεια ένα εισερχόμενο σήμα, παράγοντας έναν έγκυρο ψηφιακά κωδικοποιημένο αριθμό που αντιπροσωπεύει την αναλογική τάση εντός του διαστήματος δειγματοληψίας. Αυτό σημαίνει ότι για CD ήχου, ένας μετατροπέας πρέπει να

παράγει μια έξοδο κάθε 22,7 mS. [2] Στο σχήμα 1.1 παρουσιάζεται το διάγραμμα του συστήματος μετατροπής εισόδου με LFP, A/D και ο κωδικοποιητής (Encoder).



Σχήμα 1.1: Το σύστημα μετατροπής εισόδου καταλήγει τελικά σε αριθμητική κωδικοποίηση του σήματος εισόδου περιορισμένης ζώνης.

Η παραβίαση των κριτηρίων Nyquist θα δημιουργήσει ακουστικά σφάλματα στο σήμα με τη μορφή ενός λανθασμένα κωδικοποιημένου σήματος. Συχνότητες υψηλότερες από το Nyquist θα αναδιπλωθούν πίσω στο φάσμα. Αυτό το φαινόμενο ονομάζεται aliasing επειδή οι συχνότητες υψηλότερες από το Nyquist κωδικοποιούνται "μεταμφιεσμένες" ή ως "alias" της πραγματικής συχνότητας. Αυτό εξηγείται ευκολότερα με την εικόνα ενός αλλοιωμένου σήματος [2], που παρουσιάζεται στο Σχήμα 1.2.

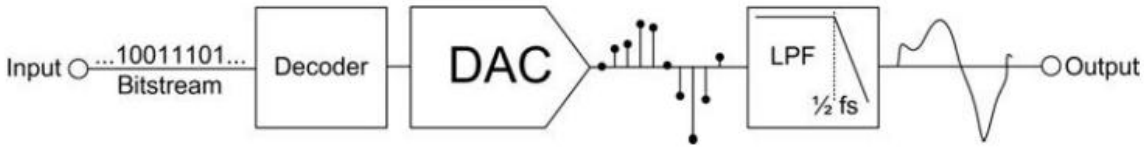


Σχήμα 1.2: (a) Η συχνότητα Nyquist είναι η υψηλότερη συχνότητα που μπορεί να κωδικοποιηθεί με δύο δείγματα ανά περίοδο. (b) Η αύξηση της συχνότητας πάνω από το Nyquist αλλά διατήρηση της στο ίδιο διάστημα δειγματοληψίας έχει ως αποτέλεσμα ένα προφανές σφάλμα κωδικοποίησης - το αποτέλεσμα είναι το αλλοιωμένο σήμα.

Μόλις δημιουργηθεί το αλλοιωμένο σήμα, δεν μπορεί ποτέ να αφαιρεθεί και θα παραμείνει ως μόνιμο σφάλμα στο σήμα. Το χαμηλοπερατό φίλτρο (Low-pass filter) που περιορίζει τη ζώνη του σήματος στην είσοδο ονομάζεται anti-aliasing φίλτρο. Ένα καλό παράδειγμα για να το κατανοήσουμε αλλά σε άλλη μορφή αλλοίωσης εμφανίζεται στις ταινίες. Μια αναλογική κινηματογραφική κάμερα λαμβάνει 30 εικόνες (καρέ) ανά δευτερόλεπτο. Ωστόσο, συχνά πρέπει να απεικονίζει αντικείμενα που περιστρέφονται με πολύ υψηλότερους ρυθμούς από 30 ανά δευτερόλεπτο, όπως πτερύγια ελικοπτέρου ή ρόδες αυτοκινήτου. Το αποτέλεσμα είναι οπτικά συγκεχυμένο - τα πτερύγια του ελικοπτέρου ή οι τροχοί του αυτοκινήτου φαίνεται να επιβραδύνονται και να σταματούν, στη συνέχεια να αντιστρέφουν τις κατευθύνσεις και να επιταχύνουν, στη συνέχεια να επιβραδύνονται και να σταματούν, να αντιστρέφουν κ.ο.κ. Αυτό είναι το οπτικό αποτέλεσμα της υψηλής συχνότητας περιστροφής που επιστρέφει στην ταινία ως λανθασμένη κωδικοποίηση του πραγματικού γεγονότος.

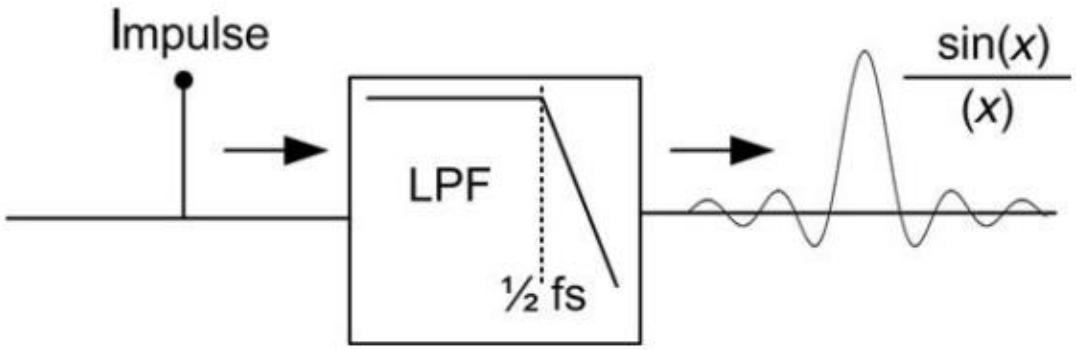
1.3 Ανακατασκευή του σήματος

Ο μετατροπέας ψηφιακού σε αναλογικό (DAC ή D/A) πρώτα αποκωδικοποιεί τη ροή των bits στη συνέχεια παίρνει τα δειγματοληπτούμενα σημεία δεδομένων ή παλμούς και τα μετατρέπει σε αναλογικές εκδοχές αυτών των παλμών. στη συνέχεια παίρνει τα δειγματοληπτούμενα σημεία δεδομένων ή παλμούς και τα μετατρέπει σε αναλογικές εκδοχές αυτών των παλμών.



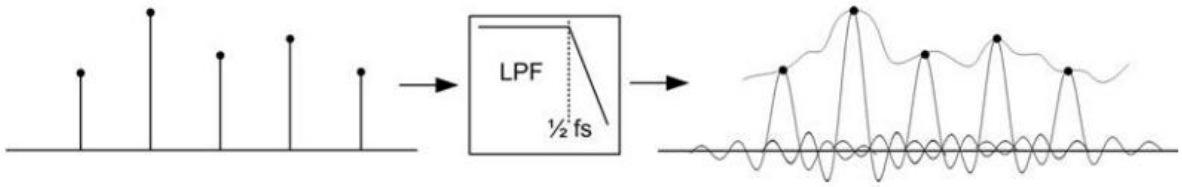
Σχήμα 1.3: Η ψηφιακή ροή των bits αποκωδικοποιείται και μετατρέπεται σε αναλογική έξοδο με τη χρήση ενός χαμηλοπερατού φίλτρου για την ανακατασκευή του αναλογικού σήματος..

Το φίλτρο εξόδου ονομάζεται φίλτρο ανακατασκευής και είναι υπεύθυνο για την αναδημιουργία της φαινομενικά χαμένης πληροφορίας που δεν απέκτησε η αρχική διαδικασία δειγματοληψίας - όλες τις διακυμάνσεις μεταξύ των δειγμάτων. Ο λόγος για τον οποίο λειτουργεί είναι ότι το χαμηλοπερατό φίλτρο μετατρέπει τους παλμούς σε αλλοιωμένες, κουνημένες εκδοχές. Οι εξομαλυμένες εκδόσεις έχουν τη μορφή f (x) = sin(x)/x η οποία ονομάζεται επίσης συνάρτηση sinc() και μοιάζει κάπως με το προφίλ ενός καπέλου σομπρέρο, όπως φαίνεται στο Σχήμα 1.4 .



Σχήμα 1.4: Το ιδανικό φίλτρο ανακατασκευής δημιουργεί μια κηλιδωμένη έξοδο με αποσβενόμενη ταλαντωτική μορφή. Το πλάτος του σχήματος sin(x)/x είναι ανάλογο του πλάτος του παλμού εισόδου.

Όταν μια σειρά παλμών φιλτράρεται, η προκύπτουσα σειρά παλμών sin(x)/x επικαλύπτει η μία την άλλη και οι αποκρίσεις τους αθροίζονται γραμμικά. Η πρόσθεση όλων των μικρότερων καμπυλών και των αποσβεννύμενων ταλαντώσεων ανακατασκευάζει τις καμπύλες και τις ταλαντώσεις μεταξύ των δειγμάτων, βλέπε Σχήμα 1.5.

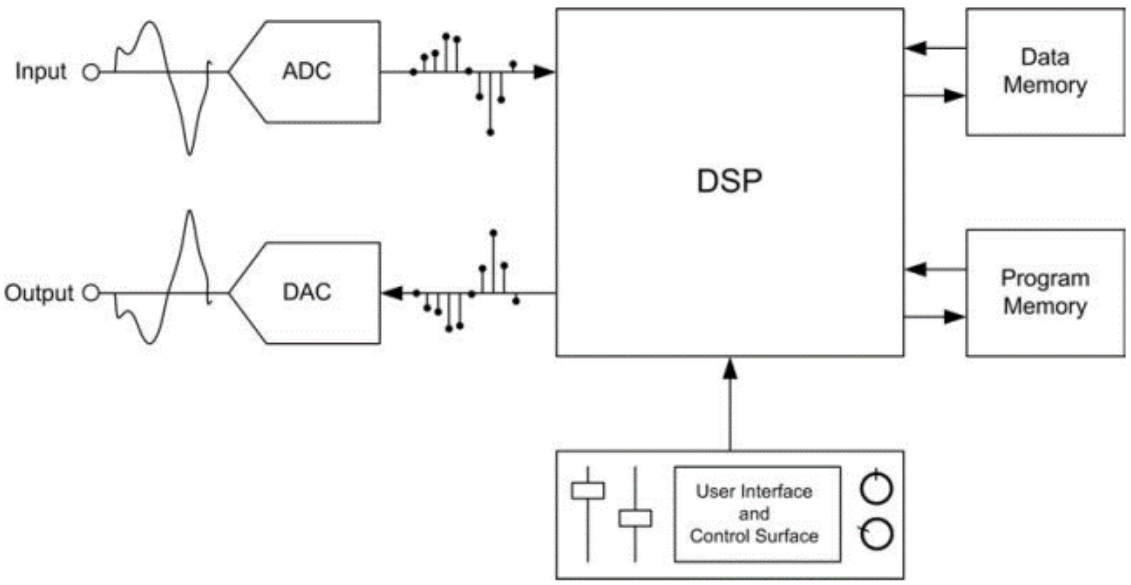


Σχήμα 1.5: Οι έξοδοι sin(x)/x του LPF αθροίζονται για την ανακατασκευή της αρχικής κυματομορφής εισόδου. Η πληροφορία μεταξύ των δειγμάτων έχει ανακατασκευαστεί.

1.4 Συστήματα επεξεργασίας σήματος

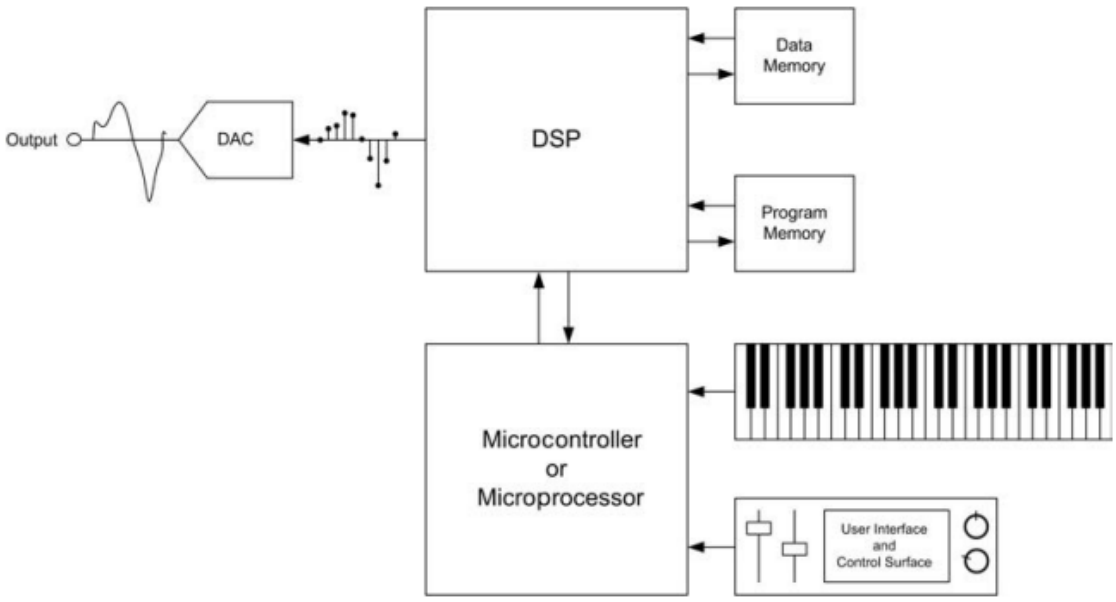
Τα συστήματα επεξεργασίας σήματος συνδυάζουν συσκευές συλλογής δεδομένων με μικροεπεξεργαστές για την εκτέλεση μαθηματικών αλγορίθμων στα δεδομένα του ήχου. Αυτοί οι αλγόριθμοι αποτελούν το επίκεντρο αυτής της εργασίας. Τα σημερινά plug-ins είναι απόγονοι των αυτόνομων επεξεργαστών υλικού που βασίζονται σε τσιπ DSP. Ένα τσιπ DSP είναι ένας εξαιρετικά εξειδικευμένος επεξεργαστής που έχει σχεδιαστεί κυρίως για την εκτέλεση DSP αλγορίθμων, θα λειτουργούσε ελάχιστα ως ένας κοινός επεξεργαστής (CPU) για έναν προσωπικό υπολογιστή, επειδή διαθέτει μόνο ένα μικρό αλλά εξαιρετικά εξειδικευμένο σύνολο εντολών. Οι συσκευές διαθέτουν έναν κεντρικό επεξεργαστή σχεδιασμένο να πολλαπλασιάζει και να συσσωρεύει δεδομένα, επειδή η λειτουργία αυτή είναι θεμελιώδης για τους αλγορίθμους DSP. Επειδή αυτή η διαδικασία επαναλαμβάνεται ξανά και ξανά, οι σύγχρονοι DSP χρησιμοποιούν pipelining για να φέρνουν τα επόμενα δεδομένα ενώ ταυτόχρονα επεξεργάζονται τα τρέχοντα δεδομένα. Αυτή η τεχνική βελτιώνει σημαντικά την αποδοτικότητα του συστήματος. [3] Ένα τυπικό σύστημα επεξεργασίας σήματος αποτελείται από τα ακόλουθα στοιχεία (Σχήμα 1.6):

- Συσκευές συλλογής δεδομένων (A/D και D/A)
- Ένα τσιπ DSP
- Μνήμη για την αποθήκευση του προγράμματος αλγορίθμου (μνήμη προγράμματος)
- Μνήμη για την αποθήκευση των δεδομένων του αλγορίθμου (μνήμη δεδομένων)
- Διεπαφή χρήστη (UI) με κουμπιά, κουμπιά, διακόπτες και οθόνες γραφικών



Σχήμα 1.6: Ένα απλό σύστημα επεξεργασίας σήματος. Ο αλγόριθμος σε αυτή την περίπτωση είναι αντιστρόφως ανάλογος της φάσης του σήματος- η έξοδος είναι ανάποδη.

Οι συσκευές μουσικής σύνθεσης αποτελούνται από μια ενσωματωμένη έκδοση μιας διάταξης CPU και DSP. Ο μικροελεγκτής ή ο μικροεπεξεργαστής διαχειρίζεται την διεπαφή του χρήστη (UI) και το πληκτρολόγιο, ενώ το DSP εκτελεί τους αλγορίθμους ηχητικής σύνθεσης και των εφέ. Οι πρακτικές υλοποιήσεις περιλαμβάνουν πλέον πολλαπλούς DSP, μερικοί από τους οποίους είναι ειδικά σχεδιασμένοι ειδικά για τη σύνθεση ήχου. Ένα παράδειγμα διαγράμματος παρουσιάζεται παρακάτω στο Σχήμα 1.7.



Σχήμα 1.7: Διάγραμμα ενός συνθεσάιζερ.

1.5 Συγχρονισμός και διακοπές

Υπάρχουν δύο θεμελιώδεις τρόποι λειτουργίας όταν πρόκειται για εισερχόμενα και εξερχόμενα δεδομένα ήχου: ο συγχρονισμός και ο ασυγχρονισμός. Στη συγχρονισμένη λειτουργία, όλα τα δεδομένων εισόδου και εξόδου συγχρονίζονται στο ίδιο ρολόι με το DSP. Πρόκειται συνήθως για απλά συστήματα με ελάχιστες εισόδους/εξόδους (I/O) και περιφερειακά. Τα πιο σύνθετα συστήματα περιλαμβάνουν συνήθως ασύγχρονη λειτουργία, όπου τα δεδομένα ήχου δεν συγχρονίζονται με το DSP. Επιπλέον, ο ίδιος ο ήχος μπορεί να μην είναι συγχρονισμένος, δηλαδή οι ροή των bits της εισόδου και της εξόδου μπορεί να μην λειτουργούν με το ίδιο ρολόι. Ένα αμιγώς σύγχρονο σύστημα θα είναι πιο ασφαλές, αλλά λιγότερο εύχρηστο.

Ένα ασύγχρονο σύστημα θα βασίζεται σχεδόν πάντα σε διακοπές. Σε μια σχεδίαση βασισμένη σε διακοπές, ο επεξεργαστής εισέρχεται σε έναν βρόχο αναμονής μέχρι να ενεργοποιηθεί μια διακοπή του επεξεργαστή. Ο επεξεργαστής διακοπής είναι ακριβώς σαν κουδούνι. Όταν μια άλλη συσκευή, όπως ο A/D, έχει δεδομένα έτοιμα να παραδώσει, τοποθετεί τα δεδομένα σε έναν προκαθορισμένο buffer, και στη συνέχεια χτυπά το κουδούνι ενεργοποιώντας το ένα pin της διακοπής. Ο επεξεργαστής εξυπηρετεί τη διακοπή με μια συνάρτηση που παραλαμβάνει τα δεδομένα, και στη συνέχεια συνεχίζει με τον κώδικα επεξεργασίας τους. Η συνάρτηση είναι γνωστή ως ISR (Interrupt Service Routine) ή χειριστής διακοπής (interrupt handler). Σε αντίθεση με την συγχρονισμένη λειτουργία που είδαμε παραπάνω, το σύστημα που βασίζεται στη διακοπή, είναι το πιο αποδοτικό στην χρήση του χρόνου του επεξεργαστή, επειδή ο επεξεργαστής δεν μπορεί να προβλέψει τον ακριβή κύκλο του ρολογιού κατά τον οποίο τα δεδομένα θα είναι έτοιμα στην είσοδο.

Μια άλλη πηγή διακοπών είναι το UI. Κάθε φορά που ο χρήστης αλλάζει ένα στοιχείο του χειριστηρίου, πατάει ένα κουμπί ή γυρίζει ένα κουμπί, οι ενημερωμένες πληροφορίες ελέγχου του UI πρέπει να αποστέλλονται στον DSP, ώστε να μπορεί να τροποποιήσει την επεξεργασία του για να προσαρμόσει τις νέες ρυθμίσεις. Αυτό επιτυγχάνεται επίσης μέσω διακοπών και χειριστές διακοπών (interrupt handler). Η διακοπή μερικές φορές χαρακτηρίζεται ως INT, INTR ή IRQ (Interrupt Request Line) στα σχήματα των διαγραμμάτων.

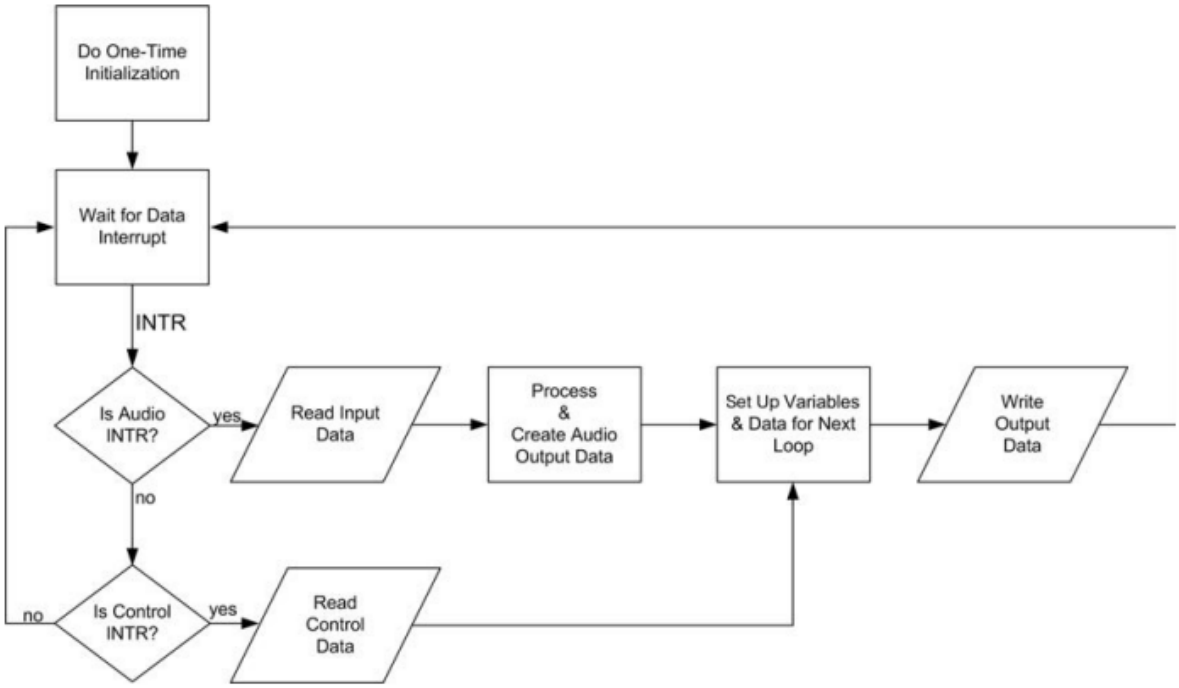
1.6 Ροή επεξεργασίας σήματος

Είτε η επεξεργασία λαμβάνει χώρα σε ένα τσιπ DSP είτε σε ένα plug-in, η συνολική ροή επεξεργασίας, γνωστή και ως βρόχος επεξεργασίας σήματος, παραμένει βασικά η ίδια. Για ένα τσιπ DSP, οι διεργασίες στο βρόχο θα κωδικοποιηθούν στο σύνολο των εντολών της μνήμης προγράμματος, συχνά με σχεδόν αμιγώς γραμμικό τρόπο για τη βελτιστοποίηση των κύκλων του επεξεργαστή. [5] Για ένα plug-in, κάθε μπλοκ επεξεργασίας παίρνει λίγο πολύ τη μορφή μιας συνάρτησης στον κώδικα, επιτρέποντας τη μέγιστη δυνατή ευελιξία.

Για παράδειγμα θα μελετήσουμε το Σχήμα 1.8 στο οποίο ο βρόχος επεξεργασίας αποτελείται από:

- Μια εφάπαξ λειτουργία αρχικοποίησης για τη διαμόρφωση της αρχικής κατάστασης του επεξεργαστή και την προετοιμασία για την άφιξη των διακοπών των δεδομένων.
- Ένας διαρκής βρόχος αναμονής, ο οποίος δεν κάνει τίποτα άλλο από το να περιμένει την εμφάνιση μιας διακοπής.
- Ένας χειριστής διακοπής, ο οποίος αποκωδικοποιεί τη διακοπή και αποφασίζει πώς να επεξεργαστεί -ή να αγνοήσει- το κουδούνι
- Λειτουργίες ανάγνωσης και εγγραφής δεδομένων τόσο για πληροφορίες ελέγχου όσο και για πληροφορίες δεδομένων
- Μια συνάρτηση επεξεργασίας για το χειρισμό και τη δημιουργία της εξόδου του ήχου.

- Μια συνάρτηση για τη ρύθμιση των μεταβλητών για την επόμενη φορά που θα γίνει ο βρόχος, μεταβάλλοντας τις μεταβλητές εάν το δικαιολογούν οι αλλαγές στον έλεγχο του UI.



Σχήμα 1.8: Το διάγραμμα ροής για ένα σύστημα επεξεργασίας ηχητικού σήματος.

Για να ισχύει το θεώρημα της δειγματοληψίας, τα δεδομένα ήχου πρέπει να φθάνουν και να φεύγουν σε αυστηρά χρονικά διαστήματα, αν και μπορεί να είναι ασύγχρονα με το εσωτερικό ρολόι του DSP. Αυτό σημαίνει ότι όταν ο DSP λαμβάνει ένα ηχητικό INTR πρέπει να κάνει όλη την επεξεργασία ήχου και να χειριστεί τυχόν διακοπές από το UI πριν φτάσει το επόμενο ηχητικό INTR σε ένα διάστημα δειγματοληψίας. Το σχήμα χειρισμού των διακοπών έχει ιεραρχηθεί έτσι ώστε η διακοπή δεδομένων ήχου να είναι η πιο σημαντική. Έτσι, κατά την εξυπηρέτηση της διακοπής δεδομένων ήχου, ο DSP μπορεί να ρυθμιστεί ώστε να αγνοεί όλες τις άλλες διακοπές (εκτός από τη διακοπή επαναφοράς) μέχρι να ολοκληρωθεί η επεξεργασία των δεδομένων ήχου.

1.7 Αριθμητική αναπαράσταση των ηχητικών δεδομένων

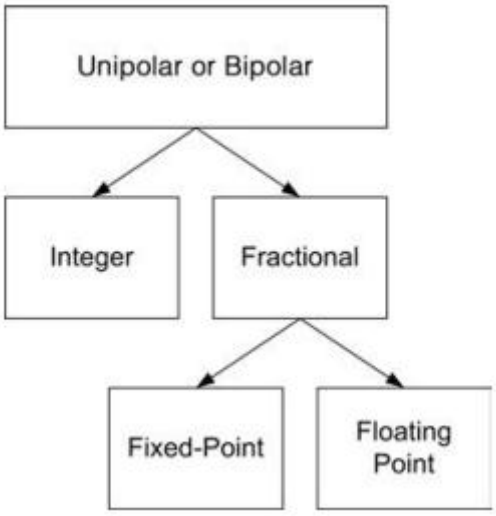
Τα δεδομένα ήχου μπορούν να κωδικοποιηθούν αριθμητικά με διάφορους τρόπους. Η βασική θεωρία του ψηφιακού ήχου αποκαλύπτει ότι ο αριθμός των επιπέδων κβαντισμού που είναι διαθέσιμα για κωδικοποίηση βρίσκεται από την Εξίσωση 1.1:

$$q = 2^N$$

Όπου, N είναι το bit depth του συστήματος.

Έτσι, ένα σύστημα 8-bit μπορεί να κωδικοποιήσει 2^8 τιμές ή 256 επίπεδα κβαντισμού. Ένα σύστημα 16-bit μπορεί να κωδικοποιήσει 65.536 διαφορετικές τιμές. Το σχήμα 1.9 δείχνει την ιεραρχία των κωδικοποιημένων δεδομένων ήχου. Ως σχεδιαστής συστήματος, πρέπει πρώτα να αποφασίσετε αν

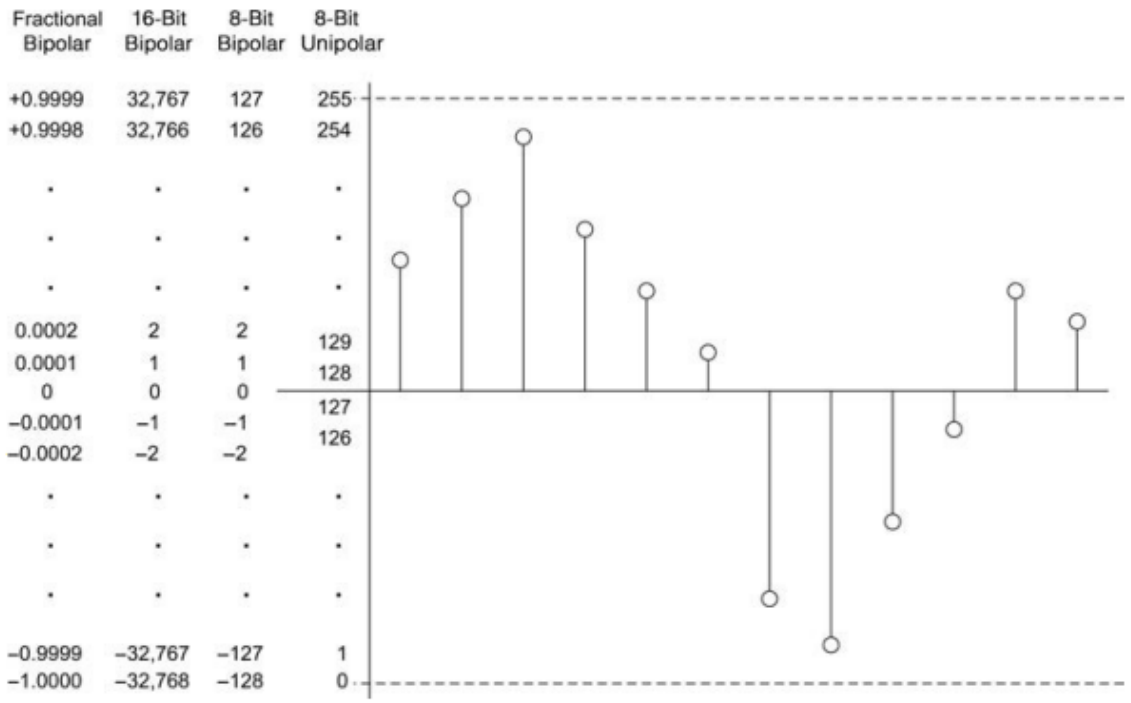
πρόκειται να χειριστείτε μονοπολικά (χωρίς πρόσημο) ή διπολικά (με πρόσημο) δεδομένα. Στη συνέχεια, πρέπει να αποφασίσετε για τους τύπους δεδομένων.



Σχήμα 1.9: Η ιεραρχία της αριθμητικής κωδικοποίησης.

- Τα μονοπολικά ή μη προσημασμένα δεδομένα (unipolar or unsigned) βρίσκονται στο εύρος 0 έως +Max ή -Min έως 0 και έχουν μόνο μία πολικότητα (+ ή -) των δεδομένων, καθώς και τον αριθμό μηδέν (0).
- Τα διπολικά (Bipolar) ή προσημασμένα δεδομένα κυμαίνονται από -Min έως +Max και είναι η πιο συνηθισμένη μορφή σήμερα. Περιλαμβάνει επίσης τον αριθμό μηδέν (0).
- Τα ακέραια δεδομένα (Integer) αναπαρίστανται με ακέραιους αριθμούς και χωρίς δεκαδικό ψηφίο. Ο μη-προσημασμένος ακέραιος ήχος κυμαίνεται από 0 έως +65.535 για συστήματα 16 bit, ενώ ο προσημασμένος ακέραιος ήχος κυμαίνεται από -32.768 έως +32.767 για τον ίδιο ήχο 16-bit. Και στις δύο περιπτώσεις, υπάρχουν 65.536 επίπεδα κβαντισμού.
- Τα κλασματικά δεδομένα (Fractional) κωδικοποιούνται με ένα ακέραιο και ένα κλασματικό τμήμα, που συνδυάζονται ως int.frac με το δεκαδικό ψηφίο να χωρίζει τα μέρη (π.χ. 12.09).

Στο υποσύνολο κλασματικών δεδομένων υπάρχουν τύποι σταθερής και κινητής υποδιαστολής. Τα δεδομένα σταθερού σημείου καθορίζουν τον αριθμό των ψηφίων με πρόσημο σε κάθε πλευρά του δεκαδικού σημείου και συνδυάζονται ως ψηφία intsig, ψηφία frac-sig. Τα δεδομένα "8.16" θα έχουν 8 ψηφία με πρόσημο πριν από το δεκαδικό ψηφίο και 16 μετά από αυτό. Τα δεδομένα κινητής υποδιαστολής έχουν κινούμενη μαντίσα (mantissa) και εκθέτη που κωδικοποιούν τις τιμές σε ένα εύρος που έχει προκαθοριστεί από το Ινστιτούτο Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών (IEEE). Το θετικό και το αρνητικό τμήμα κωδικοποιούνται σε συμπλήρωμα 2, έτσι ώστε η πρόσθεση ακριβώς αντίθετων τιμών (π.χ. -0,5 και +0,5) να οδηγεί πάντα στο μηδέν. Το Σχήμα 1.10 αποκαλύπτει τον τρόπο κωδικοποίησης των επιπέδων κβαντισμού. Οι διακεκομμένες γραμμές αντιπροσωπεύουν την ελάχιστη και τη μέγιστη τιμή.



Σχήμα 1.10: Σύγκριση διαφόρων τύπων αναπαραστάσεων δεδομένων. Η έκδοση κινητής υποδιαστολής είναι προσαρμοσμένη για ένα εύρος από -1,0 έως +0,9999, αν και μπορεί να χρησιμοποιηθεί οποιοδήποτε εύρος.

Ένα θεμελιώδες πρόβλημα είναι ότι ο αριθμός των επιπέδων κβαντισμού θα είναι πάντα ζυγός αριθμός, αφού το 2^N είναι πάντα ζυγός. Στα διπολικά συστήματα (Bipolar systems), θα θέλατε πολύ να έχετε τον αριθμό μηδέν (0) κωδικοποιημένο ως τον αριθμό 0. Αν το κάνετε αυτό, τότε χρησιμοποιείτε ένα από τα επίπεδα κβαντισμού σας γι' αυτό. Αυτό αφήνει έναν περιττό αριθμό επιπέδων που δεν μπορούν να χωριστούν ακριβώς στη μέση. Αυτό δημιουργεί την ανωμαλία που βλέπετε στο Σχήμα 1.10 -υπάρχουν πάντα περισσότερα αρνητικά (2) επίπεδα κβαντισμού από τα θετικά για διπολική κωδικοποίηση. Για τη μονοπολική περίπτωση, δεν υπάρχει καμία τιμή που να κωδικοποιεί ακριβώς το μμηδενικό ή το μμεσαίο επίπεδο- στο Σχήμα 1.10 είναι στη μέση μεταξύ των 127 και 128.

Αυτή η μικρή διαστρέβλωση του εύρους δεδομένων είναι αναπόφευκτη αν σκοπεύετε να χρησιμοποιήσετε τον αριθμό μηδέν στους αλγορίθμους σας, και αυτό είναι σχεδόν βέβαιο. Σε ορισμένα συστήματα ο αλγόριθμος περιορίζει τα αρνητικά δεδομένα ήχου στη δεύτερη πιο αρνητική τιμή. Αυτό συμβαίνει επειδή η αντιστροφή φάσης είναι συνηθισμένη στους αλγορίθμους επεξεργασίας, είτε σκόπιμα είτε με τη μορφή ενός αρνητικής τιμής συντελεστή ή πολλαπλασιαστή. Εάν ένα δείγμα ερχόταν με τιμή 232,768 και είχε αντιστραφεί, δεν θα υπήρχε θετική εκδοχή αυτής της τιμής για κωδικοποίηση. Για την προστασία από αυτό, το 232,768 αντιμετωπίζεται ως 232,767. Τα δεδομένα ήχου που μεταφέρονται από τον προσαρμογέα υλικού ήχου (DSP και κάρτα ήχου) καθώς και τα δεδομένα που αποθηκεύονται σε αρχεία WAV βασίζονται σε υπογεγραμμένους ακέραιους αριθμούς. Ωστόσο, για την επεξεργασία ηχητικού σήματος, προτιμούμε την αναπαράσταση με μονάδες κινητής υποδιαστολής.

1.7.1 Χρησιμοποιώντας δεδομένα κινητής υποδιαστολής

Σε πολλά συστήματα ήχου, είτε DSP είτε plug-in σε δεδομένα είναι διαμορφωμένα έτσι ώστε να βρίσκονται στο εύρος -1,0 έως +1,0 (το οποίο απλοποιεί το πραγματικό εύρος -1,0 έως +0,9999). Στην πραγματικότητα, τα plug-ins που προγραμματίστηκε σε αυτήν την πτυχιακή εργασία χρησιμοποιεί όλα τα δεδομένα που βρίσκονται στο ίδιο εύρος. Ο λόγος έχει να κάνει με την υπερχείλιση (overflow). Στους αλγορίθμους ήχου, η πρόσθεση και ο πολλαπλασιασμός είναι και οι δύο συνηθισμένοι αλγόριθμοι. Με αριθμούς που βασίζονται σε ακέραιους αριθμούς, μπορείτε να μπείτε γρήγορα σε μπελάδες αν συνδυάσετε δύο αριθμούς που οδηγούν σε μια τιμή που βρίσκεται εκτός του εύρους των γνωστών αριθμών.

Εξετάστε τη διπολική μορφή ακεραίων 16 bit στην περιοχή από -32,768 έως +32,767. Οι περισσότερες από τις τιμές σε αυτό το εύρος, όταν πολλαπλασιαστούν μαζί, θα προκύψει ένα αποτέλεσμα που βρίσκεται εκτός αυτών των ορίων. Η πρόσθεση και η αφαίρεση μπορούν επίσης να το προκαλέσουν αυτό, αλλά μόνο για τις μισές από τις πιθανές τιμές. Ωστόσο, οι αριθμοί μεταξύ -1,0 και +1,0 έχουν την ενδιαφέρουσα ιδιότητα ότι το γινόμενό τους είναι πάντα ένας αριθμός σε αυτό το εύρος. Η μετατροπή μιας ακέραιης τιμής σε κλασματική τιμή κατά μήκος του κανονικοποιημένου εύρους -1,0 έως +1,0 σε ένα ψηφιακό σύστημα ήχου N-bit είναι εύκολη, όπως και η αντίστροφη μετατροπή που παρουσιάζεται στην Εξίσωση 1.2:

$$\text{Fraction} = \frac{Intiger}{2^N}$$

$$Intiger = Fraction * 2^N$$

Όπου, N είναι το bit depth του συστήματος.

1.8 Βασικά σήματα DSP

Θα πρέπει να γνωρίζετε τις ακολουθίες δεδομένων για διάφορα θεμελιώδη ψηφιακά σήματα προκειμένου να ξεκινήσετε να κατανοήσετε πώς λειτουργεί η θεωρία DSP. Το βασικό σύνολο σημάτων αποτελείται από:

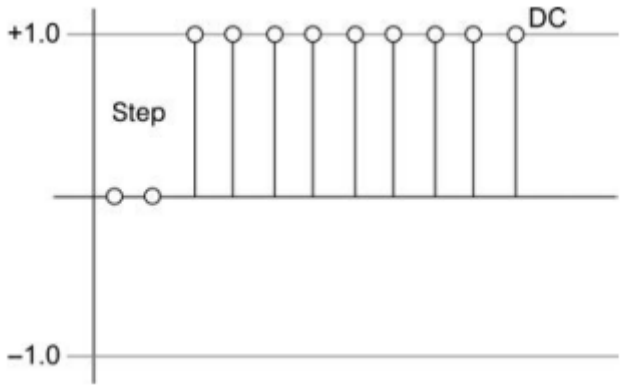
- Άμεσο ρεύμα (Direct Current ή DC) και το βήμα (step): Το DC είναι ένα σήμα 0 Hz
- Nyquist
- ½ Nyquist
- ¼ Nyquist
- Παλμός (Impulse)

Τα τέσσερα πρώτα από αυτά τα σήματα είναι το μόνο που χρειάζεστε για να πάρετε μια γενική ιδέα της απόκρισης συχνότητας ορισμένων βασικών φίλτρων DSP. Τα καλά νέα είναι ότι όλες οι ακολουθίες είναι απλές για να τις θυμάστε. [3]

1.8.1 DSP και ρεύμα

Οι ανταποκρίσεις του DC ή του 0 Hz σήματος και του βήματος μπορούν να βρεθούν και οι δύο με την ακολουθία εισόδου DC/Step: {…0, 0, 1, 1, 1, 1, 1, 1, 1, 1…}.

Αυτό το σήμα στο Σχήμα 1.11 περιέχει δύο μέρη: το βηματικό τμήμα όπου η είσοδος αλλάζει από 0 σε 1 και το τμήμα DC όπου το σήμα παραμένει στη σταθερή στάθμη 1,0 για πάντα. Όταν εφαρμόζετε αυτό το σήμα στο φίλτρο του DSP και εξετάζετε την έξοδο, θα λάβει δύο κομμάτια πληροφορίας. Το τμήμα του βήματος θα σας πει το χρόνο μεταβατικής επίθεσης (transient attack time) και το τμήμα DC θα σας δώσει την απόκριση σε DC ή 0 Hz.

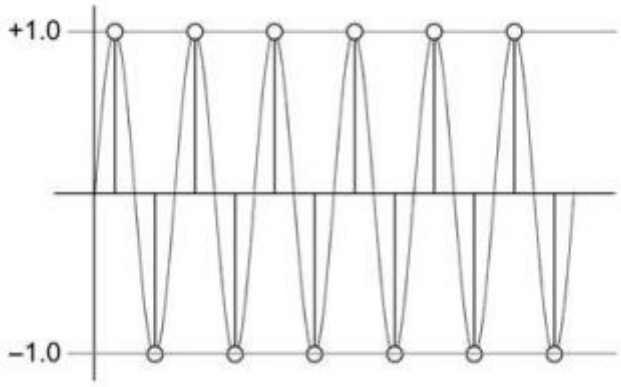


Σχήμα 1.11: Αναπαράσταση της ακολουθίας DC/Step.

1.8.2 Nyquist

Η ακολουθία εισόδου Nyquist αντιπροσωπεύει τη συχνότητα Nyquist του συστήματος και είναι ανεξάρτητη από τον πραγματικό ρυθμό δειγματοληψίας. Η ακολουθία Nyquist είναι {…-1, +1, -1, +1, -1, +1, -1, +1…}.

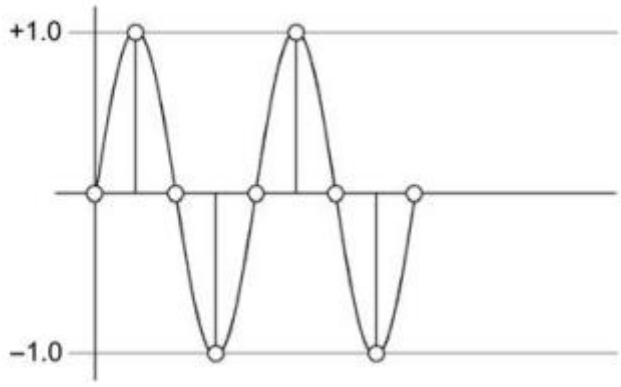
Το σήμα συχνότητας Nyquist στο Σχήμα 1.12 είναι η υψηλότερη συχνότητα που μπορεί να κωδικοποιηθεί. Περιέχει το ελάχιστο από δύο δείγματα ανά κύκλο με κάθε δείγμα να αντιπροσωπεύει τη μέγιστη και την ελάχιστη τιμή. Το ελάχιστο των δύο δειγμάτων είναι ένας άλλος τρόπος για να δηλώσουμε τη Nyquist συχνότητα όπως αυτή σχετίζεται με το θεώρημα δειγματοληψίας.



Σχήμα 1.12: Η ακολουθία Nyquist.

1.8.3 ½ Nyquist

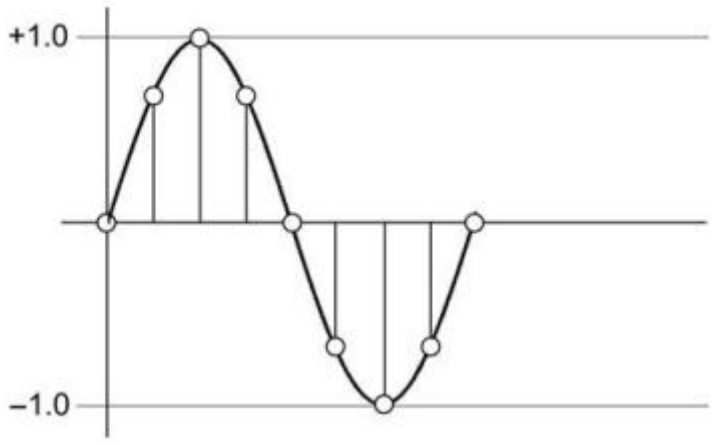
Η ακολουθία εισόδου ½ Nyquist στο Σχήμα 1.13 αντιπροσωπεύει τη συχνότητα ½ Nyquist του συστήματος και είναι ανεξάρτητη από τον πραγματικό ρυθμό δειγματοληψίας. Το σήμα κωδικοποιείται με τέσσερα δείγματα ανά κύκλο, διπλάσια από το Nyquist. Η ακολουθία ½ Nyquist είναι {...-1, 0, +1, 0, -1, 0, +1, 0, -1, 0, +1, 0, -1, 0, +1, 0, -1, 0, +1, 0...}.



Σχήμα 1.13: Η ακολουθία ½ Nyquist έχει τέσσερα δείγματα ανά κύκλο.

1.8.4 ¼ Nyquist

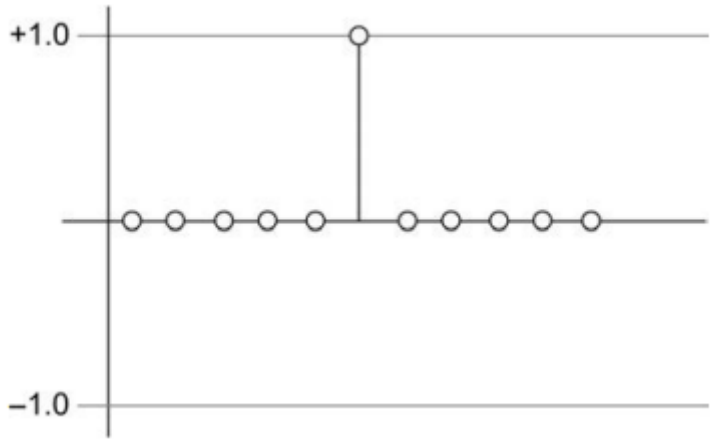
Η ακολουθία εισόδου ¼ Nyquist στο Σχήμα 1.14 αντιπροσωπεύει τη συχνότητα ¼ Nyquist του συστήματος και είναι ανεξάρτητη από τον πραγματικό ρυθμό δειγματοληψίας. Κωδικοποιείται με οκτώ δείγματα ανά κύκλο. Η ακολουθία ¼ Nyquist είναι {...0,0, 0,707, +1,0, 0,707, 0,0, -0,707, -1,0, -0.707, 0.0...}.



Σχήμα 1.14: Η ακολουθία ¼ Nyquist έχει οκτώ δείγματα ανά κύκλο.

1.8.5 Παλμός

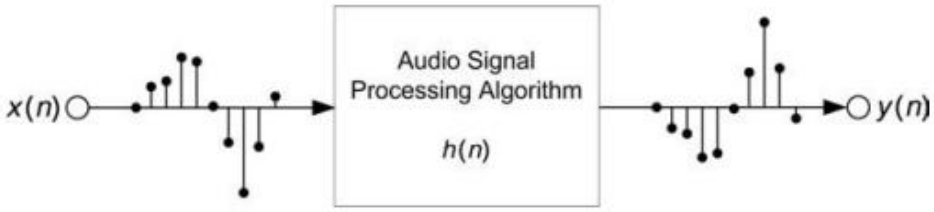
Ο παλμός (impulse) που παρουσιάζεται στο Σχήμα 1.15 είναι ένα μοναδικό δείγμα με την τιμή 1,0 σε μια απεριόριστα μεγάλη ροή μηδενικών. Η παλμική απόκριση ενός αλγορίθμου DSP είναι η έξοδος του αλγορίθμου μετά την εφαρμογή της παλμικής εισόδου. Η παλμική ακολουθία είναι {...0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,...}.



Σχήμα 1.15: Η ακολουθία ¼ Nyquist έχει οκτώ δείγματα ανά κύκλο.

1.9 Αλγόριθμοι επεξεργασίας σήματος

Με την ευρύτερη έννοια, ένας αλγόριθμος είναι ένα σύνολο εντολών που ολοκληρώνει μια προκαθορισμένη εργασία. Ο βρόχος επεξεργασίας σήματος που είδαμε προηγουμένως στο Σχήμα 1.8 είναι μια εικόνα ενός αλγορίθμου για την επεξεργασία ήχου και χειριστηρίου (UI) σε πραγματικό χρόνο. Στην εξειδικευμένη περίπτωση της επεξεργασίας ηχητικού σήματος, ένας αλγόριθμος είναι ένα σύνολο εντολών που επεξεργάζεται δεδομένα μιας ροής bit για την παραγωγή και έξοδο του ήχου. Σε αυτήν την ενότητα θα δούμε την επεξεργασία εισερχόμενων δεδομένων ήχου και τη μετατροπή τους σε μια επεξεργασμένη έξοδο. Ωστόσο, η σύνθεση μιας κυματομορφής προς έξοδο πληροί επίσης τις προϋποθέσεις και σε αυτήν τη ειδική περίπτωση, δεν υπάρχει είσοδος ήχου σε πραγματικό χρόνο προς επεξεργασία. Τα περισσότερα από τα plug-ins χρησιμοποιούν το μοντέλο εφέ, όπου μια ακολουθία δειγμάτων εισόδου επεξεργάζεται για να δημιουργήσει μια ακολουθία, όπως φαίνεται στο σχήμα 1.16.



Σχήμα 1.16: Ένας αλγόριθμος επεξεργασίας ηχητικού σήματος που μετατρέπει μια ακολουθία εισόδου x(n) σε μια ακολουθία εξόδου y(n).

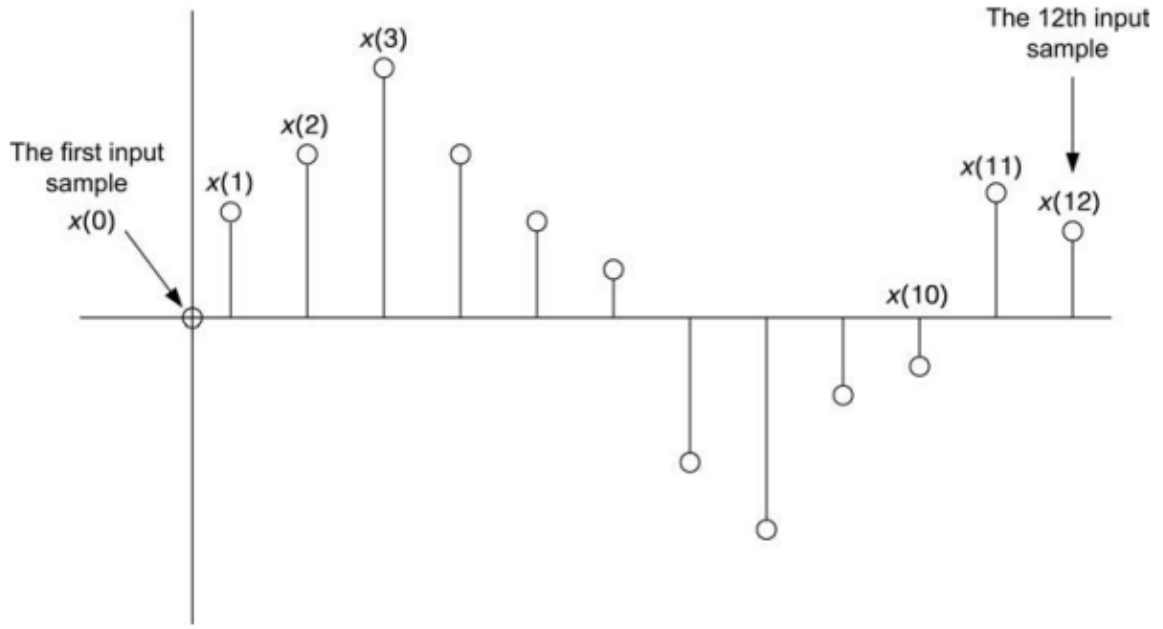
Συνθήκες και κανόνες:

- Το x(n) είναι πάντα η ακολουθία εισόδου - η μεταβλητή n αντιπροσωπεύει τη θέση του n-οστού δείγματος της ακολουθίας x.
- y(n) είναι πάντα η ακολουθία εξόδου- η μεταβλητή n αντιπροσωπεύει τη θέση του n-οστού δείγματος της ακολουθίας y.
- Το h(n) είναι η παλμική απόκριση του αλγορίθμου - μια ειδική ακολουθία που αντιπροσωπεύει την έξοδο του αλγορίθμου για ένα μόνο δείγμα εισόδου ή παλμού.
- Για επεξεργασία σε πραγματικό χρόνο, ο αλγόριθμος πρέπει να δέχεται ένα νέο δείγμα εισόδου (ή σύνολο δειγμάτων), να κάνει την επεξεργασία, και στη συνέχεια να έχει διαθέσιμο το δείγμα (ή τα δείγματα) εξόδου πριν από το επόμενο δείγμα. Εάν η επεξεργασία πάρει πολύ χρόνο, ηχητικά clicks, pops, glitches και θόρυβος θα είναι το αποτέλεσμα σε πραγματικό χρόνο.

1.9.1 Bookkeeping

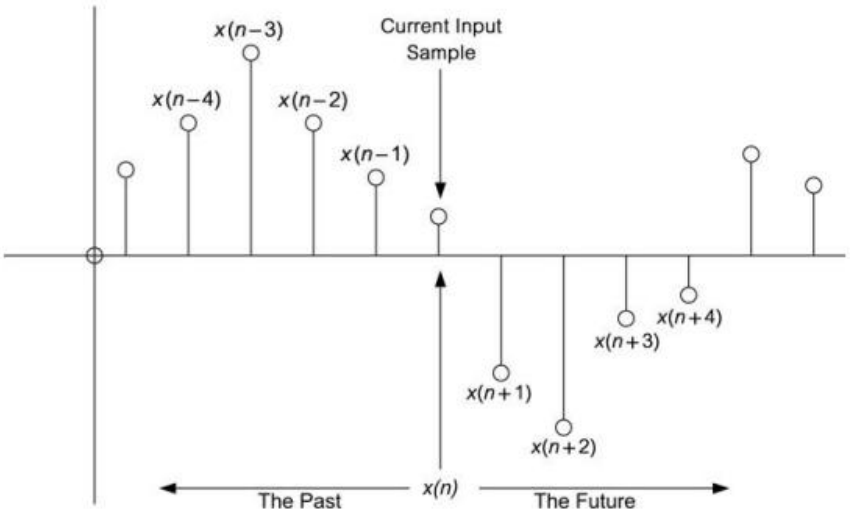
Μπορείτε να δείτε ότι υπάρχουν ήδη τρεις ακολουθίες που πρέπει να αντιμετωπιστούν: η είσοδος, η έξοδος και η παλμική απόκριση, οι οποίες είναι όλες κωδικοποιημένες με την ίδια μεταβλητή n για να εντοπίζεται η θέση των δειγμάτων μέσα στην ακολουθία. Το πρώτο βήμα είναι να αποφασίσουμε πώς θα χρησιμοποιήσουμε το n για να κάνουμε αυτή την bookkeeping στρατηγική. Η χρήση του για την αναπαράσταση της απόλυτης θέσης στην ακολουθία θα γινόταν γρήγορα κουραστική - όπως αντιμετωπίζουμε αριθμούς ευρετηρίου.

Το σχήμα 1.17 δείχνει ένα σήμα εισόδου, x(n), που ξεκινά από t = 0 ή x(0). Το δείγμα x(0) είναι το πρώτο δείγμα που εισέρχεται στον αλγόριθμο επεξεργασίας σήματος. Στο μεγάλο σχήμα των πραγμάτων, το x(0) θα είναι το παλαιότερο δείγμα εισόδου που θα υπάρχει ποτέ. Η ευρετηρίαση των αριθμών με απόλυτη θέση πρόκειται να να είναι δύσκολη υπόθεση, καθώς οι τιμές του δείκτη θα γίνουν μεγάλες, ειδικά σε πολύ υψηλούς ρυθμούς δειγματοληψίας.

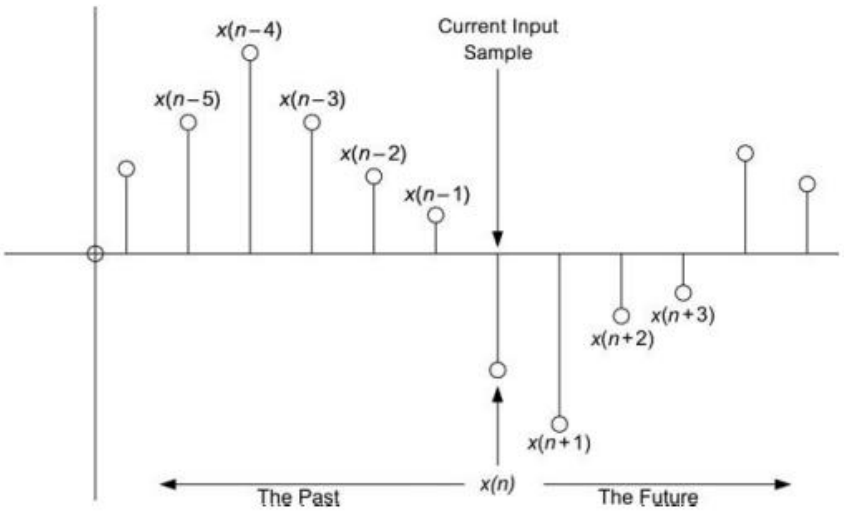


Σχήμα 1.17: Η χρήση της απόλυτης θέσης μέσα στην ακολουθία είναι ένας τρόπος για να παρακολουθείτε, αλλά οι τιμές του δείκτη θα γίνουν πολύ μεγάλες πολύ γρήγορα.

Ένα άλλο πρόβλημα με την αντιμετώπιση της απόλυτης θέσης των δειγμάτων είναι ότι οι αλγόριθμοι δεν χρησιμοποιούν την απόλυτη θέση του δείγματος στην κωδικοποίησή τους. Αντ' αυτού, οι αλγόριθμοι χρησιμοποιούν τη θέση του τρέχοντος δείγματος και κάνουν τα πάντα σχετικά με αυτό το δείγμα. Στην επόμενη δειγματοληπτική περίοδο, τα πάντα επαναπροσδιορίζονται και πάλι σε σχέση με το τρέχον δείγμα. Μπορεί αρχικά να ακούγεται μπερδεμένο, αλλά είναι μια καλύτερη μέθοδος για την παρακολούθηση των δειγμάτων και κυρίως για τον καθορισμό των χαρακτηριστικών εισόδου/εξόδου του αλγορίθμου, που ονομάζεται συνάρτηση μεταφοράς. Στο Σχήμα 1.18 παρουσιάζεται το σήμα εισόδου παγωμένο στην τρέχουσα χρονική στιγμή, x (n), και τα άλλα δείγματα δεικτοδοτούνται με βάση τη θέση του. Μια δειγματοληπτική περίοδο αργότερα (Σχήμα 1.19) μπορείτε να δείτε ότι το πλαίσιο αναφοράς έχει μετατοπιστεί προς τα δεξιά και ότι το x (n) έχει γίνει τώρα x(n - 1).



Σχήμα 1.18: Οι αλγόριθμοι DSP χρησιμοποιούν την τρέχουσα θέση του δείγματος ως θέση αναφοράς και όλα τα άλλα δείγματα δεικτοδοτούνται με βάση αυτό το δείγμα. Εδώ μπορείτε να δείτε την τρέχουσα κατάσταση του αλγορίθμου παγωμένη στο χρόνο στο τρέχον δείγμα εισόδου x(n).

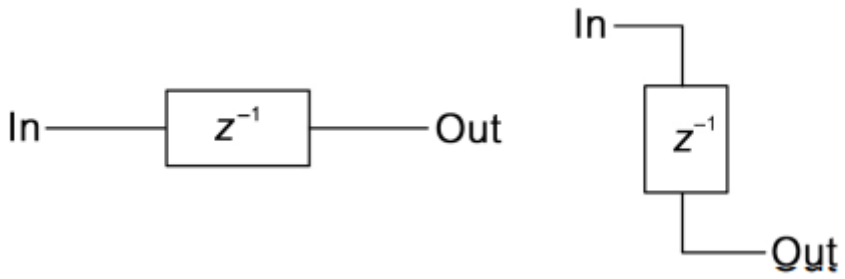


Σχήμα 1.19: Μια δειγματοληπτική περίοδο αργότερα, όλα έχουν αλλάξει. Το προηγούμενο x(n) έχει τώρα δείκτη x(n - 1) και αυτό που ήταν το επόμενο δείγμα, x(n + 1) γίνεται τώρα το τρέχον δείγμα.

1.9.2 Καθυστέρηση δείγματος

Ενώ τα αναλογικά κυκλώματα επεξεργασίας, όπως τα κυκλώματα ελέγχου τόνου, χρησιμοποιούν πυκνωτές και πηνία για να μεταβάλλουν τη φάση και την καθυστέρηση του αναλογικού σήματος, οι ψηφιακοί αλγόριθμοι χρησιμοποιούν τη χρονική καθυστέρηση. Στα διαγράμματα αλγορίθμων, η καθυστέρηση αναπαρίσταται από ένα πλαίσιο με το γράμμα z μέσα. Ο όρος z θα έχει έναν εκθέτη, όπως z^{-5} ή z^{+2} ή z^{0} - ο εκθέτης κωδικοποιεί την καθυστέρηση σε δείγματα ακολουθώντας τους ίδιους κανόνες με την bookkeeping στρατηγική, που είδαμε προηγουμένως, με αρνητικούς (-) εκθέτες που αντιπροσωπεύουν μια καθυστέρηση σε χρόνο προς τα πίσω (πρώην δείγματα που λάβαμε) και οι θετικοί (+) αντιπροσωπεύουν καθυστέρηση στο χρόνο προς τα εμπρός (στα μελλοντικά δείγματα που θα έρθουν). Ονομάζετε z ο τελεστής καθυστέρησης και όπως φαίνεται, η χρονική καθυστέρηση θα αντιμετωπιστεί ως μαθηματική πράξη.

Πιθανόν να αναρωτιέστε πώς μπορείτε να έχετε μια θετική καθυστέρηση προς το μέλλον και η απάντηση είναι ότι δεν μπορείτε για την επεξεργασία σήματος σε πραγματικό χρόνο. Στην επεξεργασία σε πραγματικό χρόνο δεν ξέρετε ποτέ ποιο δείγμα θα έρθει μετά. Ωστόσο, σε επεξεργασία μη πραγματικού χρόνου (για παράδειγμα, σε ένα αρχείο ήχου που επεξεργαζόμαστε από μία ηχογράφηση) γνωρίζετε τα μελλοντικά δείγματα επειδή βρίσκονται στο αρχείο. Στο Σχήμα 1.20 παρουσιάζονται δύο συνηθισμένοι τρόποι για να δηλώσουμε μια καθυστέρηση ενός δείγματος σε ένα διάγραμμα αλγορίθμου.



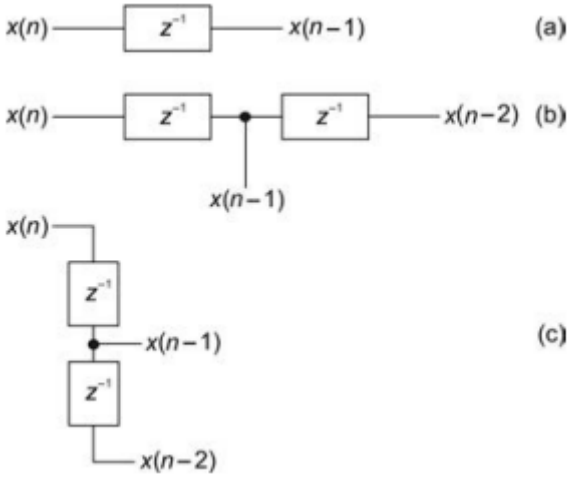
Σχήμα 1.20: Δύο συνηθισμένοι τρόποι για να σχεδιάσετε μια καθυστέρηση - η καθυστέρηση ενός δείγματος αναπαρίσταται με το z^{-1}. Και οι δύο εκδοχές είναι ισοδύναμες.

Κανόνες καθυστέρησης:

- Κάθε φορά που ένα δείγμα μπαίνει στον καταχωρητή καθυστέρησης (θέση μνήμης), το προηγουμένως αποθηκευμένο δείγμα απορρίπτεται.
- Το δείγμα που απορρίπτεται μπορεί να χρησιμοποιηθεί για επεξεργασία ή να διαγραφεί.
- Τα στοιχεία καθυστέρησης μπορούν να συνδεθούν σε κλιμακωτή διάταξη με την έξοδο του ενός, να τροφοδοτεί την είσοδο του επόμενου, για να δημιουργηθεί περισσότερος χρόνος καθυστέρησης.

Εάν ένα δείγμα x(n) εισέρχεται στο στοιχείο καθυστέρησης ενός δείγματος, τότε πώς ονομάζετε το δείγμα που αποβάλλεται; Είναι το προηγούμενο δείγμα που μπήκε μέσα, ένα δειγματικό χρονικό

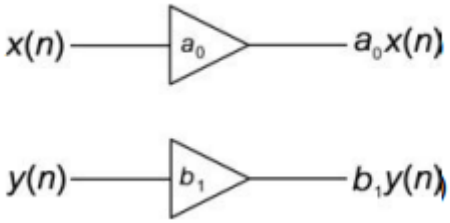
διάστημα στο παρελθόν. Έτσι, το δείγμα εξόδου είναι το x(n-1). Στο Σχήμα 1.21 μπορείτε να δείτε πώς τα στοιχεία καθυστέρησης μπορούν να κλιμακωθούν με εξόδους που λαμβάνονται σε πολλαπλές θέσεις δημιουργώντας πολλαπλά δείγματα για να τα χρησιμοποιήσει ο αλγόριθμος.



Σχήμα 1.21: Τρεις αλγόριθμοι καθυστέρησης: (a) καθυστέρηση ενός δείγματος, (b) δύο καθυστέρηση ενός δείγματος σε κλιμάκωση, που παράγουν δύο διαφορετικές εξόδους, x(n - 1) και x(n - 2), παρατηρήστε ότι ο (c) είναι λειτουργικά πανομοιότυπος με τον (b).

1.9.3 Πολλαπλασιασμός

Το επόμενο κατασκευαστικό στοιχείο του αλγορίθμου είναι η πράξη του κλιμακωτού πολλαπλασιασμού. Πρόκειται για έναν τελεστή δειγμάτων ανά δείγμα που απλώς πολλαπλασιάζει τα δείγματα εισόδου με έναν συντελεστή. Ο τελεστής πολλαπλασιασμού χρησιμοποιείται σχεδόν σε κάθε αλγόριθμο DSP. Το Σχήμα 1.22 δείχνει τον τελεστή πολλαπλασιασμού σε δράση. Οι είσοδοι απλά κλιμακώνονται με τους συντελεστές.

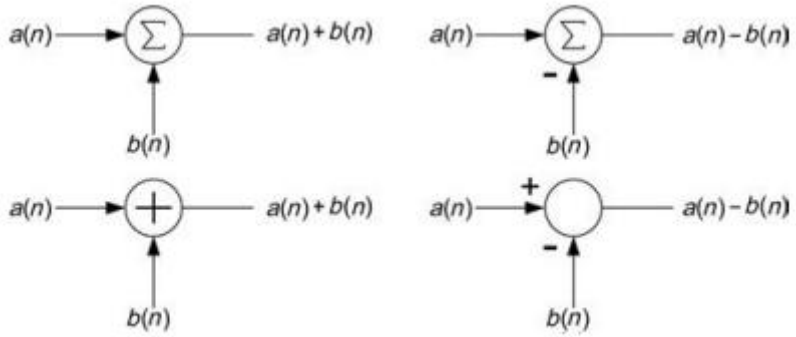


Σχήμα 1.22: Ο τελεστής πολλαπλασιασμού εμφανίζεται ως τρίγωνο με ένα γράμμα ή ένας αριθμός συντελεστή μέσα.

1.9.4 Πρόσθεση και αφαίρεση

Η πρόσθεση και η αφαίρεση είναι στην πραγματικότητα το ίδιο είδος πράξης - η αφαίρεση είναι η πρόσθεση ενός αρνητικού αριθμού. Υπάρχουν πολλά διαφορετικά σύμβολα αλγορίθμου για να δηλώσουν την πρόσθεση και την αφαίρεση. Η λειτουργία της ανάμειξης σημάτων είναι στην

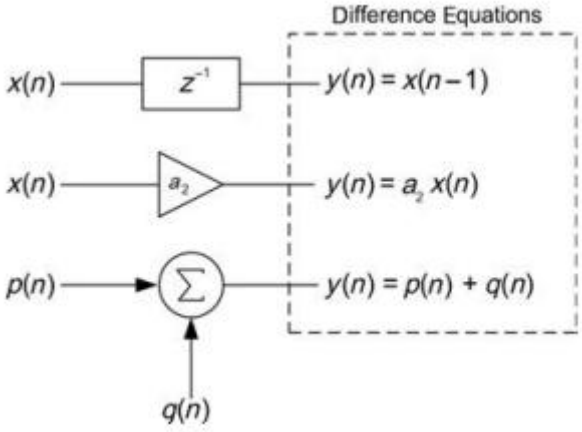
πραγματικότητα η μαθηματική λειτουργία της πρόσθεσης. Το Σχήμα 1.23 δείχνει διάφορους τρόπους απεικόνισης της λειτουργίας της πρόσθεσης και της αφαίρεσης σε διαγράμματα.



Σχήμα 1.23: Διαγράμματα πρόσθεσης και αφαίρεσης για δύο ακολουθίες εισόδου a(n) και b(n), πρόκειται για όλες τις κοινά χρησιμοποιούμενες μορφές των ίδιων συναρτήσεων.

1.9.5 Εξίσωση διαφοράς

Κατά σύμβαση, η ακολουθία εξόδου του αλγορίθμου DSP ονομάζεται y(n) και η μαθηματική εξίσωση που τη συσχετίζει με την είσοδο ονομάζεται εξίσωση διαφοράς (Difference Equation). Ο συνδυασμός των πράξεων θα σας δώσει μια καλύτερη ιδέα για το πώς μοιάζει η εξίσωση διαφοράς. Στο Σχήμα 1.24 παρουσιάζονται οι εξισώσεις διαφοράς για διάφορους συνδυασμούς στοιχείων των αλγορίθμων. Η έξοδος y(n) είναι ένας μαθηματικός συνδυασμός των εισόδων.

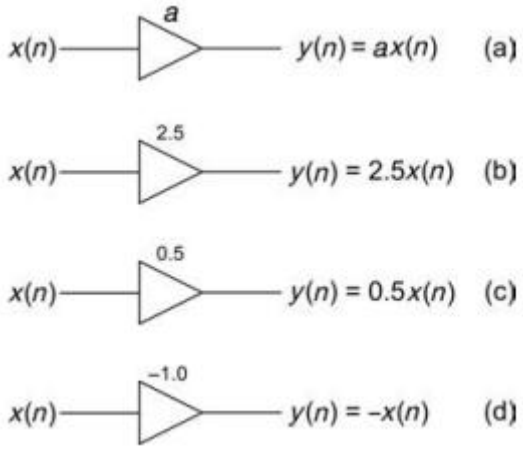


Σχήμα 1.24: Οι εξισώσεις διαφοράς συσχετίζουν την είσοδο (ή τις εισόδους) με την έξοδο μέσω μαθηματικών πράξεων.

1.9.6 Ενίσχυση, εξασθένηση και αντιστροφή φάσης

Όπως φαίνεται στο Σχήμα 1.25, ένας απλός πολλαπλασιαστής συντελεστή θα χειριστεί τις τρεις βασικές λειτουργίες επεξεργασίας ήχου, δηλαδή την ενίσχυση (Gain), την εξασθένηση (Attenuation)

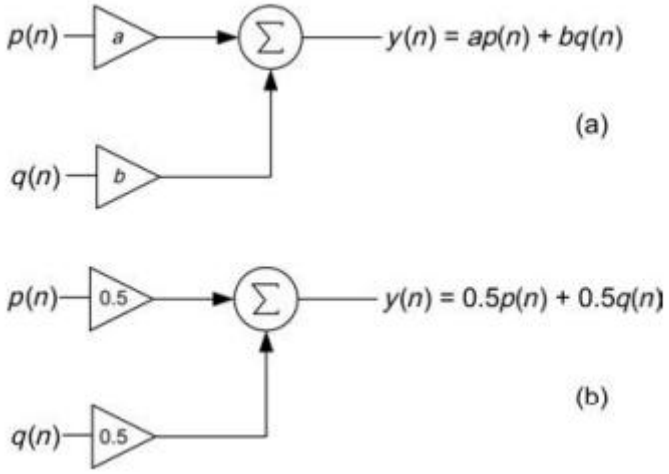
και την αντιστροφή. Εάν ο συντελεστής είναι αρνητικός αριθμός, το αποτέλεσμα θα είναι η αντιστροφή φάσης (Phase Inversion). Εάν ο συντελεστής έχει μέγεθος μικρότερο από 1,0, θα προκύψει εξασθένηση, ενώ ενίσχυση θα προκύψει εάν το μέγεθος είναι μεγαλύτερο από 1,0.



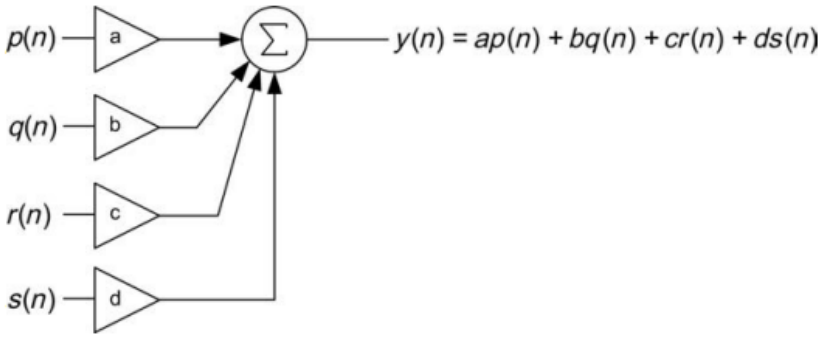
Σχήμα 1.25: Παραδείγματα απλών αλγορίθμων πολλαπλασιαστών. Παρατηρήστε τη διαφορετική σημειογραφία με τον συντελεστή να τοποθετείται έξω από το τρίγωνο - αυτός είναι ένας άλλος συνηθισμένος τρόπος για τον προσδιορισμό του. (a) Απλός κλιμακωτός πολλαπλασιασμός με μια αυθαίρετη τιμή "a". (b) Η ενίσχυση επιτυγχάνεται με ένα μέγεθος συντελεστή μεγαλύτερο από τη μονάδα. (c) Η εξασθένηση μειώνει το μέγεθος της τιμής εισόδου με συντελεστή μικρότερο της μονάδας. (d) Η αντιστροφή φάσης αντιστρέφει το σήμα ανάποδα με τη χρήση αρνητικού συντελεστή. Μια τιμή -1,0 αντιστρέφει τελείως το σήμα.

1.9.7 Αλγόριθμος μίξης

Ένα πρόβλημα με τη μίξη πολλαπλών καναλιών ψηφιακού ήχου είναι η πιθανότητα υπερχείλισης ή δημιουργίας μιας τιμής δείγματος που βρίσκεται εκτός του εύρους του συστήματος. Είδατε ότι με τον περιορισμό του διπολικού κλασματικού συστήματος στα όρια -1,0 έως +1,0, ο πολλαπλασιασμός οποιουδήποτε από αυτούς τους αριθμούς οδηγεί πάντα σε έναν αριθμό που είναι μικρότερος από οποιονδήποτε από τους δύο και πάντα εντός του ίδιου εύρους -1,0 έως +1,0. Ωστόσο, η πρόσθεση σημάτων μπορεί εύκολα να δημιουργήσει τιμές εκτός των ορίων ±1. Προκειμένου να παρακάμψουν αυτό το πρόβλημα, τα κυκλώματα μίξης N-καναλιών ενσωματώνουν αλγόριθμους εξασθένισης για να μειώσουν το μέγεθος των εισόδων, όπου η τιμή εξασθένησης είναι 1/N. Κατά τη μίξη δύο καναλιών, οι αλγόριθμους εξασθένισης έχουν τιμή ½ ο καθένας, ενώ ένας τρικάναλος μίκτης θα είχε αλγόριθμους εξασθένισης με τιμή 1/3 σε κάθε κλάδο μίξης. Εάν όλα τα κανάλια τυχαίνει να έχουν μέγιστη ή ελάχιστη τιμή μια τιμή 1/3 σε κάθε κλάδο μίξης. Εάν όλα τα κανάλια τυχαίνει να έχουν μέγιστη ή ελάχιστη τιμή στον ίδιο χρόνο δειγματοληψίας, το άθροισμα ή η διαφορά τους θα εξακολουθεί να περιορίζεται σε ±1. Τα σχήματα 1.26 και 1.27 δείχνουν τους γενικευμένους αλγορίθμους μίξης και βεβαρημένου αθροίσματος.



Σχήμα 1.26: (a) Γενικευμένος αλγόριθμος πρόσθεσης/ανάμειξης με ξεχωριστό συντελεστή σε κάθε γραμμή και (b) ένας κανονικοποιημένος αλγόριθμος ανάμειξης που δεν θα υπερχειλίζει, ούτε θα αποκόπτεται (clipping).



Σχήμα 1.27: Παράδειγμα αλγορίθμου σταθμισμένου αθροίσματος - κάθε δείγμα από κάθε κανάλι έχει το δικό του συντελεστή στάθμισης, a-d.

1.10 Επίλογος

Κλείνοντας το πρώτο κεφάλαιο εξετάσαμε μια σειρά από καίριες αρχές και τεχνικές που συνδέονται με την επεξεργασία του ψηφιακού ηχητικού σήματος (DSP). Από τη δειγματοληψία μέχρι τους βασικούς αλγορίθμους επεξεργασίας, εξετάσαμε τη δομή και τις αρχές που καθορίζουν τη συμπεριφορά των ψηφιακών συστημάτων. Κατανοήσαμε τη σημασία της δειγματοληψίας και της ανακατασκευής του σήματος στην αναπαραγωγή του ήχου με ακρίβεια. Αναλύσαμε τα βασικά στοιχεία των συστημάτων επεξεργασίας σήματος και τον τρόπο με τον οποίο επιτυγχάνεται η επεξεργασία σε πραγματικό χρόνο.

Η έρευνα επικεντρώθηκε στην αναπαράσταση των ηχητικών δεδομένων και τους βασικούς αλγορίθμους όπως ο πολλαπλασιασμός, η πρόσθεση, η αφαίρεση, η ενίσχυση, η εξασθένηση, και η αντιστροφή φάσης. Επίσης, εξετάσαμε βασικά σήματα DSP και αλγόριθμους επεξεργασίας που συνθέτουν τη βάση των συστημάτων DSP.

Όλα αυτά για να μπορούμε να κατανοήσουμε και να χρησιμοποιήσουμε αυτά τα εργαλεία για να πετύχουμε το εφέ της αντήχησης, όπως θα δούμε στην δεύτερη ενότητα.

Κεφάλαιο 2ο: Αντήχηση

2.1 Εισαγωγή

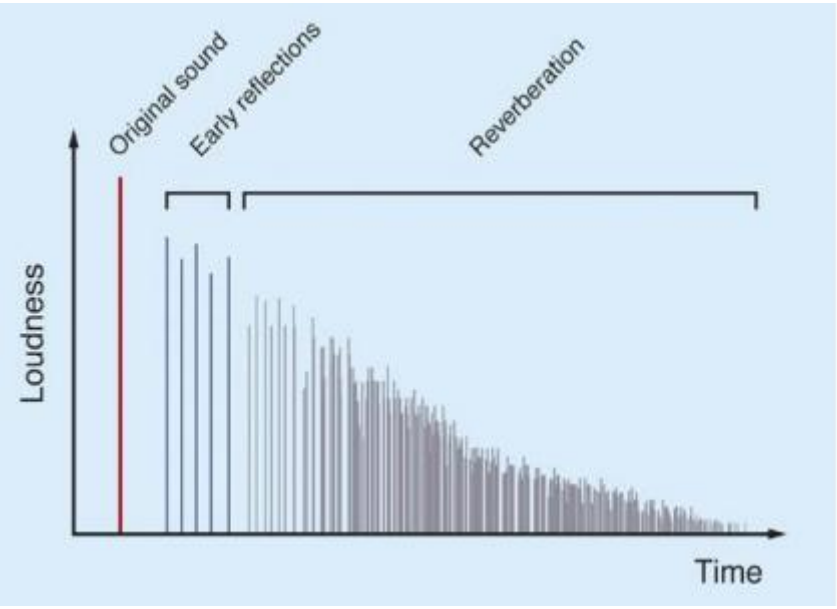
Στον πραγματικό κόσμο, η αντήχηση συμβαίνει φυσικά όταν ο ήχος αναπηδά και ανακλάται ξανά από τοίχους, οροφές και άλλες επιφάνειες μέσα σε ένα μεγάλο δωμάτιο, και μια ηλεκτρονική μονάδα αντήχησης μιμείται αυτό το φαινόμενο δημιουργώντας ηλεκτρονικά χιλιάδες ανακλάσεις. Η αντήχηση μπορεί να χρησιμοποιηθεί για τη δημιουργία της εντύπωσης ενός πραγματικού δωματίου, αλλά μπορεί επίσης να χρησιμοποιηθεί για την ενίσχυση νέων εφέ που δεν έχουν απόλυτη αντιστοίχιση στη φύση.

Εάν ο μηχανικός ήχου ηχογραφήσει σε ξηρό περιβάλλον και στη συνέχεια προσθέσει κάποια εφέ κατά τη διάρκεια μιας μίξης, ο μηχανικός μπορεί να πειραματιστεί με διαφορετικούς τύπους εφέ μετά την ολοκλήρωση της ηχογράφησης. Αυτό είναι μία πρακτική κατάσταση στο στούντιο με έμπειρους μηχανικούς μίξης, πολλά κομμάτια και ποικιλία διαφορετικών εφέ.

Για μία ηχογραφημένη φωνή τραγουδιού, ωστόσο, συνοδεύεται σχεδόν πάντα από μουσικά όργανα που στις περισσότερες περιπτώσεις είναι αρμονικά, και συσχετίζονται με το τραγούδι, δεδομένου ότι συντίθενται ώστε να αποτελούν ένα συνεκτικό σύνολο με τη φωνή του τραγουδιού [7].

Σε ένα δωμάτιο συμβαίνουν πολυάριθμες ανακλάσεις επειδή ο ήχος που εκπέμπεται από μια πηγή χτυπά στις επιφάνειες του δωματίου και σε άπειρες γωνίες. Ο ήχος προσκρούει σε επιφάνειες και ανακλάσεις μέχρι όλη του η ενέργεια να απορροφηθεί και να εξαφανιστεί.

Ο πρώτος ήχος ονομάζεται άμεσος ήχος (direct sound) ή αυθεντικός ήχος (original sound) και ο επόμενος έρχεται γνωστός ως τις πρώιμες ανακλάσεις (early reflections) [8]. Οι ανακλάσεις επιτρέπουν στον ήχο να αναπηδήσει για μικρό χρονικό διάστημα μέχρι να απορροφηθεί. Αυτό το χαρακτηριστικό του ήχου ονομάζεται αντήχηση (reverbation). Αυτές οι ανακλάσεις διαδίδουν επίσης τον ήχο σε ολόκληρο το δωμάτιο. Το σχήμα 2.1 αναπαριστά αυτές τις ιδιότητες του ήχου και της αντήχησης στους άξονες της έντασης (loudness) και του χρόνου (time).



Σχήμα 2.1: Χαρακτηριστικό του ήχου και της αντήχησης

2.2 Χρόνος αντήχησης

Ο χρόνος αντήχησης (T60) ορίζεται ως ο χρόνος σε δευτερόλεπτα που χρειάζεται ένας ήχος για να υποχωρήσει κατά 60 dB μετά τη διακοπή της ηχητικής πηγής. Γενικά, ο χρόνος αντήχησης αυξάνεται όσο αυξάνεται ο όγκος του χώρου και ο χρόνος αντήχησης μειώνεται όσο αυξάνεται η συνολική ποσότητα απορρόφησης, όπως θα ήταν αναμενόμενο [9].

Ο όγκος και η απορρόφηση μπορούν να υπολογιστούν. Οι χρόνοι αντήχησης μπορούν να υπολογιστούν από τον όγκο και την απορρόφηση. Η απορρόφηση των περισσότερων υλικών εξαρτάται από τη συχνότητα. Ως άμεσο αποτέλεσμα, οι χρόνοι αντήχησης στους χώρους εξαρτώνται από τη συχνότητα. Κατά συνέπεια, η συνολική απορρόφηση σε ένα δωμάτιο πρέπει να υπολογίζεται ξεχωριστά σε κάθε ζώνη συχνοτήτων που μας ενδιαφέρει.

Το ποσό της απορρόφησης που παρέχει ένα υλικό (σε μια δεδομένη συχνότητα) είναι η επιφάνεια αυτού του υλικού επί τον συντελεστή απορρόφησής του (στη δεδομένη συχνότητα). Το συνολικό απορρόφηση σε ένα δωμάτιο (σε δεδομένη συχνότητα) είναι το άθροισμα της απορρόφησης (σε δεδομένη συχνότητα) καθενός από τα υλικά του δωματίου.

Την μαθηματική όψη αυτού του τύπου περιγράφει η Εξίσωση 2.1:

$$A = \sum Sa = \sum a = \mathrm{S}_1\alpha_1 + \mathrm{S}_2\alpha_2 \ldots \mathrm{S}_n\alpha_n$$

Όπου,

- A = Συνολική απορρόφηση σε sabins (μονάδα μέτρησης ηχοαπορρόφησης σε τετραγωνικά πόδια)
- S = Επιφάνεια κάθε υλικού σε τετραγωνικά πόδια
- α = Συντελεστής απορρόφησης

Η εξίσωση Sabine είναι η πιο συχνά χρησιμοποιούμενη από τις εξισώσεις αντήχησης επειδή παράγει χρήσιμα αποτελέσματα σε μεγάλη ποικιλία χώρων. Οι συντελεστές απορρόφησης χρησιμοποιούνται στον υπολογισμό των χρόνων αντήχησης σε χώρους. Η απορρόφηση οποιουδήποτε υλικού μετρείται σε πολλές συχνότητες για τον προσδιορισμό των χαρακτηριστικών απορρόφησής τους [10].

Οι χρόνοι αντήχησης σε κάθε ζώνη συχνοτήτων ενδιαφέροντος μπορούν να υπολογιστούν ως εξής:

$$T_{60} = \frac{.05V}{A}$$

Όπου,

- T60 = Χρόνος αντήχησης σε δευτερόλεπτα
- V = Όγκος του δωματίου κυβικά πόδια
- A = Συνολική απορρόφηση σε sabins
- .05 = Σταθερά σε δευτερόλεπτα ανά πόδι

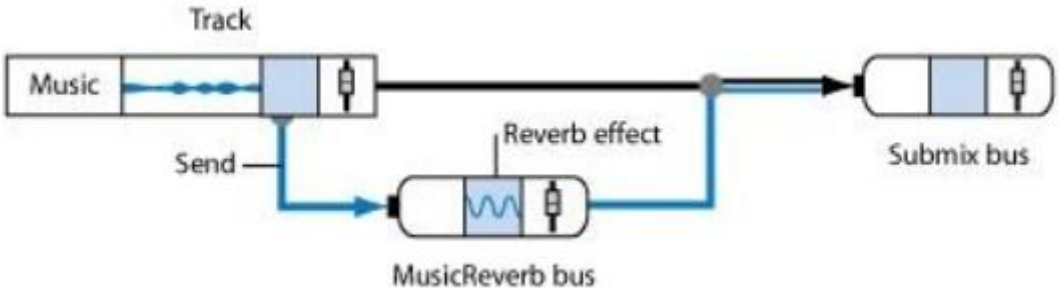
Αν και ο διαχωρισμός ομιλίας έχει μελετηθεί εκτενώς, λίγες μελέτες είναι αφιερωμένες στο διαχωρισμό της φωνής από τη μουσική συνοδεία. Η φωνή του τραγουδιού έχει πολλές ομοιότητες με την ομιλία. Για παράδειγμα, και οι δύο αποτελούνται από φωνητικούς και άφωνους ήχους. Αλλά οι διαφορές μεταξύ τραγουδιού και ομιλίας είναι επίσης σημαντικές. Μια πολύ γνωστή διαφορά είναι η παρουσία μιας πρόσθετης σχηματιστικής ζώνη, που ονομάζεται φορμάντο τραγουδιού, στην περιοχή συχνοτήτων 2000-3000 Hz στο οπερατικό τραγούδι [7]. Πρέπει να επισημανθεί ότι τα ηχητικά πεδία στα δωμάτια είναι εξαιρετικά πολύπλοκα, και πρέπει να γίνουν ορισμένες απλουστευτικές παραδοχές προκειμένου να εξάγουμε διαχειρίσιμες εξισώσεις.

Για παράδειγμα, στην εξίσωση Sabine υποθέτει ότι το ηχητικό πεδίο είναι ομοιόμορφα κατανεμημένο σε όλο το δωμάτιο και παραμελεί τις συγκεκριμένες θέσεις των διαφόρων απορροφητικών υλικών, όπως καθώς και το σχήμα του δωματίου. Επιπλέον, παρόλο που το T60 είναι ένας θεμελιώδης περιγραφικός δείκτης χώρου, άλλες εκτιμήσεις όπως η αρχική χρονική καθυστέρηση, η χωρική εντύπωση, το μείγμα και η ελευθερία από ενοχλητικούς θορύβους μπορεί επίσης να είναι σημαντικές παράμετροι.

2.3 Ψηφιακή αντήχηση

Σχεδόν όλες οι εικονικές ψηφιακές αντηχήσεις χρησιμοποιούν έναν ενιαίο χρόνο αντήχησης για όλο το εύρος των συχνοτήτων, καθώς και έναν ενιαίο αλγόριθμο. Ορισμένα μοντέλα μπορούν να μεταβάλλουν το χρόνο αντήχησης με συχνότητα και ορισμένα μπορούν να ισοσταθμίσουν (equalize) την είσοδο του σήματος, γεγονός που μπορεί να βελτιώσει τον ήχο αλλάζοντας το χρώμα του τόνου.

Η βάση της ιδέας είναι τα βασικά χαρακτηριστικά του ήχου που πολλές ροές ήχου μπορούν να αναμειχθούν σε ροή ισοδύναμη με μία μόνο ροή σε μέγεθος. Όταν διάφοροι ήχοι είναι ακουστοί, μπορούμε να ακούσουμε μερικούς από τους πιο δυνατούς ήχους. Αυτό οφείλεται στο γεγονός ότι οι ήχοι που έχουν διαφορετική συχνότητα αντισταθμίζουν ο ένας τον άλλον. Με άλλα μέσα όπως το βίντεο, το κείμενο, εμείς δεν μπορούμε να δούμε αυτά τα φαινόμενα ανάμειξης [11].



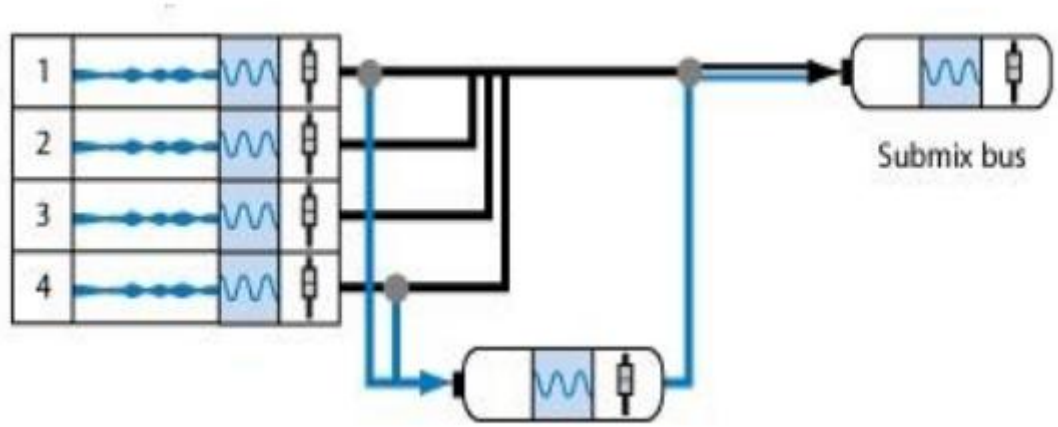
Σχήμα 2.2: Βασικό σήμα αντήχησης

Η δημιουργία ψηφιακής αντήχησης είναι πολύ πιο περίπλοκη από τον συνδυασμό μερικών ψηφιακών μονάδων αναμετάδοσης. Πολλές έρευνες έχουν δείξει ότι κάπου μεταξύ 1000 και 3000 ξεχωριστές ηχώ χρειάζονται κάθε δευτερόλεπτο για να δημιουργηθεί η ψευδαίσθηση της πυκνής, φυσικής ηχητικής αντήχησης. [12] και [13], η απόσταση μεταξύ αυτών των ανακλάσεων πρέπει να επιλέγεται πολύ προσεκτικά ή το αποτέλεσμα αντήχησης θα ηχεί με πολύ αφύσικο τρόπο.

Αντί να διασπάσει ένα σήμα σε μεμονωμένα δείγματα, η μέθοδος κωδικοποίησης ισοτιμίας διασπά ένα σήμα σε ξεχωριστές περιοχές δειγμάτων και κωδικοποιεί κάθε bit από το μήνυμα στο bit ισοτιμίας μιας περιοχής δείγματος [14].

Η απόκτηση των κατάλληλων αντανακλάσεων είναι μόνο ένα μέρος της ιστορίας. Χρειάζεται ψηφιακό φιλτράρισμα για να προσδώσουμε τα συγκεκριμένα χαρακτηριστικά συχνότητας στην αντήχηση του ήχου, και το λογισμικό πρέπει να είναι αρκετά εξελιγμένο ώστε να μπορεί να δημιουργήσει έναν αριθμό διαφορετικών προσομοιωμένων περιβαλλόντων χωρίς να είναι πολύ περίπλοκο για τους χρήστες να το ρυθμίσουν. Ωστόσο, ένα ψηφιακό σύστημα επιτρέπει στους χρήστες να δημιουργούν αντηχήσεις που είναι μεγαλύτερες και πιο φωτεινές από αυτές που υπάρχουν στην φύση, κάτι που συνήθως συναντάται στην πραγματική ηχογράφηση.

Για να πετύχουμε αυτά τα αποτελέσματα έχουν ανακαλυφθεί διάφορες τεχνικές χρήσης της αντήχησης από τους μηχανικούς του ήχου. Στο Σχήμα 2.3 βλέπουμε μια εξελιγμένη μορφή τους Σχήματος 2.2 με την τεχνική Bus. Δηλαδή ενώ στέλνονται τα σήματα (1,2,3,4) χωρίς το εφέ στην επόμενη έξοδο (Submix Bus), δημιουργούμε ένα παράλληλο κανάλι το οποίο θα έχει το εφέ της αντήχησης και θα καταλήγει και αυτό την ίδια έξοδο. Με αυτήν την τεχνική μπορούμε να επεξεργαστούμε την αντήχηση μεμονωμένα από τα άλλα σήματα και αναλόγως από το πόση ένταση θα στείλουμε από ένα σήμα στο κανάλι της αντήχησης, θα δημιουργούμε την ψευδαίσθηση ότι το σήμα έρχεται όλο και πιο μακρυά.



Σχήμα 2.3: Βασικό σήμα αντήχησης με την τεχνική Bus

Τα ψηφιακά αυτά μοντέλα είναι διαθέσιμα σήμερα και συνήθως διακρίνονται σε δύο τύπους [12]. Το ένα είναι η πλήρως προγραμματιζόμενη έκδοση και η λιγότερο ακριβή προκαθορισμένη με βάση το μηχάνημα. Μπορεί να είναι μόνο μία παράμετρος που μεταβάλλεται από τον χρήστη ανά προκαθορισμένη ρύθμιση. Από την άλλη πλευρά, η βασική τεχνική διασποράς του φάσματος έχει σχεδιαστεί για την κωδικοποίηση μιας ροής πληροφοριών με την εξάπλωση των κωδικοποιημένων δεδομένων σε όσο το δυνατόν μεγαλύτερο μέρος του φάσματος συχνοτήτων. Αυτό επιτρέπει τη λήψη σήματος, ακόμη και αν υπάρχουν παρεμβολές σε ορισμένες συχνότητες [14].

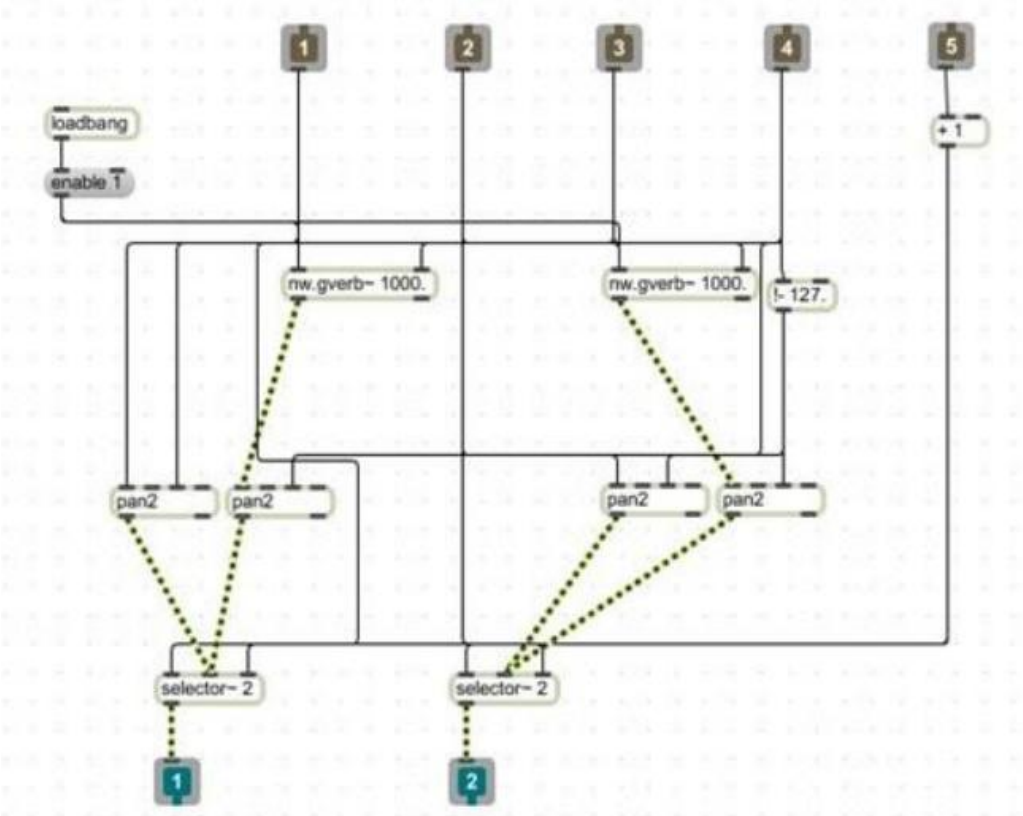
Η ραγδαία πρόοδος των τεχνολογιών των υπολογιστών συνέβαλε στην αύξηση της ταχύτητας του επεξεργαστή. Εμφανίστηκαν περισσότεροι χρήστες του Διαδικτύου, πρωτόκολλα και προγράμματα που απαιτούν μεγάλη απόδοση. Πάντα υπήρχε και θα συνεχίσει να υπάρχει κάποιου είδους

προσπάθεια για την ελαχιστοποίηση των απαιτούμενων επεξεργαστικών ικανοτήτων του υπολογιστή και της χρήσης του δικτύου, προκειμένου να εξοικονομηθεί κόστος [11].

Το ανθρώπινο ακουστικό σύστημα αντιλαμβάνεται σε ένα εύρος ισχύος μεγαλύτερο από ένα δισεκατομμύριο προς ένα και σε ένα εύρος συχνοτήτων μεγαλύτερο από χίλια προς ένα. Η ευαισθησία στον προσθετικό τυχαίο θόρυβο είναι επίσης έντονη. Οι διαταραχές σε ένα αρχείο ήχου μπορεί να είναι ανιχνεύσιμες σε επίπεδο χαμηλότερο από ένα μέρος στα δέκα εκατομμύρια (80 dB κάτω από το επίπεδο του περιβάλλοντος) [14]. Σε ένα πραγματικό ακουστικό περιβάλλον, η ομιλία συνήθως μολύνεται από παρεμβολές που μπορεί να είναι αρμονικές ή μη αρμονικές, στενής ζώνης (narrowband) ή ευρείας ζώνης (broadband).

Η παρεμβολή στις περισσότερες περιπτώσεις είναι ανεξάρτητη από την ομιλία, με την έννοια ότι το φασματικό περιεχόμενο συχνοτικά της ομιλίας και της παρεμβολής είναι ασυσχέτιστο. Για την ηχογραφημένη τραγουδιστική φωνή, ωστόσο, συνοδεύεται σχεδόν πάντα από μουσική όργανα που στις περισσότερες περιπτώσεις είναι αρμονικά, ευρυζωνικά και συσχετίζονται με το τραγούδι, δεδομένου ότι συντίθενται ώστε να αποτελούν ένα ενιαίο σύνολο με τη φωνή του τραγουδιού [7].

Εάν το ηχητικό περιεχόμενο έχει λιγότερα κανάλια διαθέσιμα από αυτά που μπορεί να παρέχει ένα ηχητικό σύστημα αναπαραγωγής, το ηχητικό σύστημα δεν μπορεί να το εκμεταλλευτεί πλήρως. Επομένως, προκειμένου να αξιοποιήσει τέτοιο ηχητικό περιεχόμενο, είναι απαραίτητη η χρήση μιας μεθόδου μίξης που μετατρέπει μονοφωνικές ή στερεοφωνικές μορφές ήχου σε πολυκάναλες μορφή ήχου, κατάλληλες για ένα τέτοιο ηχητικό σύστημα [15].



Σχήμα 2.4: Σύνθετος προγραμματισμός αντήχησης (MAX/MSP)

Στο σχήμα 2.4 περιγράφεται ένας πιο αναλυτικός προγραμματισμός των σημάτων και της επεξεργασίας τους με τελική έξοδο δύο καναλιών, για να πετύχουμε στερεοφωνικό αποτέλεσμα. Πριν την τελική έξοδο βλέπουμε πως τα σήματα και τα ξηρά (ξηρό λέγεται το σήμα που δεν έχει επηρεαστεί από κάποιο εφέ, συγκεκριμένα της αντήχησης) αλλά και αυτά του εφέ της αντήχησης καταλήγουν και στο κανάλι 1 και στο κανάλι 2 μέσω των ρυθμίσεων pan. Panning η διαδικασία τοποθέτησης ενός ήχου στο αριστερό ή δεξί κανάλι της στερεοφωνικής εικόνας.

Ωστόσο, η αντήχηση σε κάθε περίπτωση προέρχεται γενικά από μια μονοφωνική μίξη των εισόδων και καταλήγει στερεοφωνική [16]. Μόνο το ξηρό σήμα παραμένει στερεοφωνικό σε όλη τη διάρκεια. Αυτή η ιδέα είναι κατάλληλη επειδή η αντήχηση προέρχεται από όλες τις κατευθύνσεις ανεξάρτητα από τη στερεοφωνική τοποθέτηση από τον αρχικό ήχο και την αντήχηση στον φυσικό ήχο, ένα σύστημα μονοφωνικής εισόδου/στερεοφωνικής εξόδου είναι το μόνο που είναι απαραίτητο.

2.4 Διαθέσιμοι παράμετροι

Οι κύριες παράμετροι αντήχησης που είναι διαθέσιμες για έλεγχο από τον χρήστη είναι η προ-καθυστέρησης (pre-delay time), το μοτίβο πρώιμης ανάκλασης (early reflection), ο χρονική καθυστέρηση (time), ο συνολικός χρόνος απόσβεσης (decay time), η απορρόφηση της ψηλότερης συχνότητας (damping), μέγεθος του χώρου (size) και ο τύπος αντήχησης (reverb type). Στο σχήμα 2.6 φαίνεται το γραφικό περιβάλλον με τις διαθέσιμες παραμέτρους από μία επαγγελματική εφαρμογή ψηφιακής αντήχησης.

Σχήμα 2.5: Επαγγελματική εφαρμογή ψηφιακής αντήχησης Rverb από την εταιρία Waves

2.4.1 Προ-καθυστέρηση

Η προ-καθυστέρησης αναφέρεται στο χρονικό διάστημα που περνά μεταξύ της αρχής ενός ηχητικού σήματος και της εμφάνισης των πρώτων αντηχήσεων από το περιβάλλον. Όταν ένας ήχος εκπέμπεται σε έναν χώρο και ανακλάται από τους τοίχους ή άλλα αντικείμενα, η προ-καθυστέρηση καθορίζει το χρονικό διάστημα που περνά πριν από την εμφάνιση των πρώτων αντηχήσεων και μπορεί να είναι μεταβλητή από σχεδόν στιγμιαία έως μισό δευτερόλεπτο ή περισσότερο.

Ο έλεγχος της προ-καθυστέρησης σε ηχητικά εφέ μπορεί να επιτρέψει στον ηχολήπτη ή στον παραγωγό να διαμορφώσει τον χαρακτήρα του ήχου, δημιουργώντας μια πιο φυσική ακουστική αντίληψη, ανάλογα με τις απαιτήσεις της παραγωγής ή της ηχογράφησης.

2.4.2 Πρώιμη ανάκλαση

Οi πρώιμες ανάκλασης αναφέρονται στις πρώτες αντηχήσεις που προκαλούνται από επιφάνειες, από τους τοίχους, τα δάπεδα, την οροφή ή άλλα αντικείμενα στο περιβάλλον, πριν από την εμφάνιση των πιο πυκνών αντηχήσεων που χαρακτηρίζουν τον ήχο της αίθουσας ή του χώρου.

Τα πρώτα μοτίβα αντανάκλασης δεν είναι συνήθως μεταβλητά. Έτσι όπως ο χρήστης θα κάνει μια επιλογή από κάποια αποθηκευμένα μοτίβα (presets) που προσομοιώνουν διάφορα δωμάτια, αίθουσες, θαλάμους και πλάκες, ή μικρές ατμόσφαιρα δωματίου, έτσι και οι πρώιμες ανακλάσεις θα μεταβληθούν από αυτά. Όσο μεγαλύτερη είναι η απόσταση τόσο μεγαλύτεροι θα είναι και οι ήχοι του δωματίου.

2.4.3 Χρονική καθυστέρηση

Ένα στοιχείο χρονικής καθυστέρησης χρησιμοποιείται για την παροχή τέτοιων εφέ ατμόσφαιρας και η μετατόπιση φάσης ±90° απαιτείται για την παρουσίαση εφέ ευρυχωρίας. Υποθέτοντας ότι η απόσταση από τα μπροστινά ηχεία έως τα από τον τοίχο είναι περίπου 2 μέτρα και η απόσταση από τα πίσω μεγάφωνα από τον τοίχο είναι περίπου 1 μέτρο, εφαρμόζεται χρονική καθυστέρηση 12 ms στα κανάλια surround [7].

2.4.4 Συνολικός χρόνος απόσβεσης

Ο συνολικός χρόνος απόσβεσης καθορίζει απλώς πόσος χρόνος χρειάζεται για να εξαφανιστεί η αντήχηση, κατά ένα προκαθορισμένο ποσοστό (συνήθως 60 dB) από την αρχική του ένταση. Οι μεγαλύτεροι χρόνοι αντήχησης υποδηλώνουν μεγάλα περιβάλλοντα με πολύ ανακλαστικές επιφάνειες, ενώ οι μικρότεροι μπορούν να χρησιμοποιηθούν για την προσομοίωση της φυσικής ακουστικής ενός μικρού δωματίου.

Οι περισσότερες μονάδες αντήχησης μπορούν να παράγουν μεγάλο χρόνο απόσβεσης, αλλά η πραγματική δοκιμασία μιας αντήχησης είναι το πόσο πειστικά μιμείται την ατμόσφαιρα μικρού δωματίου [17].

2.4.5 Απορρόφηση υψηλής συχνότητας

Κατά την προσομοίωση του χώρου μέσω εφαρμογών αντήχησης, οι υψηλές συχνότητες μπορεί να ανακλώνται και να αποκρίνονται διαφορετικά από τις χαμηλότερες συχνότητες. Το High Frequency Damping επιτρέπει στον χρήστη να προσαρμόσει τον τρόπο με τον οποίο αυτές οι υψηλές συχνότητες θα ανταποκρίνονται κατά την προσομοίωση. Αυτό μπορεί να έχει επίπτωση στη φυσικότητα του ήχου,

καθώς ορισμένοι χώροι έχουν την τάση να απορροφούν υψηλές συχνότητες περισσότερο από χαμηλές.

2.4.6 Μέγεθος του χώρου

Η παράμετρος μεγέθους αναφέρεται στο μέγεθος του εικονικού χώρου που προσομοιώνεται από την αντήχηση. Αναφέρεται στο πόσο μεγάλος ή μικρός θα είναι ο επικαλυπτόμενος χώρος. Συνήθως, όταν αυξάνετε η παράμετρος του χώρου ακούγεται ένα πιο εκτεταμένο αποτέλεσμα, προσομοιώνοντας έναν μεγαλύτερο χώρο. Αντίστροφα, όταν μειώνετε η παράμετρος του χώρου ακούγεται μια πιο συμπυκνωμένη αντήχηση, προσομοιώνοντας έναν μικρότερο χώρο.

2.4.7 Τύπος αντήχησης

Κάθε είδος αίθουσας παρουσιάζει κάποια μοτίβα στον τρόπο με τον οποίο αναπαράγεται η αντήχηση. Έτσι, οι περισσότερες εταιρίες φτιάχνουν έτοιμα προφίλ με αυτά τα χαρακτηρίστηκα. Τα ποιο γνωστά από αυτά είναι: της αίθουσας (hall), του δωματίου (room), του θαλάμου (chamber), της εκκλησίας (church), της πλάκας (plate) (χώρος που χρησιμοποιεί μια μεγάλη μεταλλική πλάκα για να δημιουργήσει φωτεινό και ξεχωριστό ήχο). Επιλέγοντας το κατάλληλο μοτίβο για το περιβάλλον που θα προσομοιωθεί ένας χώρος και στη συνέχεια ρυθμίζοντας τις άλλες παραμέτρους. Τα διαθέσιμα εφέ μπορούν να ποικίλλουν από ένα δωμάτιο με ελάχιστο αντήχηση έως ένα τεράστιο σπήλαιο, στο οποίο η αποσύνθεση της αντήχησης βροντοχτυπά για αρκετές δεκάδες δευτερόλεπτα.

Προκειμένου να αξιοποιηθεί ένα τέτοιο ηχητικό περιεχόμενο, είναι απαραίτητη η χρήση μιας μεθόδου ανάμιξης που μετατρέπει μονοφωνικές ή στερεοφωνικές μορφές ήχου σε πολυκάναλη μορφή ήχου κατάλληλη για ένα τέτοιο σύστημα [15]. Στην πραγματικότητα, οι περισσότερες από τις χρήσιμες εφαρμογές της αντήχησης έχουν χρόνο απόσβεσης κάτω από δύο δευτερόλεπτα, επειδή η υπερβολικά μεγάλη διάρκεια της αντήχησης τείνει να θολώνει μια μίξη, αν και υπάρχουν περιστάσεις στις οποίες μια μακρά αντήχηση μπορεί να είναι αποτελεσματική ή να θεωρηθεί εσκεμμένη αισθητική για κάποιον καλλιτέχνη.

2.5 Επίλογος

Στην ενότητα αυτή, αναλύσαμε τα βασικά στοιχεία της αντήχησης, τόσο την φυσική αντήχηση όσο και την ψηφιακή. Είδαμε τις διαθέσιμες παραμέτρους όπως η προ-καθυστέρηση, η πρώιμη ανάκλαση, η χρονική καθυστέρηση, ο συνολικός χρόνος απόσβεσης, η απορρόφηση υψηλής συχνότητας, το μέγεθος του χώρου και τους βασικούς τύπους αντήχησης.

θα ήθελα να τονίσω πως η σημασία της ψηφιακής αντήχησής προσφέρει ευελιξία και εκτενείς δυνατότητες προσαρμογής, επιτρέποντας στους μηχανικούς τους ήχου και στους παραγωγούς να προσομοιώσουν διάφορα ακουστικά περιβάλλοντα, είτε να δημιουργήσουν ποικίλους μη ρεαλιστικούς ήχους.

Αφού κατανοήσαμε πως λειτουργεί η αντήχηση, στα επόμενα κεφάλαια θα δούμε την ανατομία των plugins και την δημιουργία τους. Και πως το εφέ της αντήχησης και τις παραπάνω παραμέτρους θα τις μεταφράσουμε σε κώδικα που θα πάρει γραφική υπόσταση για να μπορεί να χειραγωγηθεί από τον χρήστη.

Κεφάλαιο 3ο: Plugin

3.1 Εισαγωγή

Επί του παρόντος, υπάρχουν περισσότερα από δέκα πρότυπα για audio plugins, βασικά ένα ανά μεγάλο κατασκευαστή υποδοχής, όπως η Steinberg ή η Digidesign, αλλά η Microsoft και η Apple έχουν επίσης αναπτύξει API (Διεπαφές Προγραμματισμού Εφαρμογών) πολυμέσων ενσωματωμένα στο λειτουργικό τους σύστημα που περιλαμβάνουν audio plugins. Αυτά τα πρότυπα μοιράζονται πολλά χαρακτηριστικά και συμπεριφορές, ενώ παραμένουν ασύμβατα. Πολλοί μετατροπείς έχουν κυκλοφορήσει τα τελευταία χρόνια για να σπάσουν αυτές τις ασυμβατότητες, αλλά έχουν κάποια ανεπιθύμητα μειονεκτήματα: αυξάνουν τη φόρτιση της CPU και συχνά περιορίζονται σε ένα λειτουργικό σύστημα. Επιπλέον, η μεταφορά plugins από ένα λειτουργικό σύστημα σε ένα άλλο απαιτεί χρόνο, ειδικά όταν πρόκειται για γραφικά ή πρόσβαση στο σύστημα αρχείων. Πολλοί άνθρωποι έχουν προσπαθήσει για δική τους χρήση να αναπτύξουν εργαλεία για την αφαίρεση από τις πλατφόρμες και τα πρότυπα κατά την ανάπτυξη plugins, μειώνοντας έτσι το κόστος ανάπτυξης και διευκολύνοντας τη ζωή των προγραμματιστών. Δυστυχώς, ελάχιστα από αυτά τα δοκιμαστικά προγράμματα έχουν κυκλοφορήσει δημόσια.

Ο σκοπός του παρόντος εγγράφου, πέρα από τις μορφές των plugins, είναι να προσδιορίσει ποια είναι η ιδιαιτερότητα των audio plugins, τι είναι κοινό σε όλα τα πρότυπα και τι μπορεί να τα διαφοροποιείται. Δίνεται μια επισκόπηση του τι μπορεί να συναντήσει κανείς όταν προσπαθεί να στοχεύσει σε διαφορετικές πλατφόρμες και πρότυπα, και να αναζητήσει ένα κοινό μοντέλο που μπορεί να προκύψει. [18]

3.2 Τι είναι ένα audio plugin;

Ένα plugin έχει σκοπό να επεκτείνει ένα περιβάλλον δημιουργίας ήχου με τη χρήση third party βιβλιοθηκών. Ένα plugin έχει σκοπό να επεκτείνει ένα περιβάλλον δημιουργίας ήχου με τη χρήση ειδικών βιβλιοθηκών τρίτων. Έτσι, κάθε χρήστης θα πρέπει να είναι σε θέση να βρει plugins που ταιριάζει απόλυτα στις ανάγκες του για την προσαρμογή αυτού του περιβάλλοντος. Συνήθως τα plugins μπορούν να είναι θεωρηθούν ως ψηφιακές εφαρμογές με βάση τις συσκευές υλικού που υπάρχουν ήδη όπως reverbs, compressors, delays, synthetizers, samplers ή beatboxes που μπορούν να συνδεθούν μεταξύ τους με modular τρόπο. Αυτό επιτρέπει στους κατασκευαστές κεντρικών υπολογιστών να επικεντρωθούν σε την ευχρηστία και την αποδοτικότητα των προϊόντων τους, ενώ οι εξειδικευμένοι κατασκευαστές επικεντρώνονται στο κομμάτι της ψηφιακής επεξεργασίας σήματος.

Μπορούμε να διακρίνουμε 3 προσεγγίσεις για να κάνουμε plugins:

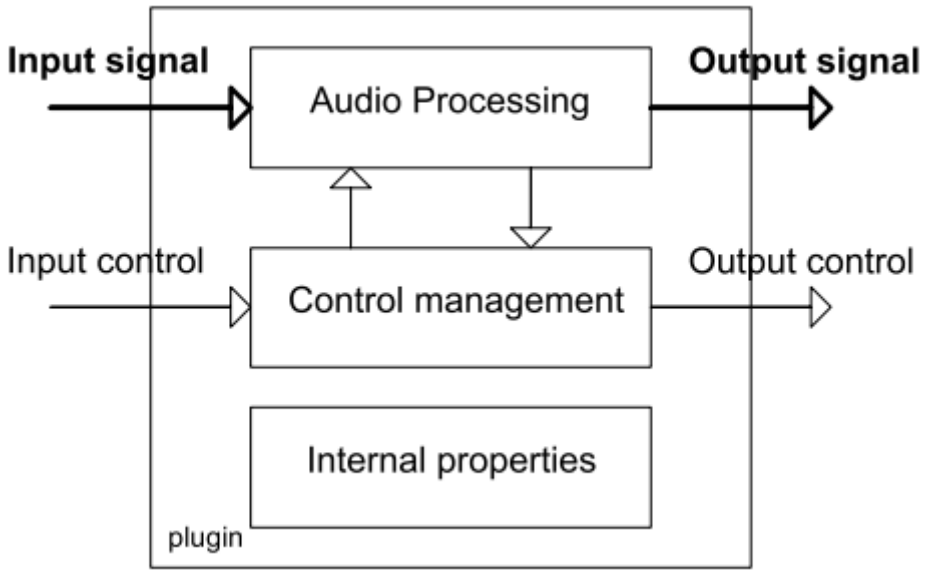
- Plugins ενσωματωμένα σε ένα API πολυμέσων, όπως το Audiounits της Apple και το DirectX της Microsoft.
- Plugins που σχετίζονται με έναν υποδοχέα αναφοράς όπως VST, RTAS, MAS. Τέτειες εφαρμογές βλέπουμε σε DAW (digital audio workstation) όπως το Cubase, το FL Studio, το Ableton κλπ. Μερικά από αυτά μπορούν να βρεθούν σε διαφορετικές πλατφόρμες όπως το VST ή το RTAS.
- Modules που συνδέονται με modular λογισμικά τις Max family (γνωστά τα Cycling 74's Max/Msp, Ircam's jMax an CRCA's Pure Data), Buzz ή EyesWeb.

Θα πρέπει επίσης να εξετάσουμε το LADSPA στο GNU-linux που είναι το de-facto πρότυπο για αυτή την πλατφόρμα, αλλά δεν μπορεί να ενταχθεί σε καμία από τις παραπάνω κατηγορίες.

3.2.1 Γενικό μοντέλο

Ένα plugin μεταχειρίζεται ένα ηχητικό σήμα και στη συνέχεια πρέπει να είναι ελεγχόμενο και προκειμένου να λειτουργήσει σε ένα περιβάλλον υποδοχής, ο υποδοχέας και το plugin πρέπει να γνωρίζουν την εκάστοτε ρύθμιση και τις ιδιότητές τους.

Η ροή ήχου αντιμετωπίζεται συνήθως με μπλοκ (buffers) μέσα στα οποία τα δείγματα είναι συγχρονισμένα με το ρυθμό δειγματοληψίας. Σε γενικές γραμμές, οι παράμετροι ελέγχου έχουν χαμηλότερο ρυθμό ενημέρωσης και μπορούν είτε να αντιμετωπίζονται μεταξύ των μπλοκ ήχου - κατά παραγγελία - εάν το τιμή έχει αλλάξει, είτε να μπαίνουν σε ουρά με χρονοσημάνσεις (timestamps) σε σχέση με μια θέση σε ένα μπλοκ για ανάλυση του ρυθμού του ήχου. Οι ιδιότητες είναι ως επί το πλείστον στατικές σημαίες, αλλά ορισμένες μπορεί να είναι και δυναμικές.



Σχήμα 3.1: Ένα κοινό μοντέλο αρχιτεκτονικής μουσικών plugins

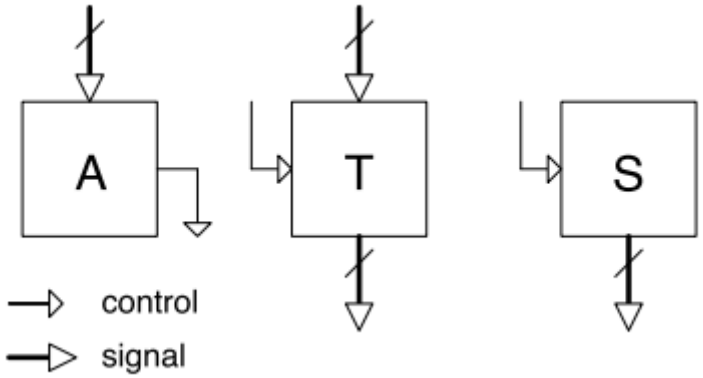
Τα plugins πρέπει να γνωρίζουν τις δυνατότητες του κεντρικού υπολογιστή, όπως η αποστολή/λήψη συμβάντων, η παροχή διαύλων πολυκάναλων κλπ. Επιπλέον, υπάρχουν πολλές περιπτώσεις όπου θα εκτιμούσαμε ότι τα plugins μπορούν να γνωρίζουν περισσότερα για το μουσικό πλαίσιο (π.χ. το τέμπο, την θέση της παρτιτούρας, το μέτρο κλπ.) από απλά δεδομένα ήχου και δεδομένα ελέγχου.

Το plugin πρέπει να δίνει πληροφορίες σχετικά με τις εισόδους και εξόδους του, τις ιδιαιτερότητές τους, όπως αν χρησιμοποιεί τεχνική side-chain, αν είναι μονοφωνικό, στερεοφωνικό ή surround και πολλά άλλα πράγματα, όπως η καθυστέρησή του (Ένας αλγόριθμος όπως ο μετασχηματισμός Fourrier μπορεί να εισάγει μια καθαρή καθυστέρηση στην αλυσίδα σήματος επειδή χρειάζεται έναν ελάχιστο αριθμό δειγμάτων για να ξεκινήσει. Σε ένα sequencer - όχι σε πραγματικό χρόνο! - αυτή η καθαρή καθυστέρηση μπορεί να αντισταθμιστεί από τον κεντρικό υπολογιστή στέλνοντας δεδομένα στο plugin εκ των προτέρων) ή ο χρόνος της ουράς του. Βασικά όλοι οι αλγόριθμοι που βασίζονται στη συνέλιξη μπορούν να εισάγουν μια ουρά, δηλαδή δείγματα εξόδου ακόμη και όταν δεν υπάρχουν άλλα δεδομένα εισόδου (π.χ. ένα delay, ένα reverb ένα FIR φίλτρο). Υπάρχουν επίσης κάποιοι hosts

που επιτρέπουν στα plugins να τους δίνουν εντολές όπως play, stop ή set tempo αλλά δεν είναι πραγματικά ευρύτατα διαδεδομένο.

3.2.2 Είσοδοι - Έξοδοι

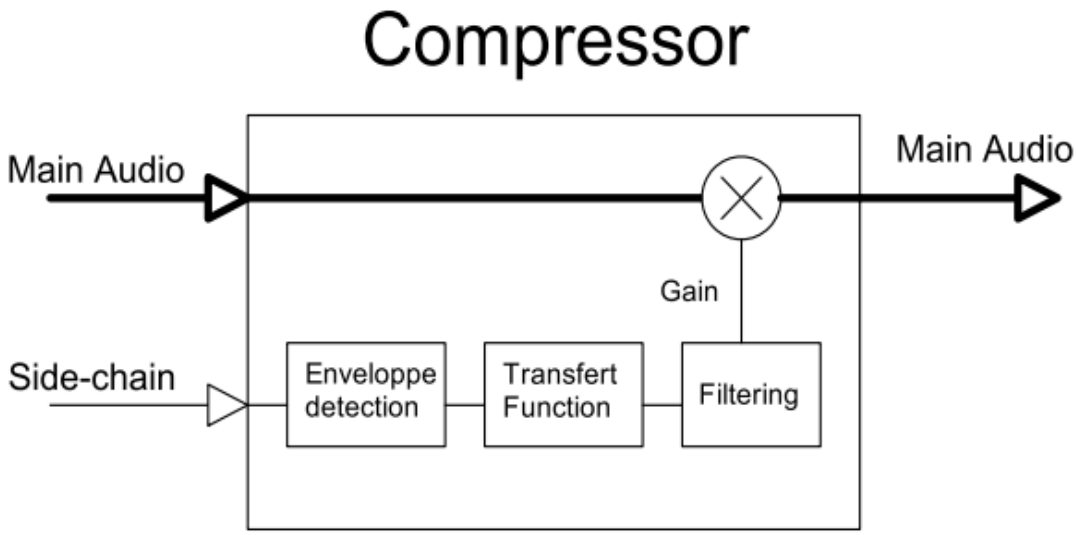
Για να κατανοήσουμε τι είναι σχετικό με τον καθορισμό του αριθμού και του είδους των Εισόδου/Εξόδου, θα αναλύσουμε τι κάνουν οι αλγόριθμοι των plugins γύρω από 3 διαφορετικές λειτουργικότητες - ανάλυση (A), μετασχηματισμός (T) και σύνθεση (S) - οι οποίες μπορούν να είναι είτε μόνοι τους ή να αναμειγνύονται μαζί στην ίδια μονάδα. Στο σχήμα 3.2 δείχνουμε πως σημειώνονται αυτές τις λειτουργίες.



Σχήμα 3.2: Λειτουργίες ανάλυσης (A), μετασχηματισμού (T) και σύνθεσης (S)

Επιπλέον, οι ρυθμίσεις των plugins εξαρτώνται σε μεγάλο βαθμό από το τι επιτρέπει ο κεντρικός υπολογιστής και πώς το plugin θα συνδεθεί με το περιβάλλον του. Ενώ οι περισσότεροι υποδοχείς υποστηρίζουν μόνο μονοφωνία και στερεοφωνία - ακόμη και αν το surround αρχίζει να γίνεται δημοφιλές και τη μορφή 5.1 surround (αριστερά, κέντρο, δεξιά, πίσω αριστερά, πίσω δεξιά συν subwoofer) - μπορούμε επίσης να διακρίνουμε μεταξύ εφέ που θα αντιμετωπίζονται ως inserts (ενσωματωμένα σε μια αλυσίδα σε ένα μόνο bus), ως sends (μπορούν να μοιραστούν από πολλές φωνές σε μια κονσόλα μίξης και η έξοδος αθροίζεται στο master bus), ως όργανα (χωρίς είσοδο ήχου) ή απλά ενσωματωμένα σε ένα modular υποδοχέα.

Ο αριθμός των εισόδων και εξόδων ήχου ποικίλλει από το ένα plugin στο άλλο, μπορούν να ομαδοποιηθούν σε busses που μπορεί να είναι μονοφωνικά, στερεοφωνικά, surround. . αλλά εξαρτώνται κυρίως από τις δυνατότητες του κεντρικού υπολογιστή. Επιπλέον, σε ορισμένα πρότυπα - ο αριθμός τους εξακολουθεί να αυξάνεται - είναι δυνατό να οριστεί side-chain. Ένα κανάλι side-chain μπορεί να αναλυθεί ή να χρησιμοποιηθεί απευθείας για την τροποποίηση της κύριας ροής επεξεργασίας του ήχου. Για παράδειγμα βλέπε στο Σχήμα 2.3 την ανάλυση ενός plugin εφέ compressor.



Σχήμα 3.3: Ένα παράδειγμα side-chain: ένας compressor

Αν και είναι δυνατό να δημιουργήσετε ένα plugin που δεν έχει τρόπο να ελέγχεται, όπως ένας μετατροπέας φάσης, έχει περιορισμένο ενδιαφέρον σε ένα μουσικό πλαίσιο όπου η ρύθμιση των παραμέτρων ή η αποστολή των νοτών είναι μέρος της δημιουργικής διαδικασίας. Μπορεί να θέλετε τόσο να αλλάξετε τις παραμέτρους του plugin όσο και να ηχογραφήσετε την μεταβολή των παραμέτρων για μεταγενέστερη αναπαραγωγή ή επεξεργασία. Ως εκ τούτου, τα περισσότερα plugins θα πρέπει να παρέχουν είσοδο και έξοδο ελέγχου ταυτόχρονα. Τις περισσότερες φορές οι hosts και τα plugins ειδοποιούν το ένα το άλλο για την εξέλιξη των παραμέτρων με απλό τρόπο, αλλά μπορεί κανείς να βρει και μηχανισμούς callback και polling. Επιπλέον, η αποστολή δεδομένων ελέγχου μεταξύ των plugins μπορεί μερικές φορές να πραγματοποιηθεί είτε απευθείας μέσω παραμέτρων (LADSPA) είτε μέσω ενός πρωτοκόλλου προσανατολισμένου σε συμβάντα όπως το MIDI (VST).

3.2.3 Η επεξεργασία

Όλα τα plugins έχουν αυτή τη λειτουργία που συχνά ονομάζεται επεξεργασία "process". Αυτή είναι η θέση στην οποία συμβαίνει το πιο σημαντικό πράγμα: η επεξεργασία ήχου. Όλα τα τρέχοντα πρότυπα επεξεργάζονται τον ήχο ως buffers εισόδου και εξόδου (συγκεκριμένου μήκους), που μπορούν να δοθούν ως πίνακες δειγμάτων κινητής υποδιαστολής, ή από πιο εξελιγμένες δομές με επιπρόσθετες πληροφορίες. Η επεξεργασία μπορεί να είναι in-place, οπότε οι buffers εισόδου και εξόδου είναι οι ίδιοι (δηλ. μοιράζονται την ίδια θέση μνήμης, η επεξεργασία είναι επομένως καταστροφική), ή buffer-to-buffer, όπου είναι διαφορετικοί. Οι παράμετροι είναι συνήθως γνωστές πριν από την έναρξη της διαδικασίας, αλλά μπορεί να χρειαστεί να τις ενημερώσετε με το χέρι. Για απλή επεξεργασία ήχου, όπως το biquadratic filtering, θα πρέπει να είναι αρκετό να χρησιμοποιήσετε απευθείας τον παρεχόμενο buffer, αλλά όταν κάνετε επεξεργασία FFT, για παράδειγμα, πρέπει να μπορεί να θέλετε να κάνετε εκ νέου buffering των δειγμάτων ήχου για να χωρέσουν στο εσωτερικό μέγεθος του buffer σας, οπότε μπορεί να εισαγάγετε πρόσθετη καθυστέρηση στην αλυσίδα σήματος.

3.2.4 Η διεπαφή χρήστη

Το γραφικό περιβάλλον θα πρέπει να θεωρείται τουλάχιστον εξίσου σημαντικό με τον αλγόριθμο επεξεργασίας ήχου, επειδή είναι το φιλτράρισμα μέσω του οποίου θα γίνει αντιληπτό το plugin σας. Είναι ένα σημείο-κλειδί για την ευχρηστία ενός plugin, όλα πρέπει να είναι γρήγορο και εύκολα ρυθμιζόμενο με ακριβή τρόπο και με βολική οπτική ανάδραση. Η αντιστοίχιση των παραμέτρων μπορεί πραγματικά να κάνει τη διαφορά στον τρόπο με τον οποίο οι άνθρωποι νιώθουν τον τρόπο με τον οποίο ακούγεται το plugin σας.

Το GUI μπορεί είτε να δημιουργηθεί αυτόματα από τον κεντρικό υπολογιστή με τη βοήθεια του τις πληροφορίες ή τις υποδείξεις που μπορεί να παρέχει το plugin για τον εαυτό του (τους τύπους των παραμέτρων του, τον τύπο του ελέγχου που πρέπει να χρησιμοποιηθεί – κουμπί ολισθητήρας (slider), combo-box, διακόπτης, κουμπί, ελεγκτής 2D κλπ.) ή παρέχεται από τον κατασκευαστή για να επιτευχθεί ένα υψηλότερο επίπεδο προσαρμογής.

3.2.5 Ανάπτυξη plugin

Το API ορίζει το επίπεδο μέσω του οποίου τα plugins και οι hosts μπορούν να βλέπουν το ένα το άλλο, καθώς και τα εργαλεία που μπορούν να χρησιμοποιηθούν για να διευκολύνουν το βήμα της ανάπτυξης. Ο Host χρειάζεται διαφορετικές πληροφορίες από τον χρήστη σχετικά με τα plugins, όπως το ID τους, τις δυνατότητές τους, την εγκατάστασή τους, τον αριθμό των παραμέτρων ή τα ονόματά τους. Ο κεντρικός υπολογιστής πρέπει να λαμβάνει δεδομένα που μπορούν να διαβαστούν από τον υπολογιστή με τον τρόπο που έχει προγραμματιστεί. Επιπλέον, τα περισσότερα από τα SDKs (Software Developpement Kits) παρέχουν ένα σύνολο εργαλείων, λειτουργιών, βοηθητικών προγραμμάτων. Στην ίδια προσπάθεια, συχνά επιτρέπουν τη χρήση OOP (αντικειμενοστραφής προγραμματισμός) γλωσσών όπως η C++, η ObjectiveC ή η Pascal για να διευκολυνθεί το στάδιο της μοντελοποίησης και η κατανομή των εργασιών στο εσωτερικό του ίδιου του προγράμματος.

3.3 Η ροή του σήματος

3.3.1 Η φύση της ροής του σήματος

Τις περισσότερες φορές, η ροή σήματος αναπαρίσταται ως δείγματα κινητής υποδιαστολής στο διάστημα [-1; 1] σε πλήρη κλίμακα. Ωστόσο, ορισμένα πρότυπα υποστηρίζουν δείγματα σταθερής υποδιαστολής για ιστορικούς λόγους, όπως για παράδειγμα, η μορφή CD που χρησιμοποιεί δείγματα ακέραιου προσημασμένου αριθμού 16 bit επιτρέπει την κωδικοποίηση του σήματος στο διάστημα [2^{15} - 1; -2^{15}] νώ το DVD χρησιμοποιεί 24-bit: εύρος [2^{23} - 1; -2^{23}], ή για τεχνικούς λόγους, όπως τα DirectX, AudioUnits και TDM που υποστηρίζουν δείγματα σταθερού σημείου επειδή τα plugins τους μπορούν να ασχοληθούν απευθείας με αρχεία, cd-rom, dvd-rom, κάρτες ήχου ή DSP με σταθερό σημείο, ενώ άλλα πρότυπα ασχολούνται μόνο με hosts για τους οποίους η αναπαράσταση κινητής υποδιαστολής είναι πιο βολική.

Για λόγους αποδοτικότητας, τα δείγματα αυτά συγκεντρώνονται σε buffers των οποίων το μέγεθος μπορεί να είναι σταθερό ή μεταβλητό από τη μία κλήση στην άλλη. Αν και η επεξεργασία ανά δείγμα (επιτρέπει την ανατροφοδότηση και την αναδρομή μεταξύ των αντικειμένων και είναι πολύ χρήσιμη για τη "φυσική μοντελοποίηση") χρησιμοποιείται ήδη σε ορισμένες βιβλιοθήκες DSP, δεν υπάρχει ήδη κάποιο πρότυπο που να κάνει την επεξεργασία με αυτόν τον τρόπο. Η συχνότητα δειγματοληψίας υποτίθεται ότι είναι σταθερή (και γνωστή) τουλάχιστον κατά τη διάρκεια του μήκους του buffer και

μπορεί να αλλάζει με το ρυθμό του buffer, αν και αυτό είναι ασυνήθιστο. Συνήθως χρησιμοποιούνται τιμές όπως 44,1kHz για το CD, 48kHz για το DAT ή το βίντεο και 96kHz για το DVD.

Οι buffers ήχου μπορούν είτε να περιβάλλονται από πρόσθετες πληροφορίες όπως χρονοσημάνσεις, όπου μπορούν να χρησιμοποιηθούν για σκοπούς συγχρονισμού από τον χρονοπρογραμματιστή του κεντρικού υπολογιστή όταν υπάρχουν πολλά σύνθετα μονοπάτια ήχου (DXi, EyesWeb, AU), samplerate, buffer-size κλπ., ή να μεταδίδονται (τις περισσότερες φορές) ως απλοί πίνακες.

Κάποια plugin API - DX, AU - επιτρέπουν τη διασύνδεση καναλιών για συμβατότητα με τύπους αρχείων και πρωτόκολλο ροής, αλλά είναι αρκετά περιθωριακό και τείνει να εξαφανιστεί τουλάχιστον για επεξεργασία σε πραγματικό χρόνο. Ο πιο συνηθισμένος (και ευκολότερος) τρόπος είναι να να μεταδίδονται τα κανάλια ως διαχωρισμένα μονοφωνικά buffers.

3.3.2 Η επεξεργασία του σήματος

Εδώ ερχόμαστε στο πιο σημαντικό μέρος ενός plugin, σχεδόν κάθε πρότυπο plugin έχει την ίδια μέθοδο που ονομάζεται process() για να κάνει την επεξεργασία ήχου, μόνο ο τρόπος που καλείται και τα ορίσματα είναι διαφορετικά μεταξύ των διαφόρων προτύπων. Αυτή η μέθοδος καλείται είτε απευθείας από τον κεντρικό υπολογιστή είτε από το επόμενο plugin στην ιεραρχία στην περίπτωση των γραφικών κεντρικών υπολογιστών. Οι buffers εισόδου και εξόδου παρέχονται, τις περισσότερες φορές, από τον host και μπορεί να είναι είτε ίδιοι είτε διαφορετικοί ώστε να επιτρέπουν την in-place ή bufferto-buffer επεξεργασία αλλά το plugin δεν μπορεί να υποθέσει το ένα ή το άλλο και θα πρέπει να λειτουργεί σωστά και στις δύο περιπτώσεις ή να διευκρινίζει αν το υποστηρίζει (LADSPA).

Ως βασική διαφορά με τους ψηφιακούς επεξεργαστές σήματος υλικού, η μέθοδος αυτή υποτίθεται ότι δεν είναι διακοπτόμενη. Επομένως, η παρεμβολή παραμέτρων (εάν απαιτείται) ή άλλη επεξεργασία με την ακρίβεια δείγματος πρέπει να γίνει μέσα στη διαδικασία και δεν μπορεί να να γίνει αυτόματα,δεδομένου ότι η επεξεργασία του ήχου από τους buffers εμποδίζεται από τον έλεγχο του ρυθμού του ήχου, μόλις το μέγεθος του buffer υπερβεί το 1 δείγμα.

3.3.3 Γενικές εκτιμήσεις για το ψηφιακό σήμα επεξεργασίας

Αυτή η ενότητα δεν σκοπεύει να εξηγήσει τους κανόνες της "βελτιστοποίησης αλγορίθμων επεξεργασίας ήχου". Θέλουμε απλώς να επισημανθεί ρητά ότι, δεδομένου ότι οι ρυθμοί ήχου μετρώνται σε δέκατα του kHz και ότι τα plugins μπορεί να χειρίζονται πολλά κανάλια, ο αριθμός των δειγμάτων προς επεξεργασία μπορεί να γίνει τεράστιος, επομένως θα πρέπει να δοθεί ιδιαίτερη προσοχή στην πολυπλοκότητα του αλγορίθμου DSP. Ειδικότερα οι αλγόριθμοι των οποίων η πολυπλοκότητα είναι εκθετική ή πολυωνυμική με τον αριθμό των δειγμάτων προς επεξεργασία, δεν θα πρέπει να χρησιμοποιούνται σε πραγματικό χρόνο.

Χωρίς να αναφερθούμε στην κωδικοποίηση με assembler, θα πρέπει να αποφεύγονται οι κατανομές μνήμης, οι εκφράσεις υπό συνθήκη μέσα σε βρόχους (μπορεί να σπάσουν τις βελτιστοποιήσεις pipe-lining), οι πάρα πολλές κλήσεις μη-στατικών-γραμμών-λειτουργιών ή η εντατική μετατροπή float-int, μεταξύ άλλων γενικών συμβουλών προγραμματισμού. Σημειώστε ότι σε πολλές περιπτώσεις η μείωση της δειγματοληψίας κατά τη διάρκεια του βήματος ανάλυσης μπορεί να εξοικονομήσει πολύτιμο χρόνο CPU για άλλο σκοπό.

3.4 Έλεγχος

Η διεπαφή ελέγχου επιτρέπει στο χρήστη να ενεργεί στις παραμέτρους, να ρυθμίζει τη συμπεριφορά της λειτουργίας της επεξεργασίας του ήχου. Αυτό μπορεί να γίνει σε πραγματικό χρόνο, έτσι ώστε ο χρήστης να μπορεί να ακούσει άμεσα τις συνέπειες των αλλαγών του στο μετασχηματισμό του ηχητικού σήματος. Υπάρχουν διάφορα είδη παραμέτρων, που απαιτούν διαφορετικό χειρισμό. Μπορούμε να θεωρήσουμε τρεις κύριες κατηγορίες παραμέτρων:

- **Συνεχείς παράμετροι**: Αυτές οι παράμετροι μπορούν να χαρακτηριστούν από το γεγονός ότι δεν εισάγουν διακοπές στη ροή ήχου όταν αλλάζουν σε μικρό βαθμό. Η διακοπή στη ροή ήχου χαρακτηρίζεται από ακουστικό θόρυβο όπως π.χ. τα "κλικς"). Κατά συνέπεια, αυτοί οι παράμετροι μπορούν να παρεμβάλλονται. Οι παράμετροι αυτοί αντιπροσωπεύονται προφανώς με αριθμητικές τιμές και μπορούν να θεωρηθούν "παθητικές", με την έννοια ότι απλώς χρησιμοποιούνται ως μεταβλητή τιμή σε έναν γενικό υπολογισμό. Ως παραδείγματα: η συχνότητα αποκοπής ενός φίλτρου, μία ενίσχυση έντασης ήχου, μία διαφοροποιημένη συχνότητα ή ένα κατώφλι αποκοπής είναι συνεχή.

- **Γεγονότα**: Αυτές οι παράμετροι, που μερικές φορές αναφέρονται ως μηνύματα ή εντολές (EyesWeb), χαρακτηρίζονται από τη διακριτή φύση τους. Τα μηνύματα αυτά μπορούν να είναι οποιουδήποτε τύπου, αλλά θα πρέπει να ανήκουν σε ένα σύνολο προκαθορισμένων μηνυμάτων, γνωστά από το plugin. Δεν χρησιμοποιούνται άμεσα στον υπολογισμό, αλλά μάλλον ερμηνεύονται για την ενεργοποίηση μιας συγκεκριμένης ενέργειας, χωρίς να διακόπτεται η διαδικασία ροής ήχου. Ως παραδείγματα: η επιλογή μεταξύ 'sawtooth', 'sine' και 'square' για ένα modulator σήμα, ένα MIDI note-on γεγονός ή ο αριθμός των ηχώ σε ένα εφέ καθυστέρησης.

- **Παράμετροι ρύθμισης**: Αυτές οι παράμετροι επηρεάζουν τη διαμόρφωση του plugin. Είναι μη πραγματικού χρόνου επειδή περιλαμβάνουν υπολογιστικά δαπανηρές λειτουργίες, όπως η κατανομή μνήμης, που σίγουρα διακόπτουν τη λειτουργία του ηχητικού σήματος. Ως παραδείγματα: η επιλογή ενός αρχείου παλμικής απόκρισης για ένα plugin συνέλιξης ή ένα μέγεθος fft σε ένα μετασχηματισμό στο πεδίο της συχνότητας.

Αυτή η διάκριση μεταξύ αυτών των τριών κατηγοριών δεν είναι πάντα τόσο σαφής και καλά καθορισμένη, αλλά αυτά τα τρία είδη παραμέτρων αντιστοιχούν σε διαφορετικούς τρόπους χειρισμού.

Με την αύξηση της πολυπλοκότητας των plugins, τα οποία απαιτούν όλο και περισσότερες εσωτερικές παραμέτρους για την επίτευξη λεπτότερων ακουστικών αποτελεσμάτων, υπάρχει συνεχής ανάγκη για διαδικασίες αντιστοίχισης και μετατροπής, έτσι ώστε ο χρήστης να μπορεί να τα ελέγχει με εργονομικό τρόπο. Οι παράμετροι που είναι διαθέσιμες στον χρήστη και τον κεντρικό υπολογιστή (εξωτερικές παράμετροι) μπορεί να διαφέρουν από τις παραμέτρους που χρησιμοποιούνται στον υπολογισμό της διαδικασίας (εσωτερικές παράμετροι). Μπορεί να υπάρχει ένα επίπεδο μετατροπής μεταξύ τους. Αυτή η μετατροπή μπορεί να είναι διαφόρων ειδών, όπως η κλιμάκωση ενός συντελεστή έντασης που εκφράζεται σε dB σε γραμμική κλίμακα, η αντιστοίχιση της συχνότητας, του συντελεστή έντασης και του συντονισμού σε συντελεστές φίλτρου ή η αποκοπή της παραμέτρου στο εύρος της.

3.4.1 Δήλωση εξωτερικής παραμέτρου

3.4.1.1 Υποχρεωτικές δηλώσεις

Οι εξωτερικές παράμετροι θα πρέπει να δηλώνονται, ώστε ο κεντρικός υπολογιστής ή/και ο χρήστης να μπορεί να τις τροποποιήσει. Ωστόσο, ο κεντρικός υπολογιστής και ο χρήστης δεν χρειάζονται τις ίδιες πληροφορίες για να τις χειριστούν. Το ελάχιστο απαιτούμενο σύνολο παραμέτρων ιδιοτήτων που ο host πρέπει να γνωρίζει, τουλάχιστον, είναι οι δύο ακόλουθες:

- **Η ταυτότητα της παραμέτρου**: Αυτό το ID χρησιμοποιείται ως δείκτης επιλογής μεταξύ της δομής της παραμέτρου. Τα ID παραμέτρων είναι προσημασμένοι ή μη προσημασμένοι μεγάλοι ακέραιοι αριθμοί σε όλα τα μεγάλα πρότυπα plugin ήχου, μερικές φορές απαριθμούμενου τύπου (DXi, VST, RTAS).

- **Ο τύπος**: Το LADSPA και το VST χρησιμοποιούν μόνο 32-bit float για όλο τον έλεγχο των δεδομένων, και το VST περιορίζει το εύρος στο [0,1]. Άλλα πρότυπα όπως τα DXi και AudioUnits υλοποιούν όλες τις παραμέτρους ως 32-bit float για λόγους αποδοτικότητας, αλλά διατηρούν ένα πεδίο σε μια δομή πληροφοριών των παραμέτρων, το οποίο ορίζει αν θα ερμηνεύσει μία τιμή ως ακέραιος, ως τιμή κινητής υποδιαστολής, boolean ή απαρίθμηση (ακέραιος series). Βρίσκουμε έναν παρόμοιο μηχανισμό στο EyesWeb, αλλά με τρεις βασικές

μορφές: double, int, και συμβολοσειρά χαρακτήρων, καθώς και μια σημαία "Τύπος παραμέτρου".

3.4.1.2 Προαιρετικές δηλώσεις

Τα περισσότερα plugin πρότυπα προσφέρουν πολύ περισσότερες πληροφορίες σχετικά με τις παραμέτρους, για να αποτρέψουν τον host και τον χρήστη από το να τις χειρίζονται λανθασμένα και για να επιτρέπουν στον χρήστη να έχει πρόσβαση σε αυτές με πιο φιλικό τρόπο. Οι πληροφορίες αυτές μπορεί να αποθηκεύονται σε μια δομή "ParameterInfo" (DXi, AU) ή να προτείνονται ως καθορισμένες σημαίες ή "υποδείξεις" (LADSPA).

- **Η ονομασία ή ετικέτα** είναι ενδιαφέρουσα για τον χρήστη που δεν μιλάει σαν τις μηχανές. Το όνομα της παραμέτρου μπορεί να είναι μέρος μιας δομής πληροφοριών παραμέτρου, όπως στα DXi, EyesWeb και AU. Το LADSPA αποθηκεύει τα ονόματα των θυρών (ήχου και χειρισμού) σε έναν πίνακα. Το όνομα της παραμέτρου δεν είναι υποχρεωτικό στο VST, αλλά υπάρχει μια μέθοδος 'GetParameterName' στο API, την οποία ο προγραμματιστής μπορεί να υλοποιήσει.

- **Οι μονάδες** δίνουν περισσότερο νόημα στις τιμές. Λίγα πρότυπα προσφέρουν αυτό το χαρακτηριστικό: το DXi και το EyesWeb αποθηκεύουν τη μονάδα της κάθε παραμέτρου στη δομή ParamInfo. Στο AU, οι μονάδες είναι μέρος του τύπου: για παράδειγμα, ο τύπος 'γωνία' εκφράζεται σε μοίρες ή ακτίνια, ο τύπος 'συχνότητα' σε Hertz κλπ.

- **Το εύρος**: Δίνει υποδείξεις σχετικά με τις τιμές που θεωρείται ότι μπορεί να πάρει αυτή η παράμετρος. Αυτό είναι χρήσιμο για να αποτρέψετε το plugin από το να κάνει υπολογισμούς που οδηγούν σε σφάλματα, όπως log(0), αρνητική συχνότητα για ένα φίλτρο κλπ. Βοηθά επίσης τον χρήστη να κατανοήσει το νόημα της παραμέτρου που χειρίζεται. Σχεδόν όλα τα πρότυπα επιτρέπουν τον καθορισμό του εύρους των παραμέτρων, εκτός από το VST για το το οποίο το εύρος είναι κανονικοποιημένο μεταξύ [0,1] και η μετατροπή πρέπει να γίνει με το χέρι.

- **Μια προεπιλεγμένη τιμή**: Μπορεί να αρχικοποιήσει την παράμετρο σε μια σχετική τιμή, εντός του εύρους των τιμών της παραμέτρου, κατά τη στιγμή της αρχικοποίησης. Αυτό το χαρακτηριστικό υλοποιείται στα LADSPA, DXi, EyesWeb AU, MAS. Στο VST, αποθηκεύεται στην προεπιλεγμένη επιλογή.

- **Η χαρτογράφηση**: Βοηθά στο χειρισμό της παραμέτρου με πιο εργονομικό τρόπο. Για παράδειγμα, η συχνότητα και η ενίσχυση είναι συχνά πιο βολικές όταν χειρίζονται με λογαριθμική χαρτογράφηση. Η αντιστοίχιση μπορεί να είναι γραμμική (όλα τα πρότυπα), λογαριθμική (LADSPA, AU), boolean (LADSPA, DXi, AU), indexed (LADSPA, DXi, AU) κλπ. Το VST παρέχει εργαλεία μετατροπής μόνο για GUI-εμφάνιση.

- **Αυτοματισμοί** (βλ. κεφάλαιο 3.4.2.2.)**:** Οι αυτοματοποιημένες παράμετροι πρέπει να δηλώνονται **ανάλογα** με το κατάλληλο μέσο:

- AudioUnits: Σημαία στη δομή πληροφοριών παραμέτρου.
- DXi: Ο δείκτης αυτοματοποιημένων παραμέτρων πρέπει να είναι μικρότερος από NUM_AUTOMATED_PARAMS εντός της απαρίθμησης **των** παραμέτρων.

- MAS: Οι αυτοματοποιημένες παράμετροι πρέπει να είναι δημόσιες.
- VST: Κατά τη διάρκεια του χρόνου εκτέλεσης, θα πρέπει να χρησιμοποιείται η συγκεκριμένη μέθοδος setParameterAutomated.

	Ονομασία/Ετικέτα	Μονάδες	Εύρος	Προεπιλεγμένη τιμή	Χαρτογράφηση
VST	Δεν είναι υποχρεωτική, αλλά μέθοδος στο API	-	[0,1] σταθερό	-	εργαλεία μετατροπής μόνο για προβολή
AU	Σε δομή ParamInfo	Σε δομή ParamInfo	Min, max αποθηκεύονται σε μια δομή ParamInfo	Σε δομή ParamInfo	Πολλά πολλά!
LADSPA	Στον πίνακα PortNames	-	Min και max προτείνονται ως υποδείξεις	προτείνεται ως υπόδειξη, σχετική	lin, log, int, bool, SR-multiple
MAS	???	???	???	???	Όχι
DXi	Σε δομή ParamInfo	Σε δομή ParamInfo	Min, max αποθηκεύονται σε μια δομή ParamInfo	Σε δομή ParamInfo	-
EyesWeb	Σε δομή ParamInfo	Σε δομή ParamInfo	Σημαίες HAS MINMAX και τιμές	Ναι	-
RTAS	???	???	???	???	???

Πίνακας 3.1: Διαθέσιμες πληροφορίες παραμέτρων

Πολλές άλλες μετα-πληροφορίες μπορούν να βρεθούν σε ορισμένα πρότυπα:

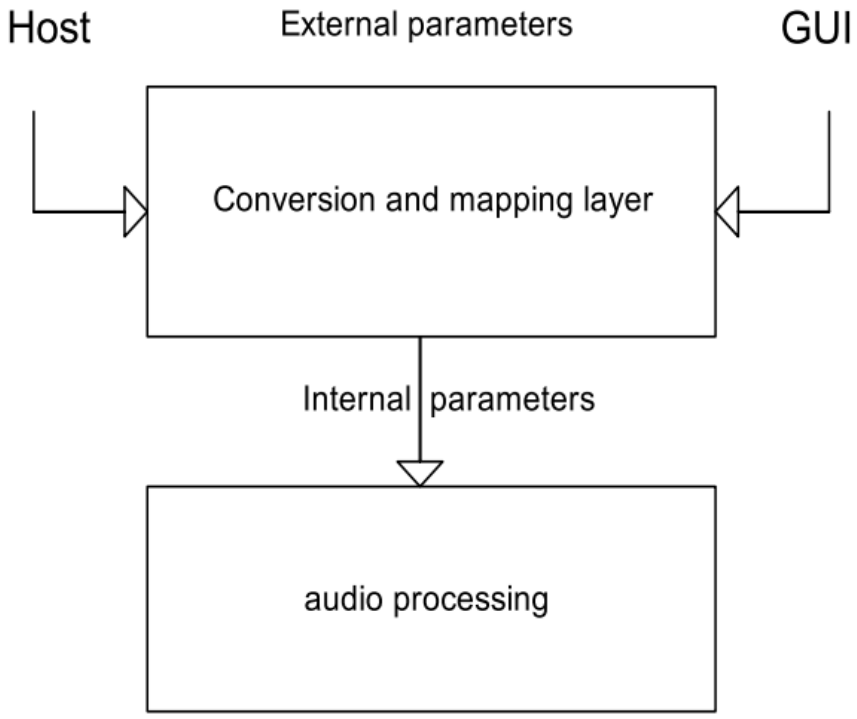
- Το **EyesWeb** προσδιορίζει με τη βοήθεια σημαιών, αν η παράμετρος μπορεί να αλλάξει. κατά τη σχεδίαση, κατά την εκτέλεση (και αν ναι, αν μπορεί να εξαχθεί), και αν είναι αρχικά απενεργοποιημένη.

- Το **AudioUnits** παρέχει ένα πραγματικά πλούσιο API, με πολλούς προκαθορισμένους τύπους παραμέτρων, οι οποίοι έχουν τις δικές τους μονάδες, χαρτογράφηση και εύρους. Σε αυτά περιλαμβάνονται και τα ευρετηριασμένα, boolean, τοις εκατό, το δευτερόλεπτο, η φάση, το cent, τα decibel, το hertz, το pan, ένας γενικός τύπος που είναι float μεταξύ 0,0 και 1,0, και άλλοι. Ο προγραμματιστής μπορεί επίσης να προσθέσει και τους δικούς του τύπους.

3.4.2 Ενημέρωση παραμέτρων

Ο μηχανισμός ενημέρωσης των παραμέτρων θα πρέπει να λαμβάνει αποτελεσματικά υπόψη το γεγονός ότι ο ρυθμός ενημέρωσης των παραμέτρων (δεν υπάρχει "ρυθμός" με την αυστηρή έννοια, αλλά χρησιμοποιήσαμε αυτή τη λέξη για λόγους ευκολίας) είναι ασύγχρονος στη φύση του, σε γενικές

γραμμές σημαντικά χαμηλότερος από τη ροή του ήχου, και ότι πολλές αλλαγές θα μπορούσαν να συμβούν ταυτόχρονα. Ορισμένες παράμετροι είναι γνωστές εκ των προτέρων (π.χ. όπως τα συμβάντα MIDI σε ένα κομμάτι του sequencer), και ορισμένες όχι (π.χ. η ζωντανή ενημέρωση μέσω κουμπιών υλικού εξοπλισμού). Η ενημέρωση των παραμέτρων μπορεί να αναπαρασταθεί ως μια λειτουργία δύο βημάτων: οι εξωτερικές παράμετροι που έχουν αλλάξει πρέπει να σταλούν στο plugin- στη συνέχεια, μια μετατροπή - αν είναι απαραίτητο - πρέπει να καταστήσει τις αντίστοιχες εσωτερικές παραμέτρους διαθέσιμες στη διαδικασία σε σωστή μορφή. Το Σχήμα 3.4 απεικονίζει αυτόν τον μηχανισμό.



Σχήμα 3.4: Εσωτερικές και εξωτερικές παράμετροι

3.4.2.1 Μηχανισμός ενημέρωσης

Ο πιο απλός τρόπος ενημέρωσης των παραμέτρων είναι αυτός που εφαρμόζεται στο LADSPA και συνίσταται σε έναν μηχανισμό δημοσκόπησης. Κατά τη στιγμή της σύνδεσης, ο κεντρικός υπολογιστής δίνει στο plugin τις διευθύνσεις στις οποίες θα γράψει τις τιμές των εξωτερικών παραμέτρων. Το plugin μπορεί να ανακτήσει αυτές τις τιμές κατά τη διάρκεια της λειτουργίας της διαδικασίας - και μόνο τότε -, υποθέτοντας ότι μπορεί να έχουν αλλάξει. Ό,τι πρέπει να γίνει για την αντιστοίχιση και τη μετατροπή αυτών των εξωτερικών παραμέτρων μπορεί να γίνει μόνο τότε.

Στη δεύτερη προσέγγιση (VST, jMax), ο κεντρικός υπολογιστής ή το GUI καλούν μια συγκεκριμένη πρόσθετη μέθοδο, που συνήθως ονομάζεται setParameter. Αυτή η μέθοδος μπορεί να κληθεί μόνο μεταξύ δύο επεξεργασιών buffer. Οι απαραίτητες μετατροπές και αντιστοιχίσεις θα πρέπει να υλοποιούνται εκεί από τον προγραμματιστή του plugin, έτσι ώστε οι εσωτερικές παράμετροι να είναι ενημερωμένες όταν καλείται η συνάρτηση process για την επόμενη επεξεργασία ρυθμιστικού διαστήματος.

Ως τρίτη περίπτωση, ορισμένα πρότυπα (DXi, AU) παρέχουν ένα επίπεδο μετατροπής στο API μεταξύ του plugin και του κεντρικού υπολογιστή ή του GUI. Η μετατροπή και η αντιστοίχιση γίνονται αυτόματα με τη βοήθεια της δομής πληροφοριών παραμέτρων. Ωστόσο, εξακολουθεί να είναι απαραίτητη η ανάκτηση της τιμής των παραμέτρων στη συνάρτηση διεργασίας, και εδώ πρέπει να γίνει πρόσθετη μετατροπή (π.χ. υπολογισμός

συντελεστών φίλτρου από τη συχνότητα αποκοπής και τον συντονισμό, ή τον έλεγχο της αμοιβαίας συνέπειας των παραμέτρων).

Οι τρόποι χειρισμού των συμβάντων διαφέρουν μεταξύ των προτύπων plugins. Μια κοινή και απλή λύση είναι η αντιμετώπιση αυτού του συνόλου τιμών όπως και άλλων παραμέτρων: ανάγνωση και εγγραφή της τιμής του μηνύματος με τις μεθόδους GetParameter και SetParameter και αποφασίζοντας τι πρέπει να γίνει με μια συνθήκη 'case' ή 'if'. Μπορεί επίσης να αντιμετωπιστεί απευθείας στη μέθοδο SetParameter (EyesWeb).

3.4.2.2 Αυτοματισμοί

Ο αυτοματισμός επιτρέπει στο χρήστη να καταγράφει τις αλλαγές των παραμέτρων κατά μήκος μιας ακολουθίας και να τις αναπαράγει. Αυτές οι αλλαγές μπορούν είτε να καταγραφούν inline (δηλ. σε πραγματικό χρόνο, κατά τη διάρκεια της αναπαραγωγής) είτε εκτός σύνδεσης (π.χ. με τη γραφική επεξεργασία μιας καμπύλης σε ένα ακολουθούμενο κομμάτι). Η εγγραφή αυτοματισμού συνίσταται στη λήψη της παραμέτρου τιμής σε κάθε χρονικό διάστημα.

Τα περισσότερα πρότυπα τείνουν να το επιτρέπουν αυτό (DXi, AU, MAS, RTAS, VST), δεδομένου ότι είναι ένα αρκετά ισχυρό εργαλείο. Οι παράμετροι που δηλώνονται ως αυτοματοποιημένες παράμετροι θα πρέπει να είναι " διαθέσιμο σε πραγματικό χρόνο", δηλαδή δεν θα πρέπει να εισάγουν πολύ βαριές υπολογιστικές διαδικασίες ή κατανομή στην μνήμη.

3.4.2.3 Ενθυλάκωση παραμέτρων και μεταδεδομένων

Κάποιες πληροφορίες επικεφαλίδας μπορεί να παρέχονται στο plugin κατά τη διάρκεια της εκτέλεσης, σε προσθήκη της τιμής της νέας παραμέτρου. Ακριβώς, η χρονική στιγμή κατά την οποία πραγματοποιείται η αλλαγή, και ο τρόπος τροποποίησης της τιμής μπορούν να καθοριστούν.

3.4.2.3.1 Τύπος δεδομένων

Ενός έχει αναφερθεί στην ενότητα «Ενημέρωση παραμέτρων», ορισμένα πρότυπα κάνουν χρήση ενός μοναδικού -ή ενός περιορισμένου συνόλου- τύπου δεδομένων για τη μεταφορά. Σε αυτή την περίπτωση, μια σημαία που επισυνάπτεται στην παράμετρο σε μια δομή parameter-info, καθορίζει τον τύπο, στον οποίο πρέπει να μετατραπεί η τιμή.

3.4.2.3.2 Χρονοσημάνσεις

Ο χρήστης που ελέγχει τη διεπαφή μπορεί να ενεργήσει ανά πάσα στιγμή, αδιαφορώντας για το τι θα κάνει το plugin. Έτσι, περισσότερες από μία αλλαγές μπορούν να συμβούν κατά τη διάρκεια του χρονικού διαστήματος που επεξεργάζεται ένα χρονικό τμήμα. Κατά την εξέταση σημειακών γεγονότων, όπως ο ήχος επιτίθεται, ο συγχρονισμός μπορεί να παίξει σημαντικό ρόλο στην απόδοση του ήχου και χρειάζεται μεγαλύτερη ακρίβεια από τη χρονική τομή, η οποία μπορεί να διαρκεί περισσότερο από 50 ms.

Ως εκ τούτου, στις αλλαγές των παραμέτρων μπορούν να επισυνάπτονται χρονοσημάνσεις, για να προσδιορίζεται η ακριβή δειγματοληπτική ακρίβεια του χρόνου κατά τον οποίο θα πρέπει να πραγματοποιηθούν. Αυτά τα συμβάντα μπορούν να είναι μπουν σε ουρά, η χρονοσήμανση τους να μετατραπεί σε θέση δείγματος εντός ενός χρονικού διαστήματος buffer, και στη συνέχεια να εκτελεστούν με τη σωστή χρονική κατανομή κατά τη διάρκεια του επόμενου buffer επεξεργασίας.

Γενικά, η χρονοσήμανση των παραμέτρων περιορίζεται στα μηνύματα midi. Μόνο η EyesWeb το χρησιμοποιεί για άλλες παραμέτρους ελέγχου.

3.4.2.3.3 Παρεμβολή

Από την άλλη πλευρά, για τις περισσότερες παραμέτρους που θεωρήθηκαν προηγουμένως ως "συνεχείς", η ακρίβεια αυτή δεν έχει σημασία, δεδομένου ότι οι χρονικές τομές του ήχου είναι πραγματικά μικρές, αλλά η συνεχής φύση των παραμέτρων πρέπει να τηρείται για να αποφεύγεται ο "θόρυβος" στο σήμα εξόδου του ήχου. Ως εκ τούτου, διάφορα πρότυπα έχουν εφαρμόσει έναν τρόπο παρεμβολής των τιμών των παραμέτρων για την εξομάλυνση της αλλαγής και την αποφυγή κενών στη ροή ήχου που θα δημιουργούσαν αυτά τα "κλικ". Ένας συνηθισμένος τρόπος είναι να λαμβάνεται υπόψη η τελευταία τιμή της συγκεκριμένης παραμέτρου και να πραγματοποιείται παρεμβολή μεταξύ της νέας τιμής και της προηγούμενης. Για την εκτέλεση της παρεμβολής μπορεί να υπάρχουν διάφορα είδη παρεμβολής, όπως γραμμική (DXi, EyesWeb, AU), τετραγωνική (DXi) ή ημιτονοειδής (Dxi).

3.4.3 Διατήρηση παραμέτρων και προεπιλογές (presets)

Η εμμονή των παραμέτρων είναι μια αποθήκευση των τιμών των παραμέτρων, που επιτρέπει τον χρήστη να βρίσκει τις ίδιες τιμές - και όχι τις προεπιλεγμένες - όταν ο χρήστης ανοίξει ξανά τις παραμέτρους στο plugin. Αυτό σημαίνει ότι όταν ο χρήστης κλείσει τον κεντρικό υπολογιστή και τον επανεκκινήσει για μια νέα συνεδρία, οι μόνιμες προσαρμογές των παραμέτρων θα είναι οι ίδιες όπως όταν έκλεισε. Αυτή η λειτουργία είναι χρήσιμη όταν χειρίζονται plugins με πολύ μεγάλο έλεγχο τιμών, όπως για παράδειγμα ένας ισοσταθμιστής (equalizer).

Οι προεπιλογές είναι στην πραγματικότητα μια προέκταση του συστήματος παραμονής στη βούληση του χρήστη: ο χρήστης μπορεί να αποφασίσει να αποθηκεύσει διαφορετικά σύνολα παραμέτρων που του αρέσουν, συνήθως προσδιορίζοντάς τα με ένα όνομα. Ο τρόπος αποθήκευσης του preset εξαρτάται από το μέγεθος του preset: αν το preset είναι μικρό, μπορεί να αποθηκευτεί από τον κεντρικό υπολογιστή (VST) ή σε έναν καταχωρητή (DXi)- ενώ αν το preset είναι μεγαλύτερο (π.χ. περιέχει μια κυματομορφή, ή μια εικόνα), θα αποθηκεύεται σε ξεχωριστό αρχείο ως bytestream.

3.5 Ενσωμάτωση του περιβάλλοντος υποδοχής

Για να συνεργαστούν, ο κεντρικός υπολογιστής και το plugin πρέπει να γνωρίζουν το ένα για το άλλο στατικές και δυναμικές πληροφορίες. Κάποιες είναι θεμελιώδεις άλλες είναι προαιρετικές, οι επιλογές μπορεί να διαφέρουν πολύ μεταξύ των προτύπων. Μπορούν είτε να ζητηθούν είτε να ειπωθούν σε οποιαδήποτε στιγμή, αλλά ο χρόνος κατασκευής - ή ο χρόνος ανοίγματος - είναι πιο συνηθισμένος. Περιγράφονται παρακάτω οι πιο σημαντικές από αυτές.

3.5.1 Γενικές πληροφορίες

3.5.1.1 Μεταπληροφορίες

Για τα περισσότερα πρότυπα είναι δυνατό να δώσετε ένα όνομα στο plugin, ένα όνομα κατασκευαστή, μια περιγραφή και μια κατηγορία. Θα πρέπει επίσης να υπάρχει ένας τρόπος να προσθέσετε αριθμό έκδοσης σε ένα plugin καθώς και να υπάρχει ένας τρόπος να ζητήσετε το όνομα και την έκδοση του host για λόγους συμβατότητας (Τα plugins μπορούν να δηλώσουν τις ικανότητές τους ανάλογα με το τι υποστηρίζουν οι hosts και επίσης ανάλογα με τα γνωστά σφάλματα σχετικά με έναν συγκεκριμένο host). Είναι πολύ πιο βολικό από το να παρέχετε ειδικές εκδόσεις του ίδιου plugin για διαφορετικές εκδόσεις και είδη host.

3.5.1.2 Ρύθμιση ήχου.

Ιδιότητες ακροδέκτη ήχου

Στο παρόν έγγραφο, υποθέτουμε ότι μια είσοδος ή έξοδος ήχου (ακροδέκτης) μπορεί να αποτελείται από μονοκάναλα ομαδοποιημένα μεταξύ τους που πρέπει να αντιμετωπίζονται ως μία οντότητα. Αυτό βγάζει νόημα όταν πρόκειται για μονοφωνικά, στερεοφωνικά ή surround κανάλια. Σε ένα γράφημα plugin ή σε ένα πλαίσιο υποδοχής, ένα plugin μπορεί να διαπραγματευτεί τις συνδέσεις του με άλλα plugins ή απευθείας με τον κεντρικό υπολογιστή. Για το σκοπό αυτό, ένα plugin πρέπει να καθορίσει το συνολικό αριθμό των καναλιών ήχου που υποστηρίζει και τον τρόπο με τον οποίο πρέπει να ομαδοποιηθούν εάν είναι απαραίτητο. Συχνά μπορούν να παρέχονται συμβολοσειρές χαρακτήρων για την ονομασία των ακροδεκτών (βλ. Πίνακα 3.2). Συχνά υπάρχει μόνο ένας ακροδέκτης εισόδου και ένας ακροδέκτης εξόδου με τα ίδια χαρακτηριστικά (κλασικά αλυσιδωτά ένθετα), αλλά στην περίπτωση των μουσικών οργάνων, πολλαπλοί ακροδέκτες εξόδου είναι συνηθισμένες (π.χ. ένας ανά κανάλι MIDI ή ένας ανά ήχο τυμπάνου προκειμένου να συμπίεση/εξισορρόπηση ξεχωριστά) και με spacializer, panner ή down mixing plugins, οι ιδιότητες των pin εισόδου και εξόδου μπορεί να είναι διαφορετικές (π.χ. mono in - surround out). Επιπλέον, ορισμένα κανάλια μπορούν να επισημανθούν ως side-chain.

	Τύπος	Κανάλια	Όνομα	Sidechain	Διασυνδεδεμένο	Διακόπτης IO
VST	32-bit float	οτιδήποτε	ναι	ναι	όχι	θεωρητικά
AU	32-bit float	οτιδήποτε		όχι	και οι δύο	ναι
LADSPA	32-bit float	οτιδήποτε	ναι	όχι	όχι	όχι
RTAS	32-bit float	οτιδήποτε		ναι	όχι	όχι
DIRECTX	WaveFormatEx	οτιδήποτε	ναι	όχι	both	θεωρητικά
MAS	32-bit float	11x11 max	ναι	ναι	όχι	όχι
EYESWEB	32-bit float	οτιδήποτε		όχι		
MAX-like	32-bit float	οτιδήποτε	όχι	ναι	όχι	ναι

Πίνακας 3.2: Ιδιότητες ακροδεκτών ήχου

Ιδιότητες επεξεργασίας ήχου

Ένα plugin μπορεί να ειδοποιεί το περιβάλλον για τις ιδιότητες του DSP του. Αυτές οι ιδιότητες περιλαμβάνουν:

- **Ο τύπος επεξεργασίας buffer**: 'in place' ή 'buffer to buffer' καθώς και 'accumulation' για εφέ αποστολής ή 'replacing' για εισαγωγές (LADSPA, DXi,)
- **Η δημιουργία μιας ουράς**: (DXi3 , VST). Η ουρά είναι το τμήμα του επεξεργασμένου ηχητικού σήματος, που προστίθεται στο τέλος της ροής που έχει υποστεί επεξεργασία, σε τέτοια εφέ όπως οι αντηχήσεις, οι καθυστερήσεις κλπ.

- **Η καθυστέρηση**: (VST). Η καθυστέρηση είναι η καθαρή καθυστέρηση που εισάγεται από τον υπολογισμό, π.χ. για τον υπολογισμό ενός FFT στα 44100Hz με 512 δείγματα hopesize, η καθυστέρηση θα ήταν 512/44100 = 11,5 ms.
- **Η ποιότητα σε πραγματικό χρόνο:** Σε κάποιο GUI του plugin, μπορεί κανείς να επιλέξει χαμηλότερη ποιότητα (που συνήθως επιτυγχάνεται με μείωση της δειγματοληψίας) για την προεπισκόπηση του ηχητικού σήματος, για να επιτύχει ταχύτερο υπολογισμό.
- Η κατανάλωση CPU

3.5.2 Πληροφορίες χρόνου εκτέλεσης

Το plugin ήχου μπορεί συχνά να χρησιμοποιηθεί μέσα σε ένα sequencer, όπου πολλά κομμάτια είναι τοποθετημένα κατά μήκος μιας κοινής γραμμής χρόνου. Κατά τη διάρκεια της εκτέλεσης, κάποιο πρόσθετο - MAS, VST, DX- μπορεί να στείλει ή να ζητήσει πληροφορίες χρόνου (οι πληροφορίες χρόνου περιλαμβάνουν το τέμπο (bpm), τη χρονική υπογραφή, τη χρονική ή/και μουσική θέση) σχετικά με τη θέση του τρέχοντος χρονικού τμήματος που επεξεργάζεται μέσα σε αυτό το χρονοδιάγραμμα. Είναι πολύ χρήσιμο να μπορούν να ρυθμίζονται οι παράμετροι σε τιμές με μουσικό νόημα (π.χ. ένας αριθμός τετάρτων αντί για χρόνο σε χιλιοστά του δευτερολέπτου) ή ακόμη και να συγχρονίζετε κάποια μοτίβα μετασχηματισμού με την τρέχουσα θέση σε ένα μεσάζον.

Με κάποιο πρότυπο - MAS - είναι δυνατή ακόμη και η αποστολή πληροφοριών ελέγχου στον κεντρικό υπολογιστή, που σχετίζεται με την εκτέλεση του sequencer, όπως 'start', 'stop', 'goto locator', 'rewind' κλπ. επιτρέποντας έτσι στους αλγορίθμους παρακολούθησης ρυθμών να συγχρονίσουν έναν κεντρικό υπολογιστή σε ένα ηχογραφημένο κομμάτι.

Μπορεί επίσης να είναι κανείς προετοιμασμένος με πρότυπα όπως το VST, DirectX, την ταχύτητα δειγματοληψίας ή το μέγεθος του buffer μπορεί να αλλάξει κατά τη διάρκεια της εκτέλεσης. Μη δύναμη του 2 ή ακόμη και το μέγεθος του buffer είναι πιθανά.

3.5.3 Access to the plugin

3.5.3.1 Τοποθεσία Plugin

Αν και τα plugins αποθηκεύονται συνήθως σε ένα κοινό μέρος, δεν υπάρχει πραγματική τυποποιημένη τοποθεσία για την αποθήκευση των plugins. Παρόλα αυτά, συχνά θα βρει κανείς μια προτεινόμενη διαδρομή για το την τοποθεσία των plugins, για να γίνουν τα πράγματα πιο καθαρά. Η δυνατότητα για την ύπαρξη πολλαπλών πιθανών διαδρομών εξαρτάται στην πραγματικότητα από τον κεντρικό υπολογιστή. Για ορισμένα πρότυπα, μπορεί να είναι είτε με την προσθήκη νέων θέσεων σε μια μεταβλητή περιβάλλοντος (LADSPA) είτε αλλάζοντας τη διαδρομή, είτε στον καταχωρητή συστήματος (DirectX), είτε αναζητώντας ένα plugin κατά τη διάρκεια του χρόνου εκτέλεσης. Ωστόσο, αυτό εξαρτάται περισσότερο από τη στρατηγική του κεντρικού υπολογιστή και από από το λειτουργικό σύστημα παρά από το πρότυπο.

3.5.3.2 Ταυτότητα Plugin

Προκειμένου να είναι γνωστό από τον κεντρικό υπολογιστή, ένα plugin θα πρέπει να παρέχει έναν μοναδικό τρόπο για να αναγνωρίζεται ανάμεσα σε άλλα plugins: μια μοναδική ταυτότητα (ID). Ανάλογα με το πεδίο εφαρμογής στο οποίο ο κεντρικός υπολογιστής υποτίθεται ότι επιδιώκει να βρει τα plugins, η ταυτότητα μπορεί να είναι μια απλή τιμή, ένα πιο σύνθετο κλειδί εγγραφής.

Αυτή η ταυτότητα μπορεί να αποτελείται από 1 ή πολλές τιμές, οι οποίες υποτίθεται ότι επιλέγονται διαφορετικά από την ταυτότητα άλλων plugins, από τον προγραμματιστή του plugin. Σε αυτή την περίπτωση, είναι συνήθως ένας μακρύς ακέραιος αριθμός ή ένας συνδυασμός μακρών ακέραιων αριθμών: αυτή είναι η περίπτωση LADSPA, AU, VST, RTAS, MAS κλπ. Στην περίπτωση πολλαπλών ταυτοτήτων, μπορεί κανείς να βρει την ταυτότητα κατασκευαστή (AU, MAS, RTAS), μια ταυτότητα παραλλαγής (MAS), μια ταυτότητα τύπου πρόσθετου (AU) κλπ. Σημειώστε ότι μια ταυτότητα plugin μπορεί να καταχωρηθεί από τον κατασκευαστή του κεντρικού υπολογιστή για να να γίνει "επίσημο" και να αποφύγει συγκρούσεις με αναγνωριστικά τρίτων κατασκευαστών.

Αξίζει να σημειωθεί ότι τα αρχεία κοινής βιβλιοθήκης στα οποία μεταγλωττίζονται τα plugins μπορεί να περιέχουν πολλά plugins - αυτό μπορεί να γίνει σε LADSPA, AU, MAS και VST (χρησιμοποιώντας ένα κέλυφος plugin). Σε αυτή την περίπτωση, κάθε plugin μέσα σε αυτή τη βιβλιοθήκη θα πρέπει να έχει μια μοναδική ταυτότητα. Η άλλη λύση είναι ειδική για το DirectX και συνίσταται στην καταχώριση του pluginobject στο μητρώο του συστήματος μέσω GUID (Globally Unique ID), το οποίο είναι κλειδιά εγγραφής 128 bit που δημιουργούνται αυτόματα από το σύστημα. Αυτό η λύση έχει το πλεονέκτημα ότι αποφεύγονται οι συγκρούσεις ταυτότητας μεταξύ των plugins, λόγου της άγνοιας των προγραμματιστών για όλα τα ήδη υπάρχοντα ID των plugins που χρησιμοποιούνται από την Cakewalk και την SonicFoundry ως "εφέ DirectX" και από την EyesWeb για να χρησιμοποιούν αυτόν τον μηχανισμό.

3.5.3.3 Σημεία εισόδου

Ο κώδικας του plugin που είναι αποθηκευμένος σε ένα αρχείο Shared Object ή Dynamic Linked Librairy (αυτές οι βιβλιοθήκες είναι αρχεία *.so στο Linux και *.dll στα Windows) μπορεί να περιέχει ένα ή περισσότερα plugins. Η εφαρμογή host προσπελαύνει το plugin καλώντας τη συνάρτηση (ή τις συναρτήσεις) του σημείου εισόδου. Ο χρήστης έχει στη συνέχεια πρόσβαση στο plugin μέσω της διεπαφής του host με έναν εντελώς ολοκληρωμένο τρόπο. Η συνάρτηση Entry Point θα ενσαρκώσει το plugin και θα παράσχει πληροφορίες στον κεντρικό υπολογιστή.

3.6 Επίλογος

Μία σύνοψη αυτού του κεφαλαίου, από μια αφηρημένη άποψη, τα πρότυπα των plugins μοιάζουν πολύ και μοιράζονται πολλές λειτουργίες. Ορισμένα είναι πολύ γρήγορα και απλά στην ανάπτυξη, όπως το LADSPA, αφήνοντας τη δύσκολη δουλειά στον κεντρικό υπολογιστή και τον χρήστη (GUI, χαρτογράφηση), ενώ άλλα, όπως το DirectX ή το RTAS, έχουν πιο δύσκολη καμπύλη εκμάθησης επειδή δίνουν πολλές δυνατότητες στους κατασκευαστές των plugins (προσαρμοσμένο GUI, χειρισμός επιφανειών χειρισμού ή ενσωματωμένου DSP, αυτοματισμός, MIDI κλπ.). Μεταξύ μπορούμε να βρούμε AudioUnits και VST που προσπαθούν να τα βγάλουν πέρα μέσω εύκολων APIs για αρχή, αλλά επιτρέποντας ακόμα πολλές δυνατότητες για (αρκετά) έμπειρους προγραμματιστές.

Ωστόσο, όταν εξετάζουμε τις λεπτομέρειες, μπορούμε να αναδείξουμε πολλές διαφορές. Αυτές οι διαφορές εξαρτώνται κυρίως από το πλαίσιο στο οποίο γεννήθηκαν τα πρότυπα των plugins. Για ιστορικούς, τεχνικούς και εμπορικούς λόγους, οι κατασκευαστές host ξεκίνησαν σε συγκεκριμένες πλατφόρμες, στοχεύοντας σε διαφορετικά είδη χρηστών ή/και δραστηριοτήτων μεταξύ μηχανικών ήχου, μουσικών, επαγγελματιών, ερασιτεχνών, post-production, πολυμέσων, βίντεο κλπ. Αυτοί οι αρχικοί περιορισμοί παραμένουν ακόμη παρόντες στα πρότυπα λόγω της συμβατότητας προς τα πίσω, παρά το γεγονός ότι από εκείνες τις πρώτες εποχές οι εταιρείες και οι υποδοχείς έχουν εξελιχθεί, έχουν αλλάξει τη μορφή της πλατφόρμας ή ακόμη και τον στόχο του πελάτη.

Σε αυτό το συγκεκριμένο πλαίσιο, η ομάδα GMPI (Generalized Musical Plugin Interface) δημιουργήθηκε στα τέλη του 2002, με πρωτοβουλία του Ron Kupper από την Cakewalk και υπό την εποπτεία της MMA (MIDI Manufacturer Association). για να να σχεδιάσει ένα νέο ανοικτό και διαπλατφορμικό πρότυπο plugins, το οποίο θα ικανοποιούσε ολονών τις ανάγκες, και να θέσει σε κοινή βάση όλη την εμπειρία που συσσωρεύτηκε από τους ανθρώπους από το τα πρώτα συστήματα host και plugin. Ωστόσο, υπάρχει ακόμη πολύς δρόμος μέχρι να μπορέσουμε να να χρησιμοποιήσουμε και να αναπτύξουμε ένα μοναδικό είδος plugin που θα είναι εύκολο και γρήγορο να αναπτύσσεται, να είναι εξαιρετικά επεκτάσιμο και αποδοτικό. Εν τω μεταξύ, η ανάπτυξη για διαφορετικά πρότυπα και πλατφόρμες εξακολουθεί να είναι απαραίτητη και χρονοβόρα. Επιπλέον, η επιλογή των υποστηριζόμενων προτύπων είναι θεμελιώδης για τον καθορισμό του κοινού που θα απευθυνθεί η εφαρμογή.

Κεφάλαιο 4ο: JUCE Framework

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το JUCE Framework, ένα πλαίσιο εφαρμογών ανοιχτού κώδικα C++ για πολλαπλές πλατφόρμες που αναπτύσσεται από το 2004 και αγοράστηκε από τη ROLI τον Νοέμβριο του 2014. Χρησιμοποιείται για την ανάπτυξη desktop εφαρμογών και εφαρμογών για κινητά τηλέφωνα. Αυτό το κεφάλαιο παρουσιάζει τη συνολική σχεδίαση των αρχείων αρχιτεκτονικής για το JUCE.

4.1 Εισαγωγή

Ο στόχος του JUCE [19] είναι να επιτρέψει στο λογισμικό να γράφεται έτσι ώστε ο ίδιος πηγαίος κώδικας να μεταγλωττίζεται και να εκτελείται πανομοιότυπα στα Windows, Mac OS X, Linux για τις desktop συσκευές, και σε Android και iOS για τις κινητές συσκευές. Ένα αξιοσημείωτο χαρακτηριστικό του JUCE σε σύγκριση με άλλα παρόμοια πλαίσια είναι το μεγάλο σύνολο λειτουργιών ήχου που διαθέτει. Αυτές οι υπηρεσίες, οι δυνατότητες διεπαφής χρήστη και η δυνατότητα εξαγωγής σε πολλές πλατφόρμες τοποθετούν το JUCE ως ένα εξαιρετικό framework ώστε να έχει στο μέλλον λιγότερο κώδικα για να διατηρείται ενημερωμένος, και απλούστερη χρήση.

4.2 Συστατικά κλάσεων JUCE

Για την υλοποίηση ενός πλήρους προγράμματος, τα γραφικά στοιχεία πρέπει να συνδυαστούν με κλάσεις του JUCE. Για την υλοποίηση ενός πλήρους προγράμματος, τα γραφικά στοιχεία που περιγράφηκαν στην προηγούμενη ενότητα πρέπει να συνδυαστούν με κλάσεις JUCE. Στο JUCE Framework, η κλάση component είναι η βασική κλάση για όλα τα αντικείμενα διεπαφής χρήστη της JUCE. Η ακόλουθη ενότητα εξηγεί τη σχέση μεταξύ των αρχείων αρχιτεκτονικής GUI και τους μηχανισμούς της JUCE.

4.2.1 Μηχανισμός γονέα και παιδιού

Όπως τα περισσότερα frameworks, το JUCE έχει μια ιεραρχία αντικειμένων component, οργανωμένων σε δενδρική δομή. Ο συνήθης τρόπος για να ορίσετε ένα component ως παιδί ενός άλλου component είναι να κάνετε:

parent -> addAndMakeVisible(child);.

Αυτή η συνάρτηση θέτει το στοιχείο-παιδί ως ορατό, επειδή δεν είναι εξ ορισμού ορατό. Πολλαπλές λειτουργίες είναι προσβάσιμες για εκτέλεση μέσω αυτού του δέντρου component, με μεθόδους που δίνουν την παιδί component στο δείκτη i, ή δίνουν το γονικό. Υπάρχει ακόμη και μια συνάρτηση που επιτρέπει να λάβετε τον γονέα ενός component με συγκεκριμένο τύπο, αυτού του τύπου είναι μια παράγωγη κλάση της Juce::Component. Ωστόσο, αυτή η συνάρτηση δεν υπάρχει για το παιδί, και συνεπάγεται ότι πρέπει να γίνει dynamic_cast αν θέλετε να πάρετε ένα παιδί συγκεκριμένου τύπου.

4.2.2 Μηχανισμός Component Setup

Πρώτα απ' όλα, ένα Component σχεδιάζεται αν είναι ορατό, όπως και ο γονέας του. Αν ένα Component δεν είναι ορατό, το παιδί του και όλα τα παιδιά του κλπ., δεν θα είναι ορατά, αλλά καθώς η συνάρτηση addAndMakeVisible χρησιμοποιείται τις περισσότερες φορές, αυτό δεν θα πρέπει να αποτελεί πρόβλημα. Ένα Component έχει ένα Rectangle<int> boundsRelativeToParent, που περιέχει

τις συντεταγμένες x και y, καθώς και το πλάτος και το ύψος του. Όπως υποδηλώνει το όνομα της μεταβλητής, τα όρια ενός Component είναι σχετικά με τον γονέα του, και όχι απόλυτα στο παράθυρο.

4.2.3 Μηχανισμός σχεδίασης

Ένα Component διαθέτει δύο εικονικές συναρτήσεις (συναρτήσεις-κατόχους θέσεων που πρέπει να υλοποιήσει ο προγραμματιστής) που αποτελούν τα κύρια εργαλεία για το χειρισμό μιας δυναμικής διάταξης, τις συναρτήσεις void resized() και void paint(Graphics& g). Η resized καλείται κάθε φορά που αλλάζουν τα όρια ενός Component και η paint όταν η σημαία του Component υποδεικνύει ότι πρέπει να ξαναβαφτεί. Για παράδειγμα, ο κέρσορας του ποντικιού που βρίσκεται πάνω του, ένα κλικ του ποντικιού, η αλλαγή των ορίων του στοιχείου ή η ανάγκη επαναζωγραφισμού ενός ή περισσοτέρων από τα θυγατρικά του υποδεικνύουν ότι πρέπει να επαναζωγραφιστεί.

Υπάρχει μια τάξη σχεδιασμού που ονομάζεται LookAndFeel που επιτρέπει την προσαρμογή της διεπαφής. Το LookAndFeel καθορίζει την εμφάνιση της όλων των widgets της JUCE και οι υποκλάσεις μπορούν να χρησιμοποιηθούν για την εφαρμογή διαφορετικών skins στην εφαρμογή.

Προφανώς υπάρχουν πολλά περισσότερα στο Juce::Component, αλλά αυτά είναι τα βασικά, ή τουλάχιστον αυτά που χρειάζονται τα αρχεία αρχιτεκτονικής δομής.

4.3 Αρχιτεκτονικό αρχείο JuceGUI

Για να συνοψίσουμε αυτό που έχει παρατηρηθεί στο παρελθόν, το σύστημα των widgets πρέπει να προσαρμοστεί στη μηχανική του Juce::Component σε ένα αρχείο αρχιτεκτονικής που ονομάζεται JuceGUI.h. Η επόμενη ενότητα εξετάζει παραδείγματα με σχολιασμούς.

4.3.1 Δύο διαφορετικά είδη αντικειμένων

Υπάρχουν δύο είδη αντικειμένων που χρησιμοποιούνται στην προσαρμογή:

- uiComponent, τα οποία είναι βασικά οποιαδήποτε στοιχεία, όπως ρυθμιστικά ή κουμπιά

- uiBox, το οποίο είναι στοιχείο δοχείου, και έτσι μπορεί να περιέχει ένα uiComponent ή κάποιο άλλο uiBox.

Και οι δύο είναι παράγωγες κλάσεις μιας uiBaseComponent, η οποία είναι η ίδια μια παραγόμενη κλάση της Juce::Component.

Η κλάση uiBaseComponent συγκεντρώνει μεθόδους που μοιράζονται τόσο το uiBox όσο και το uiComponent, όπως η void setRatio(), η int getTotalWidth(), κ.λπ. Με αυτόν τον τρόπο, αποφεύγονται τα πολλά dynamic_cast στον κώδικά μας. Ακολουθεί τι περιέχει η κλάση uiBaseComponent:

```
float fHRatio, fVRatio;
int fTotalWidth, fTotalHeight;
int fDisplayRectHeight, fDisplayRectWidth;
String fName;


uiBaseComponent (int totWidth, int totHeight, String name);
```

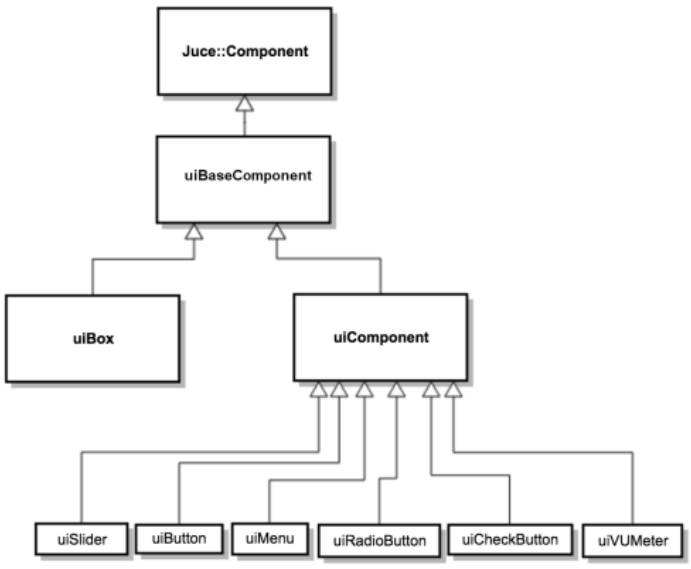
```
int getTotalHeight();
int getTotalWidth();
virtual void setRatio();
float getHRatio();
float getVRatio();
String getName();
void setHRatio();
void setVRatio();
void setBaseComponentSize (Rectangle r);
void mouseDoubleClick (const MouseEvent &event) override;

virtual void writeDebug() = 0;
virtual void setCompLookAndFeel (LookAndFeel*laf) = 0;
```

Η συνάρτηση mouseDoubleClick είναι μια συνάρτηση της JUCE που μπορεί να αντικατασταθεί, η οποία καλείται κάθε φορά που γίνεται διπλό κλικ σε ένα στοιχείο. Εδώ χρησιμοποιείται για την κλήση της συνάρτησης writeDebug, η οποία εμφανίζει διαφορετικά χαρακτηριστικά του uiBox ή του uiComponent που έχει κάνει διπλό κλικ.

Οι δύο αμιγώς εικονικές συναρτήσεις ορίζονται να έχουν τη δική τους συμπεριφορά και για τις δύο uiBox και uiComponent, χωρίς προφανώς να είναι οι ίδιες.

Η συνάρτηση virtual void setRatio(); είναι εικονική επειδή υπάρχει μια ειδική περίπτωση με το uiBox, το οποίο ρυθμίζει τη δική του αναλογία και πρέπει να ζητά από τα παιδιά του να ρυθμίσουν επίσης τις αναλογίες τους, με αναδρομικό τρόπο.

Όπως είπαμε προηγουμένως, το uiComponent κληρονομεί από το αυτές τις συναρτήσεις uiBaseComponent, και είναι η ίδια μια μητρική κλάση για πολλά διαφορετικά widgets. Εδώ είναι το διάγραμμα κληρονομικότητας:



Σχήμα 4.1: Διάγραμμα κληρονομικότητας

Μια υποκατηγορία uiComponent μπορεί να χειριστεί πολλαπλούς "τύπους" αντικειμένων.

Για παράδειγμα, το uiSlider ομαδοποιεί κάθε είδος sliders: HorizontalSlider, VerticalSlider, NumEntry και Knob.

4.3.2 Το κύριο παράθυρο

Η διεπαφή του χρήστη δεν μπορεί να συρρικνωθεί απεριόριστα ώστε να είναι πάντα ευδιάκριτη και σαφής, οπότε ορίζεται ένα ελάχιστο μέγεθος παραθύρου. Αυτό σημαίνει ότι αντί για ένα βασικό component σε ένα παράθυρο DocumentWindow (ένα παράθυρο με δυνατότητα αλλαγής μεγέθους με γραμμή τίτλου και κουμπιά μεγιστοποίησης, ελαχιστοποίησης και κλεισίματος), ένα Viewport σε ένα παράθυρο εγγράφου, το οποίο εμφανίζει γραμμές κύλισης όταν το παράθυρο αποκτά διαστάσεις μικρότερες από το ελάχιστο μέγεθος του DSP προγράμματος, επιτρέποντας πλήρη πρόσβαση στην διεπαφή του χρήστη ακόμη και στις μικρότερες διαστάσεις.

Αυτό το παράθυρο προβολής μπορεί είτε να περιέχει ένα uiBox όπως παρουσιάστηκε προηγουμένως, ή ένα uiTabs αν το πρόγραμμα απαιτεί καρτέλες.

4.3.3 Κλάση uiTabs

Η κλάση uiTabs κληρονομεί την κλάση Juce::TabbedComponent, η οποία είναι μία Juce::Component με ένα TabbedButtonBar στο ένα από τα μεγέθη του. Χρειάζεται απλώς ένα Juce::Component για κάθε καρτέλα, και ένα όνομα καρτέλας, και θα εμφανιστεί.

Μια διάταξη καρτελών είναι απαραίτητη όταν η buildUserInterface ξεκινά με μία openTabBox κλήση. Σε αυτό, ένα boolean tabLayout τίθεται true, για να γνωρίζουμε ότι πρόκειται για διάταξη καρτελών.

Κατά την ανάγνωση του buildUserInterface, ένα uiBox δίνεται στα uiTabs κάθε φορά που η τρέχουσα καρτέλα "κλείνει". Για να γίνει αυτό, μια μεταβλητή που ονομάζεται order παρακολουθεί το "επίπεδο" του τρέχοντος πλαισίου. Η order ξεκινάει από το 0, αυξάνεται όταν ανοίγει ένα νέο πλαίσιο και μειώνεται όταν κλείνει ένα πλαίσιο. Εάν η σειρά είναι 0 σε μια κλήση closeBox(), τότε κλείνει μια καρτέλα, και έτσι το τρέχον πλαίσιο προστίθεται στο uiTabs, χρησιμοποιώντας τη συνάρτηση TabbedComponent::addTab.

Μόλις κλείσουν όλες οι καρτέλες, το tabBox κλείνει επίσης, η εντολή είναι τώρα στο -1, και αυτό ενεργοποιεί τη συνάρτηση αρχικοποίησης του uiTabs, uiTabs::init(). Θα την περιγράψουμε στο επόμενη υποενότητα.

4.3.4 Αρχικοποίηση της διάταξης

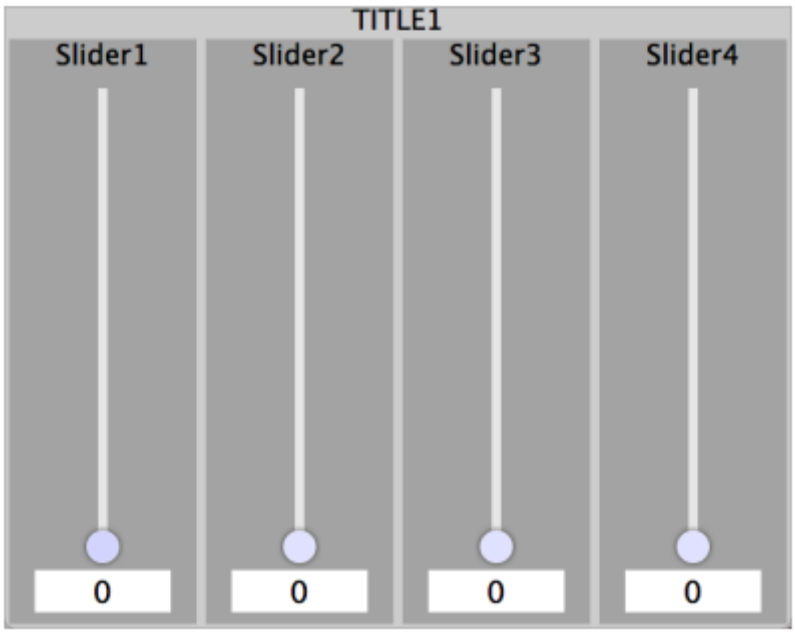
Πρώτα απ' όλα, κατά την ανάλυση των γραμμών της buildUserInterface, οι οποίες απαριθμούν τα διάφορα κουτιά και αντικείμενα που χρειάζονται να εμφανιστούν, το δέντρο δημιουργείται. Αυτό γίνεται χρησιμοποιώντας τους μηχανισμούς της Juce::Component του addAndMakeVisible. Τα διάφορα uiBaseComponent προστίθενται ως παιδιά διαφορετικών uiBox, και το μέγεθος της ορθογώνιας απεικόνισης του uiBox και το το συνολικό μέγεθος υπολογίζονται κάθε φορά όταν ένα πλαίσιο είναι κλειστό στο buildUserInterface (δηλαδή όταν καλείται η closeBox()).

Το μέγεθος του ορθογωνίου της οθόνης του uiBox είναι το άθροισμα του πλάτους του παιδιού του και το μέγιστο του ύψους του παιδιού του, και το αντίθετο ανάλογα με τον προσανατολισμό του. Αλλά τα περιθώρια προστίθενται στο πλάτος και το ύψος του ορθογωνίου απεικόνισης, 4 pixels ανά παιδί, για ένα περιθώριο 2 pixels στην κορυφή, αριστερά, κάτω και δεξιά, και προκύπτει το συνολικό μέγεθος του uiBox. Αυτό γίνεται για να αποφύγουμε ένα φαινόμενο επικάλυψης, έχοντας δύο στοιχεία να

αγγίζουν το ένα το άλλο. Ακολουθώντας το ίδιο πνεύμα, προστίθενται 12 pixels στο ύψος του box εάν πρέπει να εμφανιστεί το όνομά του, 12 pixels είναι ο χώρος που απαιτείται εξ ορισμού για την εμφάνιση του ονόματός του.

Εδώ είναι το buildUserInterface που εμφανίζει αυτό το πρόγραμμα:

```
ui_interface -> openHorizontalBox("TITLE1");
```



Σχήμα 4.2: Αναπαράσταση του μεγέθους του ορθογωνίου εμφάνισης και το συνολικό μέγεθος ενός πλαισίου με τέσσερα παιδιά.

```
ui_interface -> addVerticalSlider("Slider1", &fVslider0, 0.0f, 0.0f, 6.0f, 1.0f);
ui_interface -> addVerticalSlider("Slider2", &fVslider1, 0.0f, 0.0f, 6.0f, 1.0f);
ui_interface -> addVerticalSlider("Slider3", &fVslider2, 0.0f, 0.0f, 6.0f, 1.0f);
ui_interface -> addVerticalSlider("Slider4", &fVslider3, 0.0f, 0.0f, 6.0f, 1.0f);
ui_interface -> closeBox();
```

Στο Σχήμα 4.2, η διαφορά μεταξύ του μεγέθους του ορθογωνίου απεικόνισης και του συνολικού μεγέθους είναι εύκολα αντιληπτή. Το συνολικό μέγεθος του κουτιού εδώ με το όνομα "TITLE1" είναι το ανοιχτότερο γκρι και το μέγεθος του ορθογωνίου εμφάνισης θα ήταν τα τέσσερα πιο σκούρα γκρι ορθογώνια που είναι κολλημένα μεταξύ τους. Η διάταξη δεν ευθυγραμμίζεται άψογα λόγω του περιθωρίου που εφαρμόζεται για να αποφευχθεί η επικάλυψη των components.

Ο χώρος που αφήνεται στο επάνω μέρος του κουτιού είναι για την τίτλο, και αυτό το περιθώριο περιλαμβάνεται στο συνολικό μέγεθος.

Στις παρακάτω εξισώσεις, H είναι το συνολικό ύψος, W το συνολικό πλάτος, h το ύψος του ορθογωνίου απεικόνισης και w το πλάτος του ορθογωνίου απεικόνισης - ci είναι το νιοστό στοιχείο-παιδί του παρόντος box.

$$h = \sum_{i=0}^{n-1} (c_i * H) \tag{1}$$

$$w = \max_{i \in [0, n-1]} c_i * W \tag{2}$$

$$H = h + 4 * n \tag{3}$$

$$W = w + 4 \tag{4}$$

Το H μπορεί να αυξηθεί κατά 12 pixel, ανάλογα με την ανάγκη εμφάνισης του ονόματος του πλαισίου.

4 εικονοστοιχεία για κάθε στοιχείο-παιδί προστίθενται σε μια διάσταση για να έχουν περιθώρια μεταξύ τους, επειδή θα τοποθετηθούν το ένα δίπλα στο άλλο σε αυτή τη διάσταση, και απλά 4 εικονοστοιχεία προστίθενται στην άλλη διάσταση για να έχουν 2 εικονοστοιχεία που χωρίζουν το πλαίσιο γονέα και παιδιού σε κάθε πλευρά.

Μόλις ολοκληρωθεί το buildUserInterface, το τελευταίο πλαίσιο κλείνει και η διεπαφή χρήστη αρχικοποιείται. Αυτό το τελευταίο πλαίσιο, που θα ονομάζεται " main box " (κύριο πλαίσιο) αρχικοποιείται με τις αναλογίες 1 και 1, ακόμη και αν αυτές απαιτούνται, επειδή θα πάρει το μέγεθος του παραθύρου.

Ακολουθεί ο τρόπος με τον οποίο αρχικοποιείται η διεπαφή του χρήστη:

- Η ρύθμιση του πραγματικού μεγέθους απεικονίζεται για το κύριο πλαίσιο, επειδή το συνολικό μέγεθος ρυθμίζεται εδώ, αλλά όχι το Juce::Component bounds. Αυτό γίνεται μέσω της εντολής void setBaseComponentSize(Rectangle<int> r), η οποία θέτει το μέγεθος των components, και ειδικότερα τη σωστή τοποθέτησή τους. Συγκεκριμένα, απαιτείται μια μετατόπιση 30 pixels στο ύψος για μια καρτέλα 30 pixels είναι το ύψος που παίρνει η διάταξη η γραμμή καρτελών. Μόνο το κύριο πλαίσιο πρέπει να ρυθμιστεί με μια μετατόπιση, επειδή τα άλλα πλαίσια θα τοποθετηθούν ανάλογα με τοις συντεταγμένες των γονέων τους.

- Στη συνέχεια, υπολογίζονται οι αναλογίες για ολόκληρο το δέντρο, από τη ρίζα έως τα φύλλα. Η οριζόντια αναλογία είναι η συνολική συνιστώσα πλάτους διαιρούμενο με το πλάτος του ορθογωνίου απεικόνισης του γονέα του, το ίδιο και για το ύψος. Με αυτόν τον τρόπο, αποφεύγεται να έχουμε τα περιθώρια για να μπλέξουμε με τις αναλογίες μας, και να έχουμε ένα άθροισμα της αναλογίας, ίσο με 1 αντί για ένα που πλησιάζει το 1, αλλά όχι να είναι ακριβώς 1.

- Το τελευταίο βήμα είναι να ορίσετε το LookAndFeel για όλα τα uiComponents, τα οποία είναι για όλα τους τα φύλλα των δέντρων. Έτσι, το δέντρο είναι πλήρως αναλυμένο εκεί, από τη ρίζα μέχρι τα φύλλα.

Η μόνη πιθανή αλλαγή στην αρχικοποίηση του προγράμματος, είναι στην περίπτωση μιας διάταξη καρτελών. Η μέθοδος uiTabs::init() απλώς καλεί την uiBox::setRatio() και την uiBox::setCompLookAndFeel(LookAndFeel*) για κάθε στοιχείο της καρτέλας της.

Κατά τη μετάβαση σε όλες τις καρτέλες, ο αλγόριθμος παρακολουθεί το ελάχιστο μέγεθος του uiTabs που πρέπει να εμφανιστεί. Οι ελάχιστες διαστάσεις του είναι το μέγιστο πλάτος και το μέγιστο ύψος όλων των καρτελών του.

Εκεί, το δέντρο είναι χτισμένο, το συνολικό μέγεθος έχει ξεκινήσει, το μέγεθος του ορθογωνίου απεικόνισης και οι αναλογίες για όλα τα components, όλα τα uiBox και uiComponent δημιουργούνται.

4.3.5 Δυναμική διάταξη

Σε αυτό το σημείο που έχουμε φτάσει, έχουμε αναλύσει την διεπαφή του χρήστη που εμφανίζεται στο αρχικό της μέγεθος, αλλά πρέπει να προσαρμοστεί στην πιθανή αλλαγή μεγέθους του παραθύρου. Για να γίνει αυτό, χρησιμοποιούνται τα uiBoxes για τη διάταξη όλων των στοιχείων. Ένα στοιχείο uiBox έχει μια void arrangeComponents(Rectangle<int> functionRect), η οποία είναι το κύριο εργαλείο για την οργάνωση της διάταξης. Καλείται κάθε φορά που καλείται η συνάρτηση resized() του κύριου πλαισίου.

Στη συνάρτηση αυτή, το αρχικό ορθογώνιο που δίνεται ως όρισμα, το οποίο είναι βασικά το μέγεθος του παραθύρου, θα διαδοθεί σε όλα τα παιδιά uiBox και uiComponent, με αναδρομικό τρόπο [21].

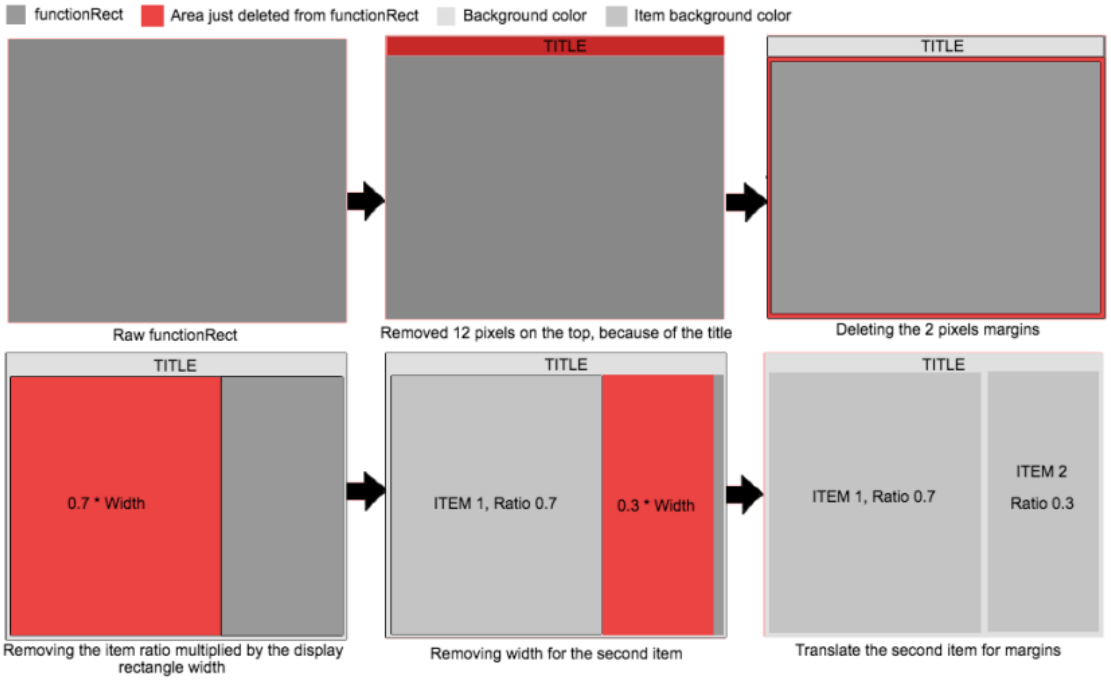
Στην αρχή, ελέγχει αν το όνομα χρειάζεται να εμφανιστεί, και καθώς δεν υπάρχουν παιδικά components δεν πρέπει να εμφανιστούν εκεί, κόβει 12 pixels από την κορυφή του functionRect, που δίνεται ως όρισμα.

Μετά από αυτό, τα περιθώρια πάνε σετ, οπότε 2 pixels είναι αριστερά, πάνω, δεξιά και κάτω. Με αυτόν τον τρόπο αποφεύγονται οι επικαλύψεις των στοιχείων. Μόλις γίνει, περνάει από όλα τα παιδιά, για να τους δώσει το σωστό χώρο που πρέπει να καταλαμβάνουν και τη σωστή θέση φυσικά.

Ο αλγόριθμος λειτουργεί με αυτόν τον τρόπο: αν το τρέχον πλαίσιο είναι κατακόρυφο, τότε πρέπει να δώσει στο παιδί του ένα κατακόρυφο τμήμα functionRect, και ένα οριζόντιο για ένα οριζόντιο πλαίσιο φυσικά. Το ποσό του κατακόρυφου ή οριζόντιου μεγέθους του παιδιού υπολογίζεται, ακόμα ανάλογα με την κατακόρυφη φύση του τρέχοντος πλαισίου. Το μέγεθος αυτό είναι το τρέχον ύψος ή πλάτος του πλαισίου, μείον τα περιθώρια, πολλαπλασιασμένο με τον οριζόντιο ή κατακόρυφο λόγο. Συγκεκριμένο παράδειγμα: Το τρέχον πλαίσιο είναι μια οριζόντια απεικόνιση και έχει 2 στοιχεία-παιδιά, εκ των οποίων το ένα έχει οριζόντια αναλογία 0,7 και το άλλο 0,3. Το μέγεθος εμφάνισης του πλαισίου είναι εδώ 1000x500 pixels, και το συνολικό του μέγεθος 1008x504 (2 στοιχεία και είναι ένα οριζόντιο πλαίσιο, οπότε $2 * (2 * \text{περιθώριο}) = 8$ στο πλάτος, και $2 * \text{περιθώριο} = 4$ στο ύψος).

Ας πούμε ότι το μέγεθος του παραθύρου σχεδόν διπλασιάστηκε και είναι τώρα 2008x1004 (αυθαίρετες απλές τιμές). Θα υπολογίσει ότι το πρώτο στοιχείο παίρνει χώρο πλάτους $0,7*(2008-2*4) = 0,7*2000 = 1400$ pixels και το δεύτερο $0,3*(2008-2*4) = 600$ pixels. Τα όρια του πρώτου στοιχείου θα είναι 1400x1000 και του δεύτερου 600x1000, με το ύψος να παραμένει το ίδιο, χωρίς φυσικά τα περιθώρια.

Συν τοις άλλης, για να παρακολουθούμε πού πρέπει να τοποθετήσουμε τα στοιχεία μας, η συνάρτησηRect κόβεται σιγά-σιγά κάθε φορά που σε ένα uiBaseComponent δίνεται ένα ορθογώνιο που πρέπει να εμφανιστεί [21]. Βασικά, κάθε ορθογώνιο που δίνεται στο παιδί αφαιρείται από το αρχικό functionRect, και αυτό μας επιτρέπει να παρακολουθούμε τις σωστές συντεταγμένες x και y που πρέπει να δώσουμε στο component παιδί, με το περιθώριο που προστίθεται. Αυτό γίνεται ξανά και ξανά για κάθε component παιδί, κόβοντας από το αριστερό ή το πάνω μέρος του rect boxRectangle<int> ανάλογα με τον προσανατολισμό του.



Σχήμα 4.3: Αναπαράσταση του αλγορίθμου διάταξης

4.3.6 Η κλάση MainContentComponent

Στην προσαρμοσμένη κλάση MainContentComponent, υπάρχει πληθώρα βιβλιοθηκών, οι οποίες είναι απαραίτητες για το πρόγραμμα. Περιλαμβάνονται κάποιες προαιρετικές επιλογές, για OSC, MIDI και για πολυφωνική λειτουργία, που εξαρτώνται από τις επιλογές μεταγλώττισης που θέτει ο χρήστης.

Η κλάση MainContentComponent είναι η κλάση Juce::Component που περιέχεται στο Viewport μας, και περιέχει η ίδια ένα αντικείμενο JuceGUI, δηλαδή μια υποκλάση της Juce::Component, μια GUI κλάση και MetaDataUI. Τα ελάχιστα πράγματα που πρέπει να γίνουν είναι:

```
addAndMakeVisible(juceGUI);
fDSP = new mydsp();
fDSP->buildUserInterface(&juceGUI);
recommendedSize = juceGUI.getSize();
setSize (recommendedSize.getWidth(), recommendedSize.getHeight());
```

```
setAudioChannels (fDSP->getNumInputs(), fDSP->getNumOutputs());
[...]

private:

        JuceGUI juceGUI;
```

Χρειάζεται μια απλή κλήση buildUserInterface, για να ορίσετε το μέγεθος του MainContentComponent και να ορίσετε την ποσότητα των καναλιών ήχου. Ακολουθώντας το ίδιο πνεύμα, υπάρχει προαιρετικός κώδικας σε περίπτωση που ένα MIDI, OSC ή πολυφωνικής λειτουργίας.

4.4 Ενσωμάτωση ήχου

Για να συνδεθεί με τον εξωτερικό κόσμο, ένας δεδομένος DSP πρέπει να συνδεθεί με έναν οδηγό ήχου και έναν ορισμό διεπαφής χρήστη. Το πλαίσιο JUCE περιέχει ήδη έναν αφηρημένο ηχητικό επίπεδο που συνδέεται με ένα σύνολο φυσικών οδηγών ήχου σε όλες τις πλατφόρμες ανάπτυξης. Οι προγραμματιστές του JUCE μπορούν να επιλέξουν να αναπτύξουν τον κώδικά τους ως αυτόνομες εφαρμογές ήχου ή plugins ήχου. Μία αυτόνομη εφαρμογή πρέπει να κατατάσσεται στην αφηρημένη κλάση AudioAppComponent και να υλοποιεί τις μεθόδους prepareToPlay, getNextAudioBlock και releaseResources:

- prepareToPlay καλείται ακριβώς πριν από τον ήχο η επεξεργασία ξεκινά με μια παράμετρο ρυθμού δειγματοληψίας. Το DSP ξεκινά με αυτήν την εντολή τιμής του ρυθμού δειγματοληψίας και ο αριθμός των καναλιών εισόδου/εξόδου προσαρμόζεται ενδεχομένως ώστε να ταιριάζει με τον τις δυνατότητες του χρησιμοποιούμενου φυσικού στρώματος (που μπορεί να έχει διαφορετικό αριθμό καναλιών εισόδου/εξόδου από τον DSP).

- Το getNextAudioBlock καλείται κάθε φορά που το υλικό ήχου χρειάζεται ένα νέο μπλοκ δεδομένων ήχου. Τα audio buffers που παρουσιάζονται ως τύπος δεδομένων AudioSourceChannelInfo ανακτώνται και προσαρμόζονται για να δοθούν στη μέθοδο υπολογισμού DSP.

- Η releaseResources καλείται όταν ολοκληρωθεί η επεξεργασία ήχου.

4.5 Επίλογος

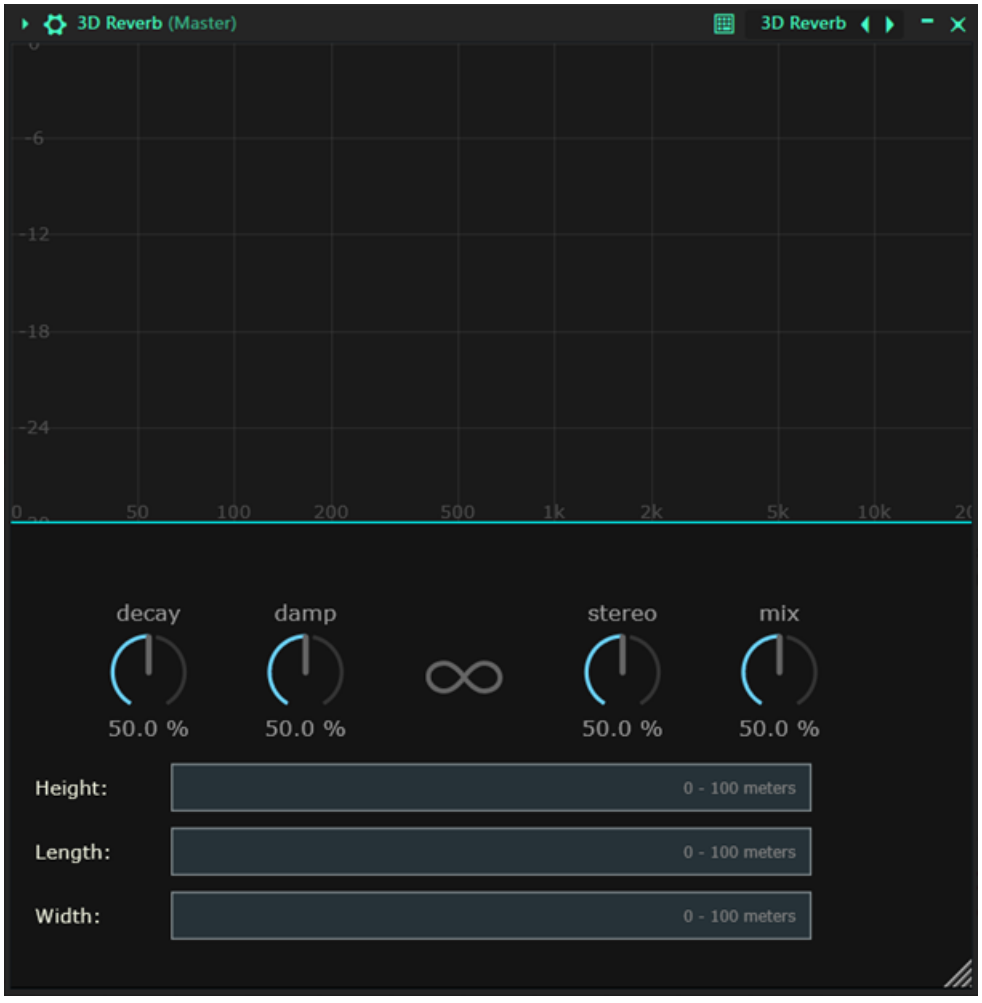
Η υλοποίηση της γλώσσας DSP ήχου είναι τώρα δυνατή με το JUCE, και θεωρητικά μπορεί να εξαχθεί σε κάθε πλατφόρμα που το JUCE υποστηρίζει. Έχει δοκιμαστεί σε Windows 11 στην παρόν εργασία, λειτουργεί σωστά, και έχει κοντινή απόδοση με τις ήδη διαθέσιμες επιλογές.

Το JUCE προσφέρει δύο τύπους " project ήχου ", αυτόνομες εφαρμογές (standalone) ή plug-in. Επί του παρόντος τα αρχεία αρχιτεκτονικής περιορίζονται για plug-ins.

Κεφάλαιο 5ο: Επεξήγηση κώδικα

Στην παρούσα ενότητα θα δούμε αναλυτικά πως όλη η παραπάνω θεωρία χρησιμοποιήθηκε για να δημιουργηθεί η εφαρμογή, καθώς και πως η θεωρία που αναφέρθηκε μεταφράστηκε σε κώδικα και κατ' επέκταση σε πρόγραμμα, μαζί με τα κομμάτια κώδικα που χρησιμοποιήθηκαν και φυσικά την επεξήγηση τους.

Η εφαρμογή αντήχησης στην τελική της μορφή, στο GUI εμφανίζεται όπως στο Σχήμα 5.1.



Σχήμα 5.1: Το plugin αντήχησης που υλοποιήθηκε στην παρούσα πτυχιακή εργασία

5.1 Πλήκτρα

Εκ πρώτης όψεως πάνω στην διεπαφή του χρήστη παρατηρούμε στο κάτω μέρος, 3 πλαίσια κειμένου (textboxes), 4 περιστροφικά κουμπιά (knobs) και 1 απλό κουμπί (button) με δικό του σχεδιασμένο εικονίδιο στο σχήμα του απείρου ή καλύτερα της λούπας.

5.1.1 Πλαίσια κειμένου

Η καινοτόμος ιδέα από δημιουργικής πλευράς της εργασίας, ήταν να κατασκευαστεί μια εφαρμογή αντήχησης η οποία θα μπορεί να προσημειώνει κανονικές αίθουσες, με βάση τα χαρακτηριστικά των διαστάσεων του Ύψους, του Μήκους και του Πλάτους των αιθουσών, και μετά από την προσομοίωση της αντήχησης, ο χρήστης να μπορεί να προσαρμόσει τις υπόλοιπες παραμέτρους για να κάνει πιο ακριβείς αναπαράσταση της αίθουσας.

Οπότε, στα πλαίσια κειμένου, ο χρήστης προσαρμόζει το Ύψος, το Μήκος και το Πλάτος μέσω τριών text boxes στα οποία ο χρήστης δίνει διαστάσεις στις μεταβλητές με τιμές από 0 μέχρι και 100 μέτρα.

5.1.1.1 Ύψος (Height)

Η παράμετρος Height επηρεάζει το ύψος της αίθουσας του reverb. Μεγαλύτερες τιμές αυξάνουν τη διάρκεια της αντήχησης και ενισχύουν τις χαμηλές συχνότητες.

- Η δημιουργία και σύνδεση του textbox με την παράμετρο Height

```
heightTextbox = std::make_unique<juce::TextEditor>("Height");
// Επιτρέπει μόνο αριθμούς και δεκαδικά
heightTextbox->setInputRestrictions(5, "0123456789.");
heightTextbox->onTextChange = [this]()
{
        auto value = heightTextbox->getText().getFloatValue();
        processor.getAPVTS().getParameter("height")->setValueNotifyingHost(value / 100.0f);
};
addAndMakeVisible(heightTextbox.get());
```

- Στο αρχείο PluginProcessor.cpp, η παράμετρος Height χρησιμοποιείται για να επηρεάσει το roomSize και τις χαμηλές συχνότητες

```
void PluginProcessor::updateReverbParams()
{
        const float currentHeight = height->get() * 0.01f;
        if (currentHeight != lastHeight)
        {
```

```
            // Συνδυασμός μεγέθους και ύψους
            params.roomSize = currentSize * currentHeight;
            // Ενίσχυση χαμηλών συχνοτήτων
            params.lowPassFrequency = 200.0f + (currentHeight * 800.0f);
            reverb.setParameters(params);
            // Ενημέρωση της τελευταίας τιμής
            lastHeight = currentHeight;
        }
}
```

5.1.1.2 Μήκος (Length)

Η παράμετρος Length επηρεάζει το μήκος της αίθουσας του reverb. Μεγαλύτερες τιμές αυξάνουν τη διάρκεια της αντήχησης, το στερεοφωνικό πλάτος και τις πρώιμες ανακλάσεις.

- Δημιουργία και σύνδεση Textbox για την παράμετρο Length:

```
lengthTextbox = std::make_unique<juce::TextEditor>("Length");
// Επιτρέπει μόνο αριθμούς και δεκαδικά
lengthTextbox->setInputRestrictions(5, "0123456789.");
lengthTextbox->onTextChange = [this]()
{
        auto value = lengthTextbox->getText().getFloatValue();
        processor.getAPVTS().getParameter("length")->setValueNotifyingHost(value / 100.0f);
};
addAndMakeVisible(lengthTextbox.get());
```

- Η παράμετρος Length επηρεάζει το μήκος της αίθουσας του reverb. Μεγαλύτερες τιμές αυξάνουν τη διάρκεια της αντήχησης, το στερεοφωνικό πλάτος και τις πρώιμες ανακλάσεις.

```
// Επέκταση του μεγέθους με βάση το μήκος
params.roomSize = currentSize * currentLength;
// Ενίσχυση στερεοφωνικού πλάτους
```

```
params.width = currentWidth * (1.0f + (currentLength * 0.5f));
// Προσθήκη καθυστερήσεων για early reflections
params.earlyReflectionsDelay = currentLength * 20.0f;
reverb.setParameters(params);
```

5.1.1.3 Πλάτος (Width)

Η παράμετρος Width επηρεάζει το στερεοφωνικό πλάτος του reverb. Μεγαλύτερες τιμές αυξάνουν την αίσθηση του χώρου, ενώ μικρότερες τιμές κάνουν τον ήχο πιο κεντραρισμένο.

- Δημιουργία και σύνδεση Textbox για την παράμετρο Width:

```
widthTextbox = std::make_unique<juce::TextEditor>("Width");
// Επιτρέπει μόνο αριθμούς και δεκαδικά
widthTextbox->setInputRestrictions(5, "0123456789.");
widthTextbox->onTextChange = [this]()
{
    auto value = widthTextbox->getText().getFloatValue();
    processor.getAPVTS().getParameter("width")->setValueNotifyingHost(value / 100.0f);
};
addAndMakeVisible(widthTextbox.get());
```

- Το Width επηρεάζει το στερεοφωνικό πλάτος, στο mid/side processing μέσα στο processBlock

```
void PluginProcessor::processBlock (juce::AudioBuffer<float>& buffer, juce::MidiBuffer& midiMessages)
{
    juce::ignoreUnused (midiMessages);
    juce::ScopedNoDenormals noDenormals;
    // Ανανέωση των παραμέτρων
    updateReverbParams();
    juce::dsp::AudioBlock<float> block (buffer);
```

```
        // Επεξεργασία μόνο για στερεοφωνικό σήμα
        if (block.getNumChannels() == 2)
        {
        const float currentWidth = width->get() * 0.01f;
        for (size_t i = 0; i < block.getNumSamples(); ++i)
        {
                auto left = block.getSample(0, i);
                auto right = block.getSample(1, i);
                // Mid/Side Processing
                auto mid = (left + right) * 0.5f;
                auto side = (left - right) * 0.5f * currentWidth;
                // Αριστερό κανάλι
                block.setSample(0, i, mid + side);
                // Δεξί κανάλι
                block.setSample(1, i, mid - side);
                }
        }
        juce::dsp::ProcessContextReplacing ctx (block);
        reverb.process (ctx);
}
```

- Στη μέθοδο updateReverbParams, η παράμετρος Width χρησιμοποιείται για να επηρεάσει το στερεοφωνικό πλάτος μέσω mid/side processing.

```
// Ανάκτηση της τρέχουσας τιμής
const float currentWidth = width->get() * 0.01f;
if (currentWidth != lastWidth)
{
        // Ενημέρωση του πλάτους
        params.width = currentWidth;
        reverb.setParameters(params);
        // Ενημέρωση της τελευταίας τιμής
        lastWidth = currentWidth;
}
```

5.1.2 Κουμπιά

Σε αυτήν την παράγραφο περιγράφεται ένα κουμπί για την αλλαγή μιας τιμής. Το κουμπί μπορεί να είναι οριζόντιο, κάθετο ή περιστροφικό και μπορεί προαιρετικά να έχει ένα πλαίσιο κειμένου μέσα του για να εμφανίζει μια επεξεργάσιμη εμφάνιση της τρέχουσας τιμής ώστε να έχει επίγνωση της ο χρήστης μέσα από το UI.

Στο αρχείο Dial.cpp υλοποιείται τη λειτουργικότητα ενός περιστροφικού κουμπιού (dial) που χρησιμοποιείται για την αλληλεπίδραση με τις παραμέτρους του ήχου, όπως αυτές που ορίζονται στο juce::RangedAudioParameter.

- Ο constructor αρχικοποιεί το Dial και το συνδέει με μια παράμετρο ήχου (juce::RangedAudioParameter).

```
// Ο paramAttachment συνδέει το Dial με την παράμετρο, ώστε να ενημερώνεται αυτόματα όταν αλλάζει η τιμή της
paramAttachment (audioParam, [&] (float v) { updateValue (v); }, um)
 // Ορίζονται τα χρώματα για το τόξο, τη βελόνα και το περίγραμμα
setColour (foregroundArcColourId, MyColours::blue);
setColour (backgroundArcColourId, MyColours::blackGrey);
// Το label εμφανίζει το όνομα της παραμέτρου
label.setText (audioParam.getName (8), juce::NotificationType::dontSendNotification);
// Το textBox επιτρέπει την εισαγωγή τιμών
textBox.onTextChange = [&] { ... };
```

- Η μέθοδος **resized** καθορίζει τη διάταξη του Dial, συμπεριλαμβανομένων του label και του textBox.

```
auto subAreaHeight = bounds.getHeight() / 4.0f;
// Το label τοποθετείται στο πάνω μέρος.
label.setBounds (bounds.removeFromTop (subAreaHeight).toNearestInt());
// Το textBox τοποθετείται στο κάτω μέρος.
textBox.setBounds (bounds.removeFromBottom (subAreaHeight).toNearestInt());
```

- Η μέθοδος paint σχεδιάζει το dial, συμπεριλαμβανομένων του τόξου και της βελόνας.

```
// Υπολογίζεται η γωνία του τόξου με βάση την τρέχουσα τιμή.
const auto radius = juce::jmin (mainArea.getWidth(), mainArea.getHeight()) * 0.5f;
```

```
// Σχεδιάζονται τα τόξα (backgroundArc και valueArc) και η βελόνα.
const auto toAngle = startAngle + value * (endAngle - startAngle);
```

- Το Dial υποστηρίζει αλληλεπίδραση με το ποντίκι για την αλλαγή της τιμής.

```
// Στη μέθοδο mouseDown ξεκινά μια "χειρονομία" για την αλλαγή της τιμής.
paramAttachment.beginGesture();
e.source.enableUnboundedMouseMovement (true);
// Στη μέθοδο mouseDrag η τιμή αλλάζει με βάση την κίνηση του ποντικιού.
auto diffY = (mousePosWhenLastDragged.y - e.position.y) * sensitivity;
value = juce::jlimit (0.0f, 1.0f, value + diffY);
// Στη μέθοδο mouseDoubleClick επαναφέρετε η προεπιλεγμένη τιμή
const auto defaultValue = audioParam.getDefaultValue();
value = defaultValue;
```

- Η updateValue **ε**νημερώνει την τιμή του Dial όταν αλλάζει η παράμετρος.

```
// Η τιμή μετατρέπεται στο εύρος [0, 1].
value = audioParam.convertTo0to1 (newValue);
// Το textBox ενημερώνεται με την τρέχουσα τιμή.
textBox.setText (audioParam.getCurrentValueAsText(), juce::NotificationType::dontSendNotification);
```

5.1.2.1 Συνολικός χρόνος απόσβεσης (Decay)

Η παράμετρος Decay επηρεάζει την διάρκεια αποσύνθεσης, δηλαδή, τον χρόνο που χρειάζεται ο ήχος να μειωθεί από το μέγιστο επίπεδο στο επίπεδο διατήρησης. Μεγαλύτερες τιμές Decay δημιουργούν την αίσθηση μεγαλύτερης αίθουσας.

- Η παράμετρος decay ορίζεται ως εξής:

```
layout.add (std::make_unique<juce::AudioParameterFloat> (
        // ID της
        juce::ParameterID { "decay", 1 },
        // Όνομα της παραμέτρου
        "Decay",
        // Εύρος τιμών (0.1s έως 10s)
        juce::NormalisableRange { 0.1f, 10.0f, 0.01f, 1.0f },
        // Αρχική τιμή
        1.0f,
        // Εμφάνιση ως ποσοστό
        percentageAttributes));
```

- Σύνδεση της παραμέτρου Decay

```
// Σύνδεση της παραμέτρου
castParameter ("decay", decay);
// Αρχικοποίηση της τελευταίας τιμής
lastDecay = decay->get();
```

- Στη μέθοδο updateReverbParams, η παράμετρος Decay χρησιμοποιείται για να επηρεάσει τη διάρκεια της αντήχησης

```
// Ανάκτηση της τρέχουσας τιμής
const float currentDecay = decay->get();
if (currentDecay != lastDecay)
{
        // Ενημέρωση της διάρκειας αποσύνθεσης
        params.decayTime = currentDecay;
        reverb.setParameters(params);
        // Ενημέρωση της τελευταίας τιμής
        lastDecay = currentDecay;
}
```

5.1.2.2 Απορρόφηση υψηλής συχνότητας (Damp)

Η παράμετρος damp χρησιμοποιείται για να ελέγξει το ποσοστό απόσβεσης (damping) του reverb, δηλαδή πόσο γρήγορα μειώνονται οι υψηλές συχνότητες κατά τη διάρκεια της αντήχησης. Ακολουθεί ανάλυση της υλοποίησης της παραμέτρου damp με παραδείγματα κώδικα και επεξηγήσεις.

- Η παράμετρος damp ορίζεται ως εξής

```
inline constexpr auto damp { "damp" };
```

- Προσθήκη της παραμέτρου damp

```
layout.add (std::make_unique<juce::AudioParameterFloat> (
        juce::ParameterID { ParamIDs::damp, 1 },
        ParamIDs::damp,
        // Το εύρος τιμών κυμαίνεται από 0.0f  με μέγιστη απόσβεση έως 1.0f με καμία απόσβεση.
        juce::NormalisableRange { 0.0f, 1.0f, 0.01f, 1.0f },
        // Αρχική τιμή μεταβλητής
        0.5f,
        percentageAttributes));
```

- Ενημέρωση των Παραμέτρων Reverb στη Μέθοδο updateReverbParams

```
// Ανάκτηση της τρέχουσας τιμής
const float currentDamp = damp->get();
if (currentDamp != lastDamp)
{
        // Ενημέρωση της απόσβεσης
        params.damping = currentDamp;
```

```
        reverb.setParameters(params);
        // Ενημέρωση της τελευταίας τιμής
        lastDamp = currentDamp;
}
```

5.1.2.3 Κουμπί παγώματος (Freeze button)

Το κουμπί παγώματος είναι ένα κουμπί το οποία διατηρεί την τρέχουσα κατάσταση της αντήχησης όταν το ενεργοποιήσει ο χρήστης, δημιουργώντας έναν "πάγωμα" στον ήχο την συγκεκριμένη χρονική στιγμή που ο χρήστης το πατάει.

- Ορίζεται η κλάση FreezeButton, η οποία κληρονομεί από την juce::Component

```
// Ο ParameterAttachment συνδέει το κουμπί με την παράμετρο freeze.
juce::RangedAudioParameter& audioParam;
juce::ParameterAttachment paramAttachment;
enum ColourIds
{
        // Χρώμα όταν είναι ενεργό
        onColourId,
        // Χρώμα όταν είναι ανενεργό
        offColourId,
        // Χρώμα όταν έχει focus
        focusColourId
};
```

- Ο constructor αρχικοποιεί το κουμπί και το συνδέει με την παράμετρο freeze.

```
// Ο paramAttachment ενημερώνει την κατάσταση του κουμπιού όταν αλλάζει η τιμή της παραμέτρου freeze.
FreezeButton::FreezeButton (juce::RangedAudioParameter& param, juce::UndoManager* um)
```

```
    : audioParam (param)
    , paramAttachment (audioParam, [&] (float v) { updateState (static_cast<bool> (v)); }, um)
{
    setWantsKeyboardFocus (true);
    setRepaintsOnMouseActivity (true);
    setColour (onColourId, MyColours::blue);
    setColour (offColourId, MyColours::midGrey);
    setColour (focusColourId, MyColours::midGrey.brighter (0.25f));
    // Το εικονίδιο φορτώνεται από ένα αρχείο SVG (FreezeIcon_svg).
    const auto svg = juce::Drawable::createFromImageData (BinaryData::FreezeIcon_svg, BinaryData::FreezeIcon_svgSize);
    jassert (svg != nullptr);
    if (svg != nullptr)
        iconPath = svg->getOutlineAsPath();
    paramAttachment.sendInitialUpdate();
}
```

- Η μέθοδος mouseDown αλλάζει την κατάσταση του κουμπιού όταν πατηθεί

```
void FreezeButton::mouseDown (const juce::MouseEvent& e)
{
    juce::ignoreUnused (e);
    // Η κατάσταση (state) αντιστρέφεται.
    paramAttachment.setValueAsCompleteGesture (! state);
    // Το εικονίδιο μικραίνει ελαφρώς για να δώσει οπτική ανατροφοδότηση.
    const auto centre = iconBounds.getCentre();
    iconPath.applyTransform (juce::AffineTransform::scale (0.95f, 0.95f, centre.x, centre.y));
}
```

5.1.2.4 Στέρεο (Stereo)

Η παράμετρος stereo χρησιμοποιείται για να ελέγξει το στερεοφωνικό πλάτος του reverb. Ακολουθεί ανάλυση της υλοποίησης της παραμέτρου stereo.

- Μεγαλύτερες τιμές αυξάνουν την αίσθηση του χώρου, ενώ μικρότερες τιμές κάνουν τον ήχο πιο "στενό"

```
layout.add (std::make_unique<juce::AudioParameterFloat> (
        juce::ParameterID { ParamIDs::stereo, 1 },
        ParamIDs:: stereo,
        // Εύρος τιμών
        juce::NormalisableRange { 0.0f, 100.0f, 0.01f, 1.0f },
        // Η αρχική τιμή είναι 50.0f που αντιστοιχεί σε ισορροπημένο πλάτος
        50.0f,
        percentageAttributes));
```

- Η σύνδεση της παραμέτρου στον Constructor του PluginProcessor

```
// Σύνδεση της παραμέτρου
castParameter (ParamIDs:: stereo, stereo);
// Αρχικοποίηση της τελευταίας τιμής
lastStereo = stereo ->get() * 0.01f;
```

- Η παράμετρος stereo χρησιμοποιείται για να ενημερώσει το πλάτος του reverb

```
// Ανάκτηση της τρέχουσας τιμής
const float currentStereo = stereo -> get() * 0.01f;
if (currentStereo != lastStereo)
{
        // Ενημέρωση του πλάτους
        params.stereo = currentStereo;
        reverb.setParameters(params);
```

```
        // Ενημέρωση της τελευταίας τιμής
        lastStereo = currentStereo;
}
```

5.1.2.5 Ανάμιξη (Mix)

Η λειτουργία Mix υλοποιείται μέσω της κλάσης DryWetMixer που παρέχεται από το JUCE DSP module. Αυτή η κλάση χρησιμοποιείται για την ανάμειξη ενός "dry" δηλαδή ενός μη επεξεργασμένου σήματος με ένα "wet" δηλαδή ενός επεξεργασμένου σήματος.

- Στο αρχείο PluginProcessor.cpp, η παράμετρος Mix ορίζεται ως εξής:

```
// Η παράμετρος Mix είναι τύπου AudioParameterFloat και έχει εύρος τιμών από 0.0f (μόνο dry) έως 100.0f (μόνο wet)
layout.add (std::make_unique<juce::AudioParameterFloat> (
        juce::ParameterID { ParamIDs::mix, 1 },
        ParamIDs::mix,
        juce::NormalisableRange { 0.0f, 100.0f, 0.01f, 1.0f },
        // Η αρχική τιμή της παραμέτρου είναι 50.0f, που αντιστοιχεί σε ισορροπημένη ανάμειξη (50% dry, 50% wet)
        50.0f,
        // Το percentageAttributes διαμορφώνει την εμφάνιση της τιμής ως ποσοστό (κυρίως για την εμφάνισή του στο GUI)
        percentageAttributes));
```

- Η παράμετρος Mix συνδέεται με το AudioProcessorValueTreeState για να μπορεί να ελέγχεται από το UI:

```
castParameter (ParamIDs::mix, mix);
// Η τιμή της παραμέτρου κανονικοποιείται (0.0 έως 1.0) για χρήση στον επεξεργαστή ήχου
lastMix = mix->get() * 0.01f;
```

- Η τιμή της παραμέτρου Mix χρησιμοποιείται για να ρυθμίσει τα επίπεδα του wet και dry σήματος:

```
// Το currentMix καθορίζει την αναλογία wet/dry
const float currentMix = mix->get() * 0.01f;
// Το wetLevel ορίζεται ίσο με το currentMix, ενώ το dryLevel είναι το συμπλήρωμα (1.0 - currentMix)
params.wetLevel = currentMix;
params.dryLevel = 1.0f - currentMix;
// Οι τιμές αυτές περνιούνται στον επεξεργαστή reverb μέσω της setParameters
reverb.setParameters(params);
```

- Η παράμετρος Mix μπορεί να συνδεθεί με ένα Dial στο UI:

```
// Το mixDial είναι ένας περιστροφικό κουμπί που ελέγχει την τιμή της παραμέτρου Mix
mixDial = std::make_unique<juce::Slider>();
mixDial->setSliderStyle(juce::Slider::Rotary);
mixDial->setRange(0.0, 100.0, 0.01);
mixDial->setValue(50.0);
// Το SliderAttachment συνδέει το mixDial με την παράμετρο Mix στο AudioProcessorValueTreeState για να μπορεί να ελέγχεται από το UI
mixDialAttachment = std::make_unique<juce::AudioProcessorValueTreeState::SliderAttachment>(
    processor.getAPVTS(), ParamIDs::mix, *mixDial);
```

5.2 Αναλυτής Φάσματος

Εκ δεύτερης όψεως παρατηρούμε έναν Αναλυτή Φάσματος (Spectrum Analyzer) των συχνοτήτων. Ένας αναλυτής φάσματος είναι ένα όργανο μέτρησης που χρησιμοποιείται για να απεικονίσει το φάσμα συχνοτήτων ενός ηλεκτρικού σήματος. Με απλά λόγια, δείχνει την ένταση των διαφόρων συχνοτήτων που συνθέτουν ένα σήμα. Έχει δύο άξονες, ο κάθετος άξονας απεικονίζει την ισχύει σε Ντεσιμπέλ (db) ενώ ο οριζόντιος άξονας τις συχνότητες του σήματος με μονάδα μέτρησης το Χέρτζ (Hz). Λόγω του ότι το ανθρώπινο αφτί ανταποκρίνεται σε ένα εύρος συχνοτήτων, από 20 Hz μέχρι και 20.000 Hz, αντίστοιχα και ο αναλυτής φάσματος απεικονίζει τις συχνότητες σε αυτό το εύρος. Στο Σχήμα 5.2 φαίνεται ο Αναλυτής Φάσματος της εφαρμογής που υλοποιήθηκε με κάποιο τυχαίο σήμα μιας λούπας κρουστών ως δείγμα για την γραφική αναπαράσταση.

Υπάρχουν διάφορες τεχνικές για την υλοποίηση ενός αναλυτή φάσματος, αλλά οι πιο κοινές βασίζονται σε δύο κύριες αρχές: Με την σάρωση φίλτρου (Swept-tuned filter) και με τον Μετασχηματισμό Fourier. Πως αυτές οι αρχές μεταφράζονται σε κώδικα περιγράφεται παρακάτω.

Σχήμα 5.2: Ο Αναλυτής Φάσματος της εφαρμογής με τυχαίο σήμα ως δείγμα.

5.2.1 Σάρωση Ζωνοπερατού Φίλτρου

Ένα στενό ζωνοπερατό φίλτρο (band-pass filter) σαρώνει ένα εύρος συχνοτήτων. Το φίλτρο επιτρέπει μόνο σε ένα μικρό εύρος συχνοτήτων να περάσει κάθε φορά. Η ισχύς του σήματος που περνάει από το φίλτρο μετριέται και απεικονίζεται σε σχέση με τη συχνότητα στην οποία είναι συντονισμένο το φίλτρο.

```
#include <juce_dsp/juce_dsp.h>

// Μέλος της κλάσης SpectrumAnalyzer

juce::dsp::IIR::Filter<float> bandPassFilter;

// Στον constructor

bandPassFilter = juce::dsp::IIR::Coefficients<float>::makeBandPass(sampleRate, centerFrequency, Q);

// Στη μέθοδο timerCallback:

for (int i = 0; i < numSamples; ++i)

{

        data[i] = bandPassFilter.processSample(data[i]);

}
```

5.2.2 Γρήγορος Μετασχηματισμός Fourier

Ο Γρήγορος Μετασχηματισμός Fourier (FFT) είναι ένα μαθηματικό εργαλείο που αναλύει ένα σήμα χρονικού πεδίου (δηλαδή ένα σήμα που μεταβάλλεται με τον χρόνο) στις συχνότητες που το συνθέτουν. Μας δείχνει ποιες συχνότητες υπάρχουν στο σήμα και ποια είναι η ένταση (ισχύς) κάθε συχνότητας. Στην περίπτωση ενός αναλυτή φάσματος, ο στόχος είναι να μετατρέψουμε ένα χρονικά μεταβαλλόμενο σήμα στην αναπαράστασή του στο πεδίο της συχνότητας.

```
void timerCallback() override
{
    if (nextFFTBlockReady)
    {
    const juce::SpinLock::ScopedLockType lock(mutex);
    // Εδώ εφαρμόζεται η συνάρτηση window Hann στα δεδομένα του buffer
    window.multiplyWithWindowingTable(fifo.data(), fftSize);
    // Εδώ εκτελείται η FFT στα δεδομένα που έχουν προετοιμαστεί. Το αποτέλεσμα αποθηκεύεται στο fftData.
    forwardFFT.performRealOnlyForwardTransform(fftData.data());
    // Εδώ υπολογίζονται τα μεγέθη (magnitudes) για κάθε συχνότητα. Το αποτέλεσμα αποθηκεύεται στο scopeData, το οποίο χρησιμοποιείται για την απεικόνιση του φάσματος.
    for (int i = 0; i < numPoints; ++i)
    {
        const float freq = freqPoints[i];
        const int binIndex = static_cast<int>(freq * fftSize / sampleRate);
        if (binIndex >= 0 && binIndex < fftSize / 2)
        {
            const float magnitude = std::sqrt(fftData[binIndex] * fftData[binIndex]);
            scopeData[i] = magnitude;
        }
    }
    nextFFTBlockReady = false;
    // Μετά την επεξεργασία των δεδομένων, καλείται η repaint() για να ενημερωθεί η γραφική απεικόνιση του φάσματος.
    repaint();
    }
}
```

Κεφάλαιο 6ο: Συμπεράσματα και προτάσεις βελτίωσης

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε επιτυχώς με την ανάπτυξη ενός plugin ψηφιακής αντήχησης (reverb plugin) για επεξεργασία ηχητικού σήματος σε πραγματικό χρόνο, χρησιμοποιώντας το JUCE framework και τη γλώσσα προγραμματισμού C++. Το έργο αυτό αποτέλεσε μια ολοκληρωμένη προσπάθεια που συνδύασε τη μελέτη των θεμελιωδών αρχών της επεξεργασίας ψηφιακού σήματος (όπως παρουσιάστηκαν στο Κεφάλαιο 1) και της ακουστικής των χώρων (Κεφάλαιο 2) με την πρακτική υλοποίηση ενός λειτουργικού εργαλείου επεξεργασίας ήχου.

Μέσω της υλοποίησης, αναδείχθηκε η δυνατότητα του JUCE framework στην ταυτόχρονη διαχείριση σύνθετων DSP αλγορίθμων και την ανάπτυξη λειτουργικής και αισθητικά αποδεκτής γραφικής διεπαφής χρήστη (GUI), όπως περιγράφηκε στα Κεφάλαια 3 και 4. Η ανάπτυξη του plugin περιελάμβανε την υλοποίηση ενός αλγορίθμου αντήχησης ικανού να προσομοιώνει ορθογώνιους χώρους, λαμβάνοντας ως είσοδο τις διαστάσεις τους, δηλαδή το ύψος, το μήκος και το πλάτος τους. Η ενσωμάτωση παραμέτρων όπως decay, stereo, mix, damp και η λειτουργία freeze προσέφερε ευελιξία στον έλεγχο των χαρακτηριστικών της αντήχησης, επιτρέποντας στον χρήστη να διαμορφώσει το αποτέλεσμα σύμφωνα με τις ανάγκες του.

Ιδιαίτερης σημασίας ήταν η ενσωμάτωση ενός αναλυτή φάσματος, ο οποίος, είτε μέσω σάρωσης ζωνοπερατού φίλτρου είτε μέσω Γρήγορου Μετασχηματισμού Fourier (FFT) όπως αναλύθηκε στο Κεφάλαιο 5, παρείχε ένα απαραίτητο οπτικό εργαλείο για την παρακολούθηση σε πραγματικό χρόνο της επίδρασης του plugin στο φάσμα του ηχητικού σήματος. Αυτή η οπτική ανατροφοδότηση είναι ζωτικής σημασίας για την κατανόηση και την ακριβή ρύθμιση των παραμέτρων αντήχησης, ιδίως της απόσβεσης (damp).

Συνολικά, η εργασία αυτή επιβεβαίωσε την επιτυχή σύνθεση θεωρητικής γνώσης και πρακτικής υλοποίησης. Το αναπτυχθέν plugin αποτελεί ένα λειτουργικό εργαλείο που μπορεί να χρησιμοποιηθεί σε περιβάλλοντα ψηφιακών σταθμών εργασίας ήχου (DAWs), αποδεικνύοντας την ικανότητα του συγγραφέα στην ανάπτυξη σύνθετων εφαρμογών επεξεργασίας ήχου.

Η παρούσα υλοποίηση θέτει τις βάσεις για την ανάπτυξη ενός ολοκληρωμένου reverb plugin. Για μελλοντική επέκταση και βελτίωση του έργου, προτείνονται οι ακόλουθες κατευθύνσεις:

- Επέκταση του Αλγορίθμου Αντήχησης: Ο υπάρχων αλγόριθμος θα μπορούσε να επεκταθεί ώστε να προσομοιώνει πιο σύνθετες γεωμετρίες χώρων πέραν των ορθογώνιων ή/και να ενσωματώσει μοντέλα απορρόφησης ήχου από διαφορετικά υλικά επιφανειών. Αυτό θα προσέδιδε μεγαλύτερο ρεαλισμό και ποικιλία στα παραγόμενα αποτελέσματα αντήχησης.

- Προηγμένες Παράμετροι Ελέγχου: Προσθήκη πιο λεπτομερών παραμέτρων ελέγχου, όπως:

    - Ξεχωριστός έλεγχος για τον χρόνο προ-καθυστέρησης (pre-delay).
    - Πιο granular έλεγχος των πρώιμων ανακλάσεων (early reflections), ίσως με δυνατότητα οπτικής επεξεργασίας του μοτίβου τους.
    - Έλεγχος της διάχυσης (diffusion) της αντήχησης.
    - Πιο προηγμένος έλεγχος της απόσβεσης συχνοτήτων (π.χ. multi-band dampening) ή προσομοίωση διαφορετικών τύπων φίλτρων απόσβεσης.
    - Επιπλέον "τύποι" αντήχησης (π.χ. προσομοίωση plate, spring ή non-linear reverbs).

- Αναβάθμιση του Αναλυτή Φάσματος: Ο αναλυτής φάσματος θα μπορούσε να βελτιωθεί με περισσότερες επιλογές απεικόνισης (π.χ. μέσος όρος, κορυφές, waterfall display) ή/και με δυνατότητα εστίασης (zoom) σε συγκεκριμένες περιοχές συχνοτήτων για πιο λεπτομερή ανάλυση.

- Βελτιστοποίηση Απόδοσης: Παρότι η παρούσα υλοποίηση είναι λειτουργική, η περαιτέρω βελτιστοποίηση του κώδικα για μείωση της κατανάλωσης CPU είναι πάντα επιθυμητή, ειδικά σε περιβάλλοντα όπου μπορεί να χρησιμοποιηθούν πολλαπλά instances του plugin.

- Ενίσχυση της Γραφικής Διεπαφής (GUI): Βελτιώσεις στην αισθητική και την εργονομία της GUI, προσθήκη οπτικής ανατροφοδότησης για παραμέτρους όπως η καμπύλη απόσβεσης συχνοτήτων, καθώς και η δυνατότητα αλλαγής μεγέθους του παραθύρου του plugin.

- Υποστήριξη Presets και Automation: Εξασφάλιση πλήρους υποστήριξης για αποθήκευση και φόρτωση προεπιλογών (presets) από τον χρήστη, καθώς και πλήρης συμβατότητα με το automation των παραμέτρων από το host DAW.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Jurgens, R. K., ed. Digital Consumer Electronics Handbook, Chapter 2. New York: McGraw-Hill, 1997.

[2] Kirk, R. and Hunt, A. Digital Sound Processing for Music and Multimedia, Chapter 1. Massachusetts: Focal Press, 1999.

[3] Stearns, S. D. and Hush, D. R. Digital Signal Analysis , pp. 44–52. Englewood Cliffs, NJ: Prentice-Hal, 1990.

[4] Pirkle, Will C. *Designing Software Synthesizer Plug-ins in C++: For RackAFX, VST3, and Audio Units*. Routledge, 2014.

[5] Limberis, A. and Bryan, J. An architecture for a multiple digital-signal processor based music synthesizer with dynamic voice allocation. Journal of the Audio Engineering Society, Preprint No. 3699, 1993.

[6] Yoemun Yun, "Audio Analysis & Creating Reverberation Plug-ins In Digital Audio Engineering", International Journal of Multimedia and Ubiquitous Engineering, Vol.8, No.6, (2013), pp.201-208

[7] H. T. Kim, T. H. Kim and J. S. Park, "A Real Time Singing Voice Removal System Using DSP and Multichannel Audio Interface", International Journal of Multimedia and Ubiquitous Engineering, vol. 7, no. 2, 2012.

[8] D. E. Hall, "Musical Acoustics", Third Edition, Books/Cole, 2002.

[9] K. A. Hoover, "An Appreciation of Acoustics", Cavanaugh Tocci Publishing, 1991.

[10] G. D. White, "The Audio Dictionary", Second Edition, University of Washington, 2005.

[11] Y. M. Kim, "The Audio Control using TDM (Tree-structural Distributed-terminal Mixing) in the Webcast", International Journal of Multimedia and Ubiquitous Engineering, vol. 1, no. 4, 2006.

[12] Apple Computer, Inc, "Soundtrack Pro User Manual", 2005. [Online]. Available: https://device.report/m/1dc7d35f02f3322a03b7b7a8d39121bdbe587793a7ffc6d95f47f8b3bea3744a.pdf.

[13] F. Rumsey and T. McCormick, "Sound and Recording", Fourth Edition, Focal Press, 2002.

[14] P. Dutta, D. Bhattacharyya and T. Kim, "Data Hiding in Audio Signal: A Review", International Journal of Database Theory and Application, vol. 2, no. 2, 2009.

[15] C. J. Chun, Y. G. Kim, J. Y. Yang and H. K. Kim, "Real-Time Conversion of Stereo Audio to 5.1 Channel Audio for Providing Realistic Sounds", International Journal of Signal Processing, Image Processing and Pattern Recognition, vol. 2, no. 4, 2009.

[16] Cycling '74, "MAX/MSP Manual", (2009). [Online]. Available: www.cycling74.com

[17] J. Chappell, "Digital Home Recording", Backbeat Books, 2003.

[18] Vincent Goudard and Remy Muller, "Real-time audio plugin architectures", June 2, 2003

[19] JUCE online documentation https://www.juce.com/doc/classes

[20] JUCE "Tutorial: Advanced Rectangle techniques"

https://www.juce.com/doc/tutorial_rectangle_advanced

[21] J. Storer "Developing Graphical User Interfaces with JUCE", JUCE Summit, 2015.