ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σχεδίαση και Υλοποίηση Ψηφιακού Audio Compressor Plug-in με χρήση C++ και JUCE Framework»

Του φοιτητή:
Τσιουραλίδη Αλέξανδρου
Αρ. Μητρώου: 2020179

Επιβλέπων:
Ρήγας Κωτσάκης
Αναπληρωτής Καθηγητής

28-05-2026

Τίτλος Δ.Ε. :
Σχεδίαση και Υλοποίηση Ψηφιακού Audio Compressor Plug-in με χρήση C++ και JUCE Framework
Κωδικός Δ.Ε. 25333
Ονοματεπώνυμο φοιτητή:
Αλέξανδρος Τσιουραλίδης
Ονοματεπώνυμο εισηγητή:
Ρήγας Κωτσάκης
Ημερομηνία ανάληψης Δ.Ε. 23-10-2025
Ημερομηνία περάτωσης Δ.Ε. 28-05-2026

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 μουσική υπήρξε για εμένα πηγή έμπνευσης και δημιουργικότητας από μικρή ηλικία. Το ενδιαφέρον μου για τον ήχο και η ενασχόληση με την πληροφορική με οδήγησαν στη μελέτη της ψηφιακής επεξεργασίας σήματος και στην ανάπτυξη ενός συστήματος δυναμικής συμπίεσης ήχου. Μέσα από αυτήν την εργασία απέκτησα πολύτιμες γνώσεις σχετικά με τη θεωρία της δυναμικής συμπίεσης και τον τρόπο λειτουργίας των compressors, κατανοώντας όχι μόνο τις βασικές αρχές αλλά και τη σημασία τους στη μουσική παραγωγή. Παράλληλα, η ανάπτυξη του συστήματος μου επέτρεψε να βελτιώσω τις δεξιότητές μου στην C++ και να εξοικειωθώ με διάφορες τεχνικές υλοποίησης. Η εμπειρία αυτή συνδύασε τη θεωρητική γνώση με την πρακτική εφαρμογή, επιτρέποντάς μου να κατανοήσω σε βάθος τόσο τα επιστημονικά όσο και τα τεχνολογικά χαρακτηριστικά ενός σύγχρονου συστήματος επεξεργασίας ήχου.

Περίληψη

Η παρούσα εργασία πραγματεύεται τη σχεδίαση και υλοποίηση ενός ψηφιακού συστήματος δυναμικής συμπίεσης ήχου, το οποίο αναπτύχθηκε ως plugin για ψηφιακούς σταθμούς επεξεργασίας ήχου με υποστήριξη των μορφότυπων VST3 και Audio Unit και ως standalone εφαρμογή. Η εργασία δίνει έμφαση τόσο στη δυναμική επεξεργασία του ηχητικού σήματος όσο και στην οπτική αναπαράσταση της λειτουργίας του μέσω ενός προσαρμοσμένου γραφικού περιβάλλοντος χρήστη.

Αρχικά παρουσιάζεται η ιστορική αναδρομή των compressors από τα αναλογικά συστήματα έως τα σύγχρονα ψηφιακά περιβάλλοντα, καθώς και βασικές αρχές ψηφιακής επεξεργασίας σήματος, όπως η δειγματοληψία, η κβαντοποίηση και η ανάλυση RMS και Peak. Παράλληλα, εξετάζονται οι απαιτήσεις πραγματικού χρόνου που χαρακτηρίζουν τις εφαρμογές επεξεργασίας ήχου.

Στη συνέχεια περιγράφεται η υλοποίηση της δυναμικής επεξεργασίας του συστήματος, το οποίο υποστηρίζει τόσο single-band όσο και multi-band συμπίεση. Η διαδικασία βασίζεται στον υπολογισμό της στάθμης του σήματος και στην εφαρμογή μηχανισμών συμπίεσης μέσω των παραμέτρων threshold, ratio, attack και release, ενώ παράλληλα παρέχονται ρυθμίσεις input και output gain για τον έλεγχο της στάθμης. Στη multi-band λειτουργία, το σήμα διαχωρίζεται σε επιμέρους περιοχές συχνοτήτων, οι οποίες αντιστοιχούν στη χαμηλή, τη μεσαία και την υψηλή ζώνη, ώστε να καθίσταται δυνατή η ανεξάρτητη ρύθμιση της δυναμικής συμπεριφοράς κάθε περιοχής.

Ιδιαίτερη έμφαση δίνεται στο γραφικό περιβάλλον, το οποίο περιλαμβάνει VU meter, LED meters, αναλυτή φάσματος και signal display για την παρακολούθηση του ηχητικού σήματος σε πραγματικό χρόνο. Επιπλέον, παρέχονται λειτουργίες επιλογής προκαθορισμένων ρυθμίσεων και εναλλαγής μεταξύ διαφορετικών τρόπων επεξεργασίας, ενώ στη standalone λειτουργία ενσωματώνονται δυνατότητες φόρτωσης, αναπαραγωγής και εξαγωγής αρχείων ήχου.

Τέλος, η λειτουργικότητα και η ακρίβεια του συστήματος αξιολογούνται μέσω δοκιμών σε διαφορετικά είδη ηχητικών σημάτων και συνθήκες επεξεργασίας, καθώς και μέσω αξιολόγησης από χρήστες με εμπειρία στην επεξεργασία ήχου, επιβεβαιώνοντας την ορθότητα και τη χρηστικότητα της υλοποίησης. Η εργασία ολοκληρώνεται με την εξαγωγή συμπερασμάτων και προτάσεων για μελλοντική επέκταση και βελτίωση του συστήματος.

«Design and Implementation of a Digital Audio Compressor Plug-in using C++ and the JUCE Framework»

«Alexandros Tsiouralidis»

Abstract

This thesis focuses on the design and implementation of a digital dynamic audio compression system, developed both as a plugin for modern digital audio workstations with support for the VST3 and Audio Unit formats and as a standalone application. The work emphasizes both the dynamic processing of audio signals and the visual representation of the system's operation through a customized graphical user interface.

Initially, the thesis presents a historical overview of compressors, from analog systems to modern digital environments, as well as fundamental principles of digital signal processing such as sampling, quantization, and RMS and Peak analysis. In addition, the real-time requirements of audio applications are examined.

The implementation of the system's dynamic processing is then presented in detail. The developed system supports both single-band and multi-band compression. Its operation is based on signal level detection and the application of compression through the parameters of threshold, ratio, attack, and release, while input and output gain controls are provided for level adjustment. In multi-band mode, the signal is divided into distinct frequency regions corresponding to the low, mid, and high bands, thus enabling independent control of the dynamic behavior of each region.

Particular emphasis is placed on the graphical user interface, which incorporates a VU meter, LED meters, a spectrum analyzer, and a signal display for real-time audio monitoring. In addition, the system provides preset selection and switching between different processing modes, while in standalone mode it includes audio file loading, playback and export capabilities.

Finally, the functionality and accuracy of the system are evaluated through testing with different types of audio signals and processing conditions, as well as through evaluation by users with experience in audio processing, confirming the validity and usability of the implementation. The thesis concludes with remarks and suggestions for future extension and improvement of the system.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
DAWs	Digital Audio Workstations
VST	Virtual Studio Technology
AU	Audio Unit
DSP	Digital Signal Processing
RMS	Root Mean Square
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
FET	Field Effect Transistor
VU Meter	Volume Unit Meter
APVTS	AudioProcessorValueTreeState
UI	User Interface
GUI	Graphical User Interface
Hz	Hertz
dB	Decibel
dBFS	Decibels relative to Full Scale
WAV	Waveform Audio File Format
MP3	MPEG Audio Layer III
MIDI	Musical Instrument Digital Interface
LED	Light Emitting Diode
PNG	Portable Network Graphics

Κεφάλαιο 1ο: Εισαγωγή στον Ήχο και στην Ψηφιακή Τεχνολογία

1.1 Εισαγωγή

Ο ήχος αποτελεί θεμελιώδες στοιχείο της ανθρώπινης επικοινωνίας, της μουσικής δημιουργίας και των σύγχρονων οπτικοακουστικών εφαρμογών. Η ανάγκη για αξιόπιστη και αποτελεσματική διαχείριση ηχητικού υλικού έχει αυξηθεί σημαντικά τα τελευταία χρόνια, κυρίως λόγω της ανάπτυξης της τεχνολογίας και της ευρείας διάδοσης των ψηφιακών μέσων στην παραγωγή, επεξεργασία και αναπαραγωγή ήχου. Στο πλαίσιο της σύγχρονης ψηφιακής τεχνολογίας, η επεξεργασία ήχου πραγματοποιείται κατά κύριο λόγο μέσω λογισμικού, αξιοποιώντας τεχνικές ψηφιακής επεξεργασίας σήματος.

Σημαντικό ρόλο σε αυτή τη διαδικασία διαδραματίζουν οι επεξεργαστές ήχου, όπως οι ισοσταθμιστές, οι χωρικοί επεξεργαστές και οι δυναμικοί επεξεργαστές. Μεταξύ των δυναμικών επεξεργαστών, οι compressors αποτελούν ένα από τα βασικότερα εργαλεία στη διαδικασία της μίξης και της παραγωγής ήχου, καθώς ελέγχουν το δυναμικό εύρος ενός σήματος.

Η ραγδαία ανάπτυξη των ψηφιακών σταθμών επεξεργασίας ήχου και των audio plugins έχει καταστήσει εφικτή την υλοποίηση πολύπλοκων αλγορίθμων επεξεργασίας ήχου, προσφέροντας αυξημένη ευελιξία και δυνατότητες παραμετροποίησης στον χρήστη. Στο πλαίσιο αυτό, η ανάπτυξη εξειδικευμένων εργαλείων επεξεργασίας ήχου αποτελεί αντικείμενο έντονου ερευνητικού και τεχνολογικού ενδιαφέροντος.

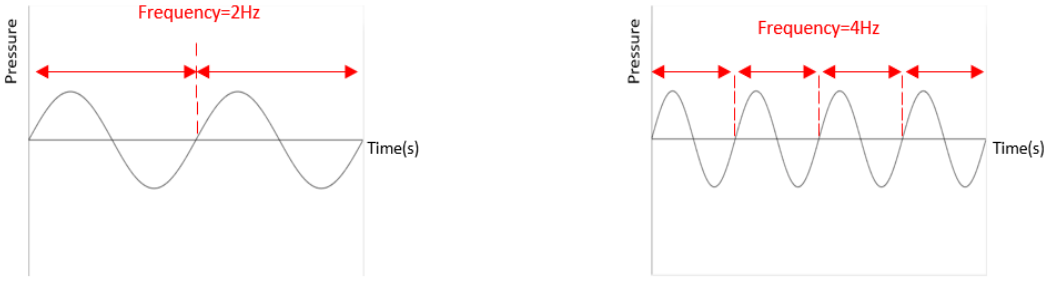
1.2 Ο Ήχος και τα Βασικά του Χαρακτηριστικά

Ο ήχος είναι ένα φυσικό φαινόμενο που προκύπτει από τη διάδοση μηχανικών κυμάτων σε ένα ελαστικό μέσο. Παράγεται όταν μια πηγή πάλλεται, προκαλώντας περιοδικές μεταβολές στην πίεση του αέρα. Οι μεταβολές αυτές φτάνουν στο ανθρώπινο αυτί, όπου μετατρέπονται σε ηλεκτρικά σήματα και ερμηνεύονται από τον εγκέφαλο ως αντιληπτός ήχος.

Στο πλαίσιο της τεχνολογίας, ο ήχος μοντελοποιείται ως χρονικά μεταβαλλόμενο σήμα, δηλαδή ως μαθηματική συνάρτηση της πίεσης ως προς τον χρόνο. Η μετατροπή από ακουστικό κύμα σε ηλεκτρικό σήμα πραγματοποιείται μέσω ηλεκτροακουστικών μετατροπέων, όπως το μικρόφωνο, το οποίο μετασχηματίζει τις διακυμάνσεις της ηχητικής πίεσης σε αντίστοιχες μεταβολές ηλεκτρικής τάσης. Με τον τρόπο αυτό, το φυσικό φαινόμενο αποκτά μορφή κατάλληλη για ανάλυση, αποθήκευση και επεξεργασία από ηλεκτρονικά και ψηφιακά συστήματα.

Τα βασικά χαρακτηριστικά ενός ηχητικού σήματος περιλαμβάνουν:

- **Συχνότητα (Frequency):** Η συχνότητα καθορίζει το ύψος (pitch) του ήχου και μετριέται σε Hertz (Hz). Αντιστοιχεί στον αριθμό των ταλαντώσεων του σήματος ανά δευτερόλεπτο, με τις υψηλότερες συχνότητες να γίνονται αντιληπτές ως ψηλότεροι ήχοι και τις χαμηλότερες ως βαθύτεροι.



Μικρότερη Συχνότητα
Χαμηλότερο Pitch Ήχου

Μεγαλύτερη Συχνότητα
Υψηλότερο Pitch Ήχου

Σχήμα 1.1: Κυματομορφές με διαφορετικές συχνότητες

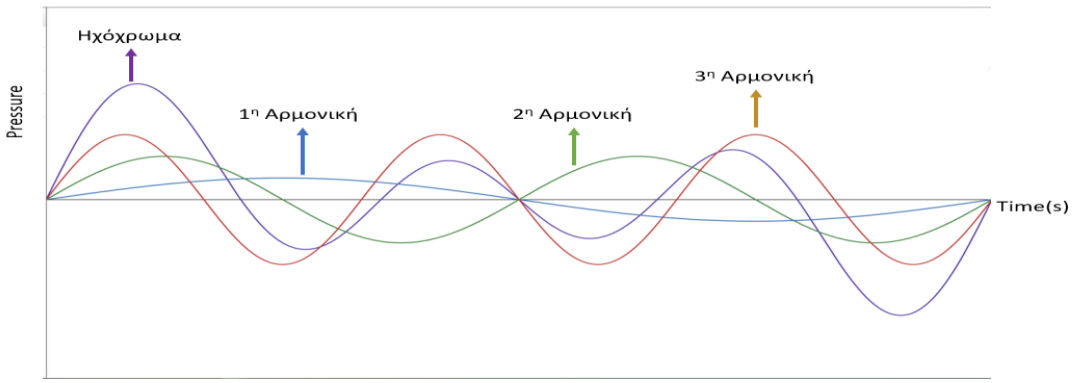
- **Πλάτος (Amplitude):** Το πλάτος σχετίζεται άμεσα με τη στάθμη και την ένταση του ήχου. Αποτελεί ένα από τα σημαντικότερα μεγέθη στην επεξεργασία ήχου, καθώς καθορίζει την ενεργειακή συμπεριφορά του σήματος [1]. Μεγάλες μεταβολές στο πλάτος οδηγούν σε αντίστοιχες διακυμάνσεις έντασης, οι οποίες επηρεάζουν τη δυναμική ισορροπία του ήχου.

Μικρότερο Πλάτος
Μικρότερη Ένταση Ήχου

Μεγαλύτερο Πλάτος
Μεγαλύτερη Ένταση Ήχου

Σχήμα 1.2: Κυματομορφές με διαφορετικό πλάτος

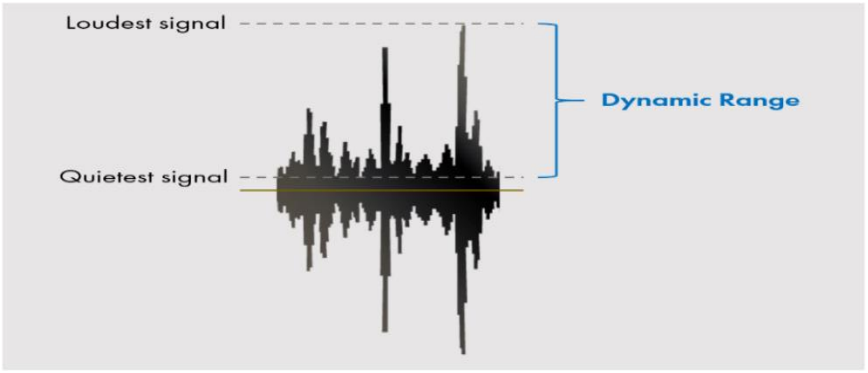
- **Ηχόχρωμα (Timbre):** Το ηχόχρωμα προσδίδει τον ιδιαίτερο χαρακτήρα ενός ήχου και εξαρτάται από την παρουσία και τη σχετική ένταση των αρμονικών συνιστωσών του σήματος. Διαφορετικές κατανομές αρμονικών οδηγούν σε διαφορετική αντιληπτή ποιότητα, ακόμη και όταν η συχνότητα και το πλάτος παραμένουν ίδια [2].



Σχήμα 1.3: Κυματομορφή με αρμονικές συνιστώσες

Στην πράξη, τα περισσότερα ηχητικά σήματα δεν παρουσιάζουν σταθερή ένταση στον χρόνο. Αντίθετα, χαρακτηρίζονται από δυναμική συμπεριφορά, με συνεχείς διακυμάνσεις της στάθμης, απότομες κορυφές και μεταβατικά σημεία.

Η δυναμική περιοχή (dynamic range) ενός σήματος περιγράφει το εύρος των επιπέδων έντασης και ορίζεται ως η διαφορά μεταξύ της χαμηλότερης και της υψηλότερης στάθμης του. Η δυναμική περιοχή εκφράζεται συνήθως σε λογαριθμική κλίμακα decibel, η οποία χρησιμοποιείται ευρέως στην ανάλυση και επεξεργασία ηχητικών σημάτων [3].



Σχήμα 1.4: Δυναμική Περιοχή [3]

Μεγάλο δυναμικό εύρος μπορεί να δημιουργήσει δυσκολίες κατά την αναπαραγωγή ή τη μίξη, καθώς οι υψηλές κορυφώσεις ενδέχεται να υπερβούν τα όρια του συστήματος, προκαλώντας clipping και ανεπιθύμητη ακουστική παραμόρφωση. Για τον λόγο αυτό, ο έλεγχος της δυναμικής συμπεριφοράς ενός σήματος αποτελεί βασικό στόχο της επεξεργασίας ήχου.

Η κατανόηση των παραπάνω ιδιοτήτων αποτελεί κρίσιμο θεωρητικό υπόβαθρο για την ανάπτυξη και χρήση δυναμικών επεξεργαστών, όπου απαιτείται ακριβής έλεγχος της στάθμης και της συμπεριφοράς του σήματος στον χρόνο.

1.3 Από τον Αναλογικό στον Ψηφιακό Ήχο

Παρότι ο ήχος στη φύση αποτελεί ένα συνεχές αναλογικό φαινόμενο, τα σύγχρονα συστήματα βασίζονται κυρίως στην ψηφιακή του αναπαράσταση. Η μετάβαση από το αναλογικό στο ψηφιακό πεδίο επιτρέπει την αξιοποίηση της υπολογιστικής ισχύος και καθιστά δυνατή την εφαρμογή προηγμένων τεχνικών επεξεργασίας ήχου οι οποίες δεν είναι πρακτικά υλοποιήσιμες με αποκλειστικά αναλογικά μέσα.

Η ψηφιοποίηση του ήχου πραγματοποιείται μέσω δύο βασικών διαδικασιών όπου είναι η δειγματοληψία (sampling) και η κβαντοποίηση (quantization). Με τη δειγματοληψία, το συνεχές σήμα αποτυπώνεται σε διακριτές χρονικές στιγμές, ενώ με την κβαντοποίηση κάθε δείγμα στρογγυλοποιείται στην πλησιέστερη διαθέσιμη τιμή ενός συνόλου διακριτών αριθμητικών τιμών. Μέσω αυτών των διαδικασιών καθίσταται δυνατή η ψηφιακή μοντελοποίηση μουσικών οργάνων και ηχητικών εφέ, προσφέροντας σημαντικά πλεονεκτήματα όπως φορητότητα και αυτοματοποιημένο έλεγχο της επεξεργασίας [4].

Η ψηφιακή επεξεργασία σήματος αποτελεί σήμερα τον βασικό μηχανισμό επεξεργασίας του ήχου. Μέσω DSP καθίσταται δυνατός ο έλεγχος παραμέτρων όπως η στάθμη, το φασματικό περιεχόμενο και η δυναμική συμπεριφορά ενός σήματος, προσφέροντας σταθερότητα και ακρίβεια. Οι δυνατότητες αυτές έχουν καταστήσει το ψηφιακό πεδίο κυρίαρχο στην ανάπτυξη εργαλείων επεξεργασίας ήχου, όπως οι δυναμικοί επεξεργαστές.

1.4 Επίλογος

Στο κεφάλαιο αυτό αναλύθηκαν οι βασικές αρχές του ήχου, τα χαρακτηριστικά των ηχητικών σημάτων και η μετάβαση από τον αναλογικό στον ψηφιακό ήχο. Η κατανόηση της συχνότητας, του πλάτους, του ηχοχρώματος και της δυναμικής περιοχής παρέχει τη θεωρητική βάση για την ανάπτυξη επεξεργαστών ήχου. Επιπλέον, η παρουσίαση των διαδικασιών ψηφιοποίησης και της ψηφιακής επεξεργασίας σήματος τεκμηριώνει την αναγκαιότητα χρήσης λογισμικού και DSP. Οι έννοιες αυτές διαμορφώνουν το θεωρητικό πλαίσιο για την κατανόηση των τεχνολογιών δυναμικής επεξεργασίας, όπως οι αναλογικοί και οι ψηφιακοί compressors.

Κεφάλαιο 2ο: Ιστορική Αναδρομή & Τεχνολογίες Συμπίεσης

2.1 Εισαγωγή

Η δυναμική επεξεργασία του ήχου και ειδικότερα η συμπίεση αποτελεί θεμελιώδες εργαλείο στην ηχοληψία, την ραδιοτηλεοπτική μετάδοση και την μουσική παραγωγή. Η ανάγκη ελέγχου του δυναμικού εύρους των ηχητικών σημάτων οδήγησε, ήδη από τα μέσα του 20ού αιώνα, στην ανάπτυξη αναλογικών συσκευών συμπίεσης, οι οποίες έθεσαν τις βάσεις για τους σύγχρονους ψηφιακούς compressors.

2.2 Δυναμικοί Επεξεργαστές Ήχου

Οι δυναμικοί επεξεργαστές αποτελούν κατηγορία εργαλείων επεξεργασίας ήχου που στοχεύουν στον έλεγχο της δυναμικής συμπεριφοράς ενός σήματος, δηλαδή των μεταβολών της στάθμης του στον χρόνο. Σε αντίθεση με άλλους τύπους επεξεργαστών, όπως οι ισοσταθμιστές που επηρεάζουν την φασματική περιοχή, οι δυναμικοί επεξεργαστές δρουν κυρίως στο πλάτος του σήματος, προκειμένου να περιορίσουν ή να τροποποιήσουν τη δυναμική περιοχή, να αποφευχθεί το clipping και να εξασφαλιστεί ομοιόμορφη ηχητική απόδοση.

Οι δυναμικοί επεξεργαστές διακρίνονται σε τέσσερις βασικές κατηγορίες:

- **Compressor:** Μειώνει το δυναμικό εύρος ενός σήματος περιορίζοντας τα επίπεδα που υπερβαίνουν ένα προκαθορισμένο όριο (threshold) [5]. Με αυτόν τον τρόπο, οι υψηλές κορυφώσεις συμπιέζονται , ενώ μέσω του make-up gain μπορεί να αυξηθεί η συνολική ένταση του σήματος ενισχύοντας έμμεσα και τα σήματα με χαμηλότερη στάθμη.

- **Limiter:** Αποτελεί ειδική περίπτωση του compressor με πολύ υψηλό λόγο συμπίεσης σχεδιασμένος για να αποτρέπει την υπέρβαση ενός μέγιστου επιπέδου στάθμης [5]. Στην πράξη λειτουργεί ως οροφή (ceiling) για το σήμα, εξασφαλίζοντας ότι οι κορυφώσεις δεν προκαλούν clipping ή παραμόρφωσεις.

  Οι limiters χρησιμοποιούνται κυρίως:
  - Στο mastering για τον έλεγχο της κορυφής των τελικών μίξεων.
  - Σε live sound για την προστασία του συστήματος από υπερφορτώσεις.

- **Expander:** Λειτουργεί αντίστροφα από τον compressor. Αυξάνει το δυναμικό εύρος ενισχύοντας τη διαφορά μεταξύ χαμηλών και υψηλών επιπέδων στάθμης [6]. Μπορεί να χρησιμοποιηθεί για να μειώσει θόρυβο ή ανεπιθύμητες χαμηλές στάθμες, αυξάνοντας την αντίθεση μεταξύ σιωπής και χρήσιμου σήματος.

  Υπάρχουν δύο βασικοί τύποι:
  - **Downward expander:** Μειώνει τα επίπεδα κάτω από ένα threshold.
  - **Upward expander:** Αυξάνει τα επίπεδα πάνω από ένα threshold.

- **Noise gate:** Δυναμικός επεξεργαστής που μειώνει δραστικά σήματα που βρίσκονται κάτω από ένα προκαθορισμένο επίπεδο. Χρησιμοποιείται κυρίως για τη μείωση ανεπιθύμητου θορύβου σε ηχογραφήσεις ή live setups.

2.3 Η Αναλογική Εποχή

Οι πρώτοι δυναμικοί επεξεργαστές αναπτύχθηκαν για να εξυπηρετήσουν πρακτικές ανάγκες ελέγχου στάθμης σε ραδιοφωνικές και τηλεοπτικές μεταδόσεις. Οι πρώτες συσκευές βασίζονταν αποκλειστικά σε αναλογικά ηλεκτρονικά κυκλώματα και παρουσίαζαν χαρακτηριστική ηχητική συμπεριφορά, η οποία μέχρι σήμερα θεωρείται επιθυμητή. Στα αρχικά τους στάδια, οι συσκευές αυτές λειτουργούσαν κυρίως ως broadcast limiters. Με την εξέλιξη των αναλογικών τεχνολογιών, εμφανίστηκαν οι πρώτοι compressors, οι οποίοι επέτρεπαν πιο ομαλή και μουσική συμπίεση, ελέγχοντας τη δυναμική των σημάτων χωρίς να αλλοιώνεται η ηχητική τους ποιότητα.

2.3.1 Broadcast Limiters

Κατά τη δεκαετία του 1950, οι broadcast limiters χρησιμοποιήθηκαν ευρέως σε ραδιοφωνικούς σταθμούς με σκοπό την αποφυγή υπερφόρτωσης του σήματος μετάδοσης. Οι συσκευές αυτές λειτουργούσαν κυρίως ως limiters, περιορίζοντας απότομα τις κορυφές του σήματος, ώστε να διασφαλίζεται σταθερή στάθμη εξόδου και συμμόρφωση με τις τεχνικές προδιαγραφές εκπομπής [7].

Σχήμα 2.1: SA39B Limiter [8]

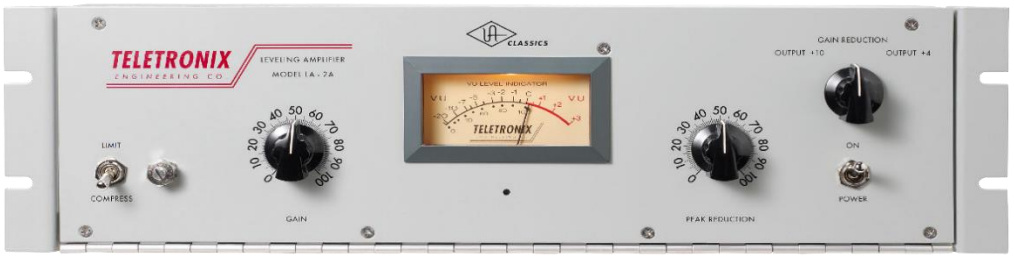
2.3.2 Tube/Vari-Mu Compressors: Fairchild 670

Οι tube ή Vari-Mu compressors, όπως ο Fairchild 670, εμφανίστηκαν τη δεκαετία του 1950 και βασίζονται σε ηλεκτρονικές συσκευές vacuum tubes για τον έλεγχο της μείωσης του κέρδους [9]. Χαρακτηρίζονται από αργό attack και release, μουσικό κορεσμό και «ζεστό» ήχο, προσφέροντας ομαλή συμπίεση που θεωρείται ιδιαίτερα φυσική για φωνές και πλήκτρα. Ο Fairchild 670 αποτέλεσε σημείο αναφοράς στα στούντιο mastering, χάρη στην ικανότητά του να διατηρεί την τονική ισορροπία του σήματος.

Σχήμα 2.2: Fairchild 670 Compressor [10]

2.3.3 Optical Compressors: Teletronix LA-2A

Οι optical compressors βασίζονται σε φωτοηλεκτρικά στοιχεία για τον έλεγχο της ενίσχυσης. Ένα από τα πιο χαρακτηριστικά παραδείγματα αποτελεί ο Teletronix LA-2A, ο οποίος παρουσιάστηκε την δεκαετία του 1960. Η λειτουργία του στηρίζεται σε συνδυασμό πηγής φωτός (λάμπα) και φωτοαντίστασης (LDR), προσδίδοντας ομαλή συμπίεση, με αργό attack και εξαρτώμενο release. Ο ήχος του LA-2A θεωρείται ιδιαίτερα φυσικός και χρησιμοποιείται συχνά σε φωνές και μπάσο.



Σχήμα 2.3: Teletronix LA-2A Compressor [11]

2.3.4 FET Compressors: UREI 1176

Ο UREI 1176 αποτελεί έναν από τους πιο γνωστούς compressors τύπου FET (Field Effect Transistor) και κυκλοφόρησε το 1967. Η λειτουργία του βασίζεται στη χρήση τρανζίστορ πεδίου FET ως στοιχείο μεταβλητής ενίσχυσης, επιτρέποντας εξαιρετικά γρήγορο έλεγχο της στάθμης του σήματος. Σε αντίθεση με τους optical compressors, οι FET compressors χαρακτηρίζονται από πολύ γρήγορο attack και release, καθιστώντας τους κατάλληλους για δυναμικά και κρουστικά σήματα. Ο 1176 έγινε δημοφιλής λόγω της επιθετικής συμπίεσης και του χαρακτηριστικού χρωματισμού που προσδίδει στο σήμα [12].

Σχήμα 2.4: Urei 1176 Compressor [13]

2.3.5 VCA Compressors: dbx 160

Οι compressors τύπου VCA (Voltage Controlled Amplifier) εμφανίστηκαν τη δεκαετία του 1970 και πρόσφεραν μεγάλη ακρίβεια και σταθερότητα στον έλεγχο της δυναμικής. Η λειτουργία τους βασίζεται σε ενισχυτές ελεγχόμενους από τάση, όπου η μεταβολή της στάθμης του σήματος καθορίζεται από ένα control voltage που προκύπτει από κύκλωμα ανίχνευσης. Ο dbx 160 αποτελεί χαρακτηριστικό παράδειγμα για την προβλέψιμη συμπεριφορά και την καθαρή συμπίεση. Οι VCA compressors χρησιμοποιούνται ευρέως σε εφαρμογές όπου απαιτείται αυστηρός και ακριβής έλεγχος στάθμης, όπως σε drums, ομάδες καναλιών και live εφαρμογές.

Σχήμα 2.5: Dbx 160 Compressor [14]

2.4 Η Μετάβαση στην Ψηφιακή Εποχή

Η εξέλιξη της ψηφιακής τεχνολογίας επέφερε ριζικές αλλαγές στον τρόπο επεξεργασίας και διαχείρισης του ήχου. Η σταδιακή αντικατάσταση των αναλογικών μέσων εγγραφής από ψηφιακά συστήματα, καθώς και η ευρεία υιοθέτηση των υπολογιστών στην μουσική παραγωγή, δημιούργησαν τις συνθήκες για τη μετάβαση των δυναμικών επεξεργαστών από το hardware στο software.

Οι πρώτοι ψηφιακοί compressors αναπτύχθηκαν με κύριο στόχο την αναπαραγωγή της βασικής λειτουργικότητας των αναλογικών συσκευών, που είναι ο έλεγχος του δυναμικού εύρους. Υλοποιούνται μέσω ψηφιακής επεξεργασίας σήματος σε περιβάλλον λογισμικού ή σε εξειδικευμένες ψηφιακές συσκευές. Σε αντίθεση με τα αναλογικά κυκλώματα, οι ψηφιακές υλοποιήσεις βασίζονται σε μαθηματικά μοντέλα και αριθμητικούς υπολογισμούς, επιτρέποντας μεγαλύτερη ακρίβεια στον έλεγχο της συμπεριφοράς της συμπίεσης.

Τα ψηφιακά συστήματα επεξεργασίας παρουσιάζουν σημαντικά πλεονεκτήματα σε σύγκριση με τα αναλογικά κυκλώματα. Καταρχάς, προσφέρουν υψηλό βαθμό ευελιξίας, καθώς οι λειτουργίες τους μπορούν να τροποποιηθούν μέσω λογισμικού, χωρίς να απαιτούνται παρεμβάσεις στο υλικό επίπεδο. Επιπλέον, διακρίνονται για την αξιοπιστία τους, αφού η απόδοσή τους παραμένει σταθερή με την πάροδο του χρόνου και δεν επηρεάζεται από φθορές εξαρτημάτων, όπως συμβαίνει στα αναλογικά συστήματα. Παράλληλα, η λειτουργία τους χαρακτηρίζεται από υψηλή επαναληψιμότητα, καθώς η επεξεργασία ενός σήματος μπορεί να πραγματοποιηθεί με την ίδια συμπεριφορά κάθε φορά σε σχέση με τα αναλογικά συστήματα όπου έχουν ανοχές [15].

Κατά τη διάρκεια αυτής της μετάβασης, παρατηρήθηκε έντονο ενδιαφέρον για τη διατήρηση του ηχητικού χαρακτήρα των κλασικών αναλογικών compressors. Αυτό οδήγησε στην ανάπτυξη τεχνικών μοντελοποίησης, μέσω των οποίων επιχειρείται η προσομοίωση της μη γραμμικής συμπεριφοράς, του κορεσμού και της δυναμικής απόκρισης των ιστορικών συσκευών που αναλύθηκαν. Έτσι, οι ψηφιακοί compressors δεν περιορίστηκαν μόνο σε καθαρές και διαφανείς υλοποιήσεις, αλλά απέκτησαν και χαρακτήρα, προσεγγίζοντας την αισθητική των αναλογικών προτύπων.

Η μετάβαση στην ψηφιακή εποχή επέτρεψε επίσης την ενσωμάτωση επιπλέον λειτουργιών που ήταν αδύνατο να υλοποιηθούν σε αναλογικό επίπεδο, όπως η ακριβής οπτική απεικόνιση στάθμης, η αποθήκευση και ανάκληση ρυθμίσεων (presets), καθώς και η αυτοματοποίηση παραμέτρων σε πραγματικό χρόνο. Οι δυνατότητες αυτές κατέστησαν τους ψηφιακούς compressors αναπόσπαστο εργαλείο στα σύγχρονα περιβάλλοντα παραγωγής ήχου, ανοίγοντας τον δρόμο για την εξέλιξη των audio plugins και των DAWs.

2.5 Digital Audio Workstations

Τα Digital Audio Workstations (DAWs) αποτελούν ολοκληρωμένα ψηφιακά συστήματα παραγωγής και επεξεργασίας ήχου, τα οποία λειτουργούν σε περιβάλλον υπολογιστή και επιτρέπουν την εγγραφή, την επεξεργασία, την σύνθεση, την μίξη και την διανομή ηχητικού υλικού. Η δημιουργία των DAWs προέκυψε από την ανάγκη αντικατάστασης των αναλογικών στούντιο ηχογράφησης, τα οποία βασίζονταν σε μαγνητοταινίες, αναλογικές κονσόλες μίξης και εξωτερικές μονάδες επεξεργασίας.

Η βασική αρχή λειτουργίας ενός DAW στηρίζεται στην ψηφιακή αναπαράσταση του ήχου, επιτρέποντας την ακριβή αριθμητική επεξεργασία του σήματος και την επαναληψιμότητα των αποτελεσμάτων. Το DAW λειτουργεί ως host περιβάλλον, εντός του οποίου φορτώνονται και εκτελούνται τα audio plugins [16]. Ένα DAW συνδυάζει λειτουργίες πολυκάναλης εγγραφής, επεξεργασίας κυματομορφών, δρομολόγησης σήματος (routing), καθώς και υποστήριξη εικονικών οργάνων και επεξεργαστών σήματος μέσω plugins.

Τα DAWs μπορούν να διακριθούν βάση δύο βασικές κατηγορίες [17]:

- **DSP-based DAWs:**
  Στα συστήματα αυτά, η επεξεργασία του ηχητικού σήματος πραγματοποιείται μέσω εξειδικευμένων καρτών ψηφιακής επεξεργασίας σήματος, οι οποίες λειτουργούν ανεξάρτητα από τον κεντρικό επεξεργαστή του υπολογιστή.
- **Host-based DAWs:**
  Στην περίπτωση αυτή, όλες οι λειτουργίες της επεξεργασίας ήχου εκτελούνται από τον κεντρικό επεξεργαστή του υπολογιστή. Δεν απαιτείται εξειδικευμένο DSP hardware, γεγονός που μειώνει το κόστος και αυξάνει την ευελιξία του συστήματος.

Στην πράξη, η πλειονότητα των σύγχρονων DAWs ανήκει στην κατηγορία των host-based συστημάτων, αξιοποιώντας την υπολογιστική ισχύ των σύγχρονων υπολογιστών για την εκτέλεση όλων των λειτουργιών επεξεργασίας. Ενδεικτικά παραδείγματα αποτελούν τα FL Studio, Ableton Live, Logic Pro και Cubase.

Αντίστοιχα, στην κατηγορία των DSP-based συστημάτων που βασίζονται σε εξειδικευμένο hardware χαρακτηριστικό παράδειγμα αποτελεί το Pro Tools HD, το οποίο αξιοποιεί κάρτες DSP (HDX) για την επεξεργασία ήχου σε επαγγελματικά στούντιο, προσφέροντας σταθερό latency και υψηλή αξιοπιστία σε απαιτητικά περιβάλλοντα παραγωγής.

2.6 Digital Plugins

Η καθιέρωση των ψηφιακών compressors ως audio plugins αποτέλεσε καθοριστικό βήμα στην εξέλιξη της δυναμικής επεξεργασίας ήχου. Με την ευρεία διάδοση των DAWs, τα plugins ανέλαβαν τον ρόλο των κλασσικών εξωτερικών συσκευών, επιτρέποντας την ενσωμάτωση πολύπλοκων αλγορίθμων επεξεργασίας σε περιβάλλοντα λογισμικού. Η μετάβαση αυτή οδήγησε στη δημιουργία ενός μεγάλου αριθμού ψηφιακών compressors, οι οποίοι καλύπτουν τόσο ανάγκες τεχνικής ακρίβειας όσο και ηχητικού χαρακτήρα.

Στον χώρο των ψηφιακών compressors, ιδιαίτερη αναφορά αξίζουν plugins όπως ο Waves Renaissance Compressor και ο FabFilter Pro-C, τα οποία έχουν σχεδιαστεί με έμφαση στην ακρίβεια, την σταθερότητα και τον λεπτομερή έλεγχο των παραμέτρων. Τα συγκεκριμένα plugins χρησιμοποιούνται ευρέως σε εφαρμογές μίξης και mastering, καθώς προσφέρουν καθαρή δυναμική επεξεργασία χωρίς ανεπιθύμητο χρωματισμό.

Σχήμα 2.6: Fab Filter Pro C2 Compressor [18]

Παράλληλα, αναπτύχθηκαν ψηφιακά plugins που βασίζονται στη μοντελοποίηση αναλογικών συσκευών, αξιοποιώντας τεχνικές ψηφιακής προσομοίωσης για την αναπαραγωγή της μη γραμμικής συμπεριφοράς των ιστορικών compressors. Χαρακτηριστικά παραδείγματα αποτελούν οι UAD 1176 και UAD LA-2A, οι οποίοι προσομοιώνουν αντίστοιχα τους κλασικούς FET και optical compressors. Παρόμοια φιλοσοφία ακολουθούν plugins όπως ο Waves CLA-76 και ο Waves CLA-2A, τα οποία χρησιμοποιούνται ευρέως για τη χαρακτηριστική τους απόκριση και τον αναλογικό χαρακτήρα που προσδίδουν στο σήμα.

Σχήμα 2.7: CLA-76 Compressor [19]

Ιδιαίτερη εξέλιξη σημειώθηκε και στον τομέα της οπτικής ανατροφοδότησης και της ανάλυσης του σήματος. Σύγχρονα plugins, όπως το iZotope Neutron και το TDR Kotelnikov, ενσωματώνουν προηγμένες μετρήσεις gain reduction, RMS και Peak ανάλυση, καθώς και γραφικές απεικονίσεις της δυναμικής συμπεριφοράς. Οι δυνατότητες αυτές επιτρέπουν στον χρήστη να κατανοεί σε βάθος την επίδραση της συμπίεσης και να πραγματοποιεί ακριβείς ρυθμίσεις σε πραγματικό χρόνο.

Σχήμα 2.8: Izotope Neutron Plugin [20]

Συνολικά, τα digital plugins έχουν μετατρέψει τη δυναμική επεξεργασία ήχου σε μία πλήρως ελεγχόμενη και οπτικά υποστηριζόμενη διαδικασία. Η αλγοριθμική ακρίβεια και η δυνατότητα επεξεργασίας σε πραγματικό χρόνο καθιστά τους σύγχρονους ψηφιακούς compressors βασικό εργαλείο στην επαγγελματική προσέγγιση της επεξεργασίας ήχου.

2.7 Audio Plugin Formats

Η ενσωμάτωση εργαλείων επεξεργασίας ήχου σε περιβάλλοντα DAW καθίσταται δυνατή μέσω τυποποιημένων μορφότυπων audio plugins. Τα plugin formats αποτελούν πρότυπα λογισμικού που καθορίζουν τον τρόπο με τον οποίο ένα plugin επικοινωνεί με το σύστημα host, διαχειρίζεται τα δεδομένα ήχου και αλληλεπιδρά με τις παραμέτρους του περιβάλλοντος εργασίας. Μέσω αυτών των προτύπων διασφαλίζεται η συμβατότητα και η ομαλή λειτουργία μεταξύ διαφορετικών εφαρμογών και λειτουργικών συστημάτων.

Η ύπαρξη πολλαπλών μορφότυπων οφείλεται σε τεχνολογικούς, αρχιτεκτονικούς και εμπορικούς παράγοντες. Κάθε format ορίζει συγκεκριμένο τρόπο διαχείρισης παραμέτρων, συγχρονισμού με το project του DAW, καθώς και αξιοποίησης των διαθέσιμων υπολογιστικών πόρων.

Από τα πιο γνωστά plugin formats είναι το VST (Virtual Studio Technology), το Audio Unit (AU) και το AAX (Avid Audio eXtension). Κάθε ένα από αυτά έχει αναπτυχθεί για να εξυπηρετεί συγκεκριμένες πλατφόρμες και περιβάλλοντα παραγωγής, προσφέροντας διαφορετικά επίπεδα ενσωμάτωσης και βελτιστοποίησης.

Η δημιουργία του VST από τη Steinberg αποτέλεσε καθοριστικό βήμα για την ενσωμάτωση audio plugins σε περιβάλλοντα DAW, διευρύνοντας σημαντικά τις δυνατότητες ανάπτυξης και αξιοποίησης ψηφιακών εφέ. Το πρότυπο εξελίχθηκε στη συνέχεια σε νεότερες εκδόσεις, όπως το VST3, ενώ παράλληλα αναπτύχθηκαν και άλλα plugin formats [21].

Η υποστήριξη πολλαπλών audio plugin formats επιτρέπει στους κατασκευαστές λογισμικού να διανέμουν τα προϊόντα τους σε ευρύ φάσμα συστημάτων, ενισχύοντας την διαλειτουργικότητα και την προσβασιμότητα. Κατ' αυτόν τον τρόπο, τα plugin formats αποτελούν θεμελιώδες στοιχείο της σύγχρονης ψηφιακής επεξεργασίας ήχου, υποστηρίζοντας τόσο την επαγγελματική παραγωγή όσο και την ερευνητική και εκπαιδευτική δραστηριότητα.

Πίνακας 2.1: Πίνακας Μορφότυπων

Μορφότυπο	Εταιρεία	Λειτουργικό Σύστημα
VST (Virtual Studio Technology)	Steinberg	Windows, MacOs
AU (Audio Units)	Apple	MacOs
AAX (Avid Audio Extension)	Avid Technology	Windows, MacOs
RTAS (Real Time AudioSuite)	Avid Technology	Windows, MacOs

2.8 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκε η ιστορική εξέλιξη των δυναμικών επεξεργαστών ήχου, από τις πρώτες αναλογικές συσκευές έως τα σύγχρονα ψηφιακά plugins. Μέσα από αυτή την αναδρομή, αναλύθηκαν οι βασικές κατηγορίες τους και τα χαρακτηριστικά τους, ενώ παράλληλα επισημάνθηκε η σημαντική μετάβαση από το hardware στο software. Η μετάβαση αυτή προσέφερε νέες δυνατότητες και προώθησε την ενσωμάτωση με τα DAWs. Επιπλέον, η αναφορά στα διάφορα audio plugin formats ανέδειξε τη σημασία της συμβατότητας σε διαφορετικά συστήματα παραγωγής.

Το θεωρητικό και τεχνολογικό υπόβαθρο που αναπτύχθηκε στο παρόν κεφάλαιο θέτει τις βάσεις για την ανάλυση των αρχών της ψηφιακής επεξεργασίας σήματος, οι οποίες αποτελούν τον πυρήνα για την κατανόηση και τον σχεδιασμό ψηφιακών audio compressors.

Κεφάλαιο 3ο: Θεμελιώδεις Αρχές Ψηφιακής Επεξεργασίας Σήματος

3.1 Εισαγωγή στην Ψηφιακή Επεξεργασία Σήματος

Η ψηφιακή επεξεργασία σήματος αποτελεί έναν κλάδο της μηχανικής και της επιστήμης υπολογιστών που ασχολείται με την ανάλυση, τροποποίηση και βελτίωση σημάτων σε ψηφιακή μορφή. Σήματα μπορούν να είναι ήχος, εικόνα, βίντεο, βιοϊατρικά δεδομένα, τηλεπικοινωνιακά σήματα και πολλά άλλα. Η επεξεργασία τους σε ψηφιακή μορφή επιτρέπει μεγαλύτερη ακρίβεια, ευελιξία και δυνατότητα αυτοματισμού σε σχέση με τα αναλογικά συστήματα.

Η ψηφιακή επεξεργασία σήματος περιλαμβάνει πολλές βασικές διαδικασίες, όπως δειγματοληψία, κβαντοποίηση, φιλτράρισμα, μετασχηματισμούς Fourier, στατιστική ανάλυση και ανίχνευση χαρακτηριστικών. Κάθε διαδικασία έχει σκοπό την εξαγωγή ή τροποποίηση πληροφοριών που περιέχονται στο σήμα με αποδοτικό και ακριβή τρόπο.

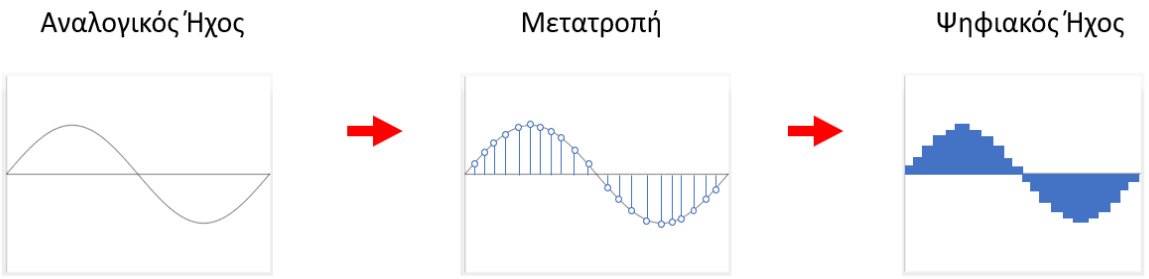
Η DSP εφαρμόζεται σε πλήθος τομέων, όπως:

- **Τηλεπικοινωνίες:** Επεξεργασία και συμπίεση σήματος, διόρθωση σφαλμάτων, φιλτράρισμα.
- **Μουσική:** Φίλτρα, compression, επεξεργασία ηχογραφήσεων.
- **Ιατρική:** Ανάλυση ηλεκτροκαρδιογραφήματος (ECG), επεξεργασία σήματος από αισθητήρες.
- **Ρομποτική και αυτοματισμοί:** Ανάλυση και φιλτράρισμα αισθητήρων, αναγνώριση προτύπων [22].

Η εξέλιξη των ψηφιακών συστημάτων και των αλγορίθμων DSP έχει καταστήσει δυνατή την άμεση και ακριβή επεξεργασία σημάτων σε πραγματικό χρόνο, καθιστώντας την ψηφιακή επεξεργασία θεμελιώδες εργαλείο σε ένα ευρύ φάσμα επιστημονικών και τεχνολογικών πεδίων.

3.2 Αναλογικά και Ψηφιακά Σήματα στον Ήχο

Ένα αναλογικό σήμα μεταβάλλεται συνεχώς ως προς τον χρόνο και μπορεί, θεωρητικά, να λάβει άπειρες τιμές μέσα σε ένα δεδομένο εύρος. Στον ήχο, το αναλογικό σήμα αντιστοιχεί στην συνεχή μεταβολή της πίεσης του αέρα, η οποία καταγράφεται και μεταφέρεται μέσω ηλεκτρικών κυκλωμάτων. Αντίθετα, ένα ψηφιακό σήμα αποτελεί διακριτή αναπαράσταση του αναλογικού, όπου το συνεχές κύμα μετατρέπεται σε ακολουθία αριθμητικών τιμών μέσω της διαδικασίας της δειγματοληψίας και της κβαντοποίησης.



Σχήμα 3.1: Μετατροπή αναλογικού σήματος σε ψηφιακό

Στα ψηφιακά συστήματα ήχου, κάθε δείγμα αντιπροσωπεύει το πλάτος του σήματος σε μια συγκεκριμένη χρονική στιγμή. Η ακολουθία αυτών των δειγμάτων επιτρέπει την αποθήκευση, την αντιγραφή και την επεξεργασία του ήχου με υψηλή ακρίβεια και επαναληψιμότητα. Σε αντίθεση με τα αναλογικά συστήματα, όπου ο θόρυβος και οι ανοχές των κυκλωμάτων συσσωρεύονται σε κάθε στάδιο επεξεργασίας, τα ψηφιακά συστήματα διατηρούν την ποιότητα του σήματος ανεξάρτητα από τον αριθμό των επεξεργαστικών βημάτων.

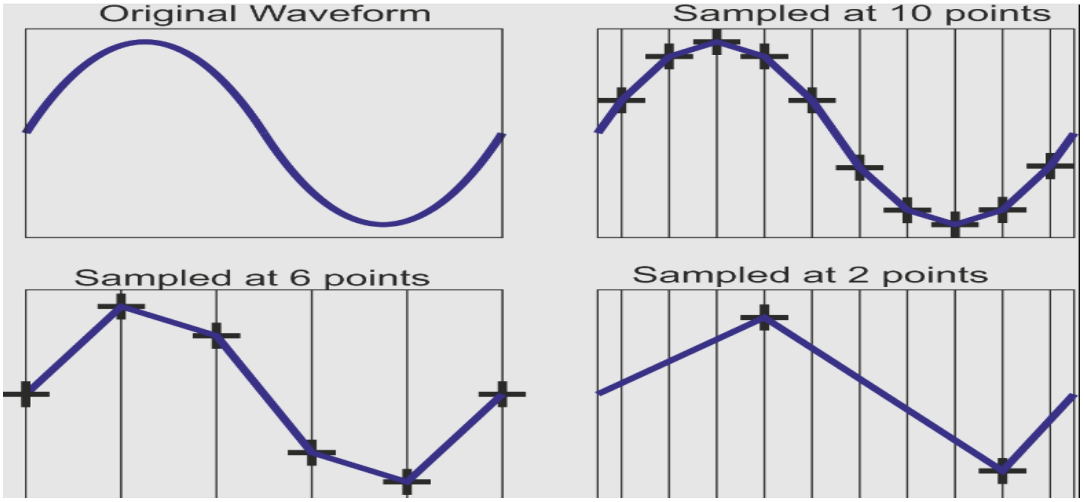
Ένα από τα σημαντικότερα πλεονεκτήματα των ψηφιακών σημάτων είναι η αυξημένη ανθεκτικότητά τους στον θόρυβο και στις παραμορφώσεις που προκύπτουν από εξωτερικούς παράγοντες. Επιπλέον, η ψηφιακή επεξεργασία επιτρέπει την ακριβή υλοποίηση πολύπλοκων αλγορίθμων, όπως φίλτρα, δυναμικούς επεξεργαστές και φασματικές αναλύσεις, οι οποίοι θα ήταν δύσκολο ή αδύνατο να υλοποιηθούν αποκλειστικά με αναλογικά μέσα [23].

Στην πράξη, τα αναλογικά ηχητικά σήματα συχνά γίνονται αντιληπτά έχοντας έναν πιο «γεμάτο» ή «ζεστό» ήχο. Η αντίληψη αυτή δεν οφείλεται στο ίδιο το σήμα, αλλά στη φυσική συμπεριφορά των αναλογικών κυκλωμάτων που συνθέτουν την αναλογική βαθμίδα επεξεργασίας του σήματος. Στοιχεία όπως ενισχυτές, μετασχηματιστές και λαμπάτες βαθμίδες δεν λειτουργούν απόλυτα γραμμικά, με αποτέλεσμα να δημιουργούνται πρόσθετες αρμονικές συνιστώσες στο φάσμα του σήματος. Συγκεκριμένα, όταν η ένταση του εισερχόμενου σήματος αυξάνεται και προσεγγίζει τα όρια γραμμικής λειτουργίας του κυκλώματος, εμφανίζεται κορεσμός (saturation), κατά τον οποίο οι κορυφές του σήματος περιορίζονται ή κόβονται. Αυτή η συμπεριφορά οδηγεί σε παραμόρφωση, η οποία εκδηλώνεται με την εμφάνιση αρμονικών συνιστωσών που δεν υπήρχαν στο αρχικό σήμα.

Οι αρμονικές αυτές συμβάλλουν σε έναν χαρακτηριστικό ηχητικό χρωματισμό. Αντίθετα, τα ψηφιακά σήματα δεν εισάγουν εγγενώς τέτοιου είδους αρμονικές αλλοιώσεις, καθώς βασίζονται σε γραμμικά μαθηματικά μοντέλα. Για τον λόγο αυτό, η αναπαραγωγή αντίστοιχων χαρακτηριστικών σε ψηφιακά περιβάλλοντα επιτυγχάνεται μέσω αλγοριθμικών τεχνικών μη γραμμικής επεξεργασίας και μοντελοποίησης. Ωστόσο, οι προσεγγίσεις αυτές δεν αναπαράγουν πάντοτε με απόλυτη πιστότητα την φυσική συμπεριφορά των αναλογικών κυκλωμάτων καθώς η πολυπλοκότητα και οι μεταβολές της φυσικής υλοποίησης ενός αναλογικού κυκλώματος δεν μπορούν να αποτυπωθούν πλήρως σε ένα πεπερασμένο μαθηματικό μοντέλο.

3.3 Δειγματοληψία

Η δειγματοληψία αποτελεί τη διαδικασία μετατροπής ενός αναλογικού σήματος σε ψηφιακό, μέσω της λήψης μετρήσεων του πλάτους του σήματος σε τακτά χρονικά διαστήματα. Κάθε δείγμα αντιπροσωπεύει την τιμή του σήματος σε μια συγκεκριμένη χρονική στιγμή και αποθηκεύεται ως αριθμητική τιμή, επιτρέποντας την επεξεργασία του από ψηφιακά συστήματα. Ο ρυθμός δειγματοληψίας (sampling rate) εκφράζεται σε Hertz και καθορίζει τον αριθμό των δειγμάτων που λαμβάνονται ανά δευτερόλεπτο.



Σχήμα 3.2: Δειγματοληψία με διαφορετικό αριθμό δειγμάτων [24]

Σύμφωνα με το θεώρημα Nyquist–Shannon, για την ακριβή ανακατασκευή ενός σήματος χωρίς απώλειες πληροφορίας, ο ρυθμός δειγματοληψίας (Fs) πρέπει να είναι τουλάχιστον διπλάσιος της μέγιστης συχνότητας (fmax) που περιέχεται στο σήμα [25].

$$Fs > 2 \times fmax \tag{3.1}$$

Σε διαφορετική περίπτωση εμφανίζεται το φαινόμενο του aliasing, κατά το οποίο υψηλές συχνότητες «αναδιπλώνονται» σε χαμηλότερες, προκαλώντας ανεπιθύμητες παραμορφώσεις στο φάσμα του σήματος. Για τον λόγο αυτό, πριν τη δειγματοληψία εφαρμόζονται αναλογικά φίλτρα χαμηλής διέλευσης (anti-aliasing filters).

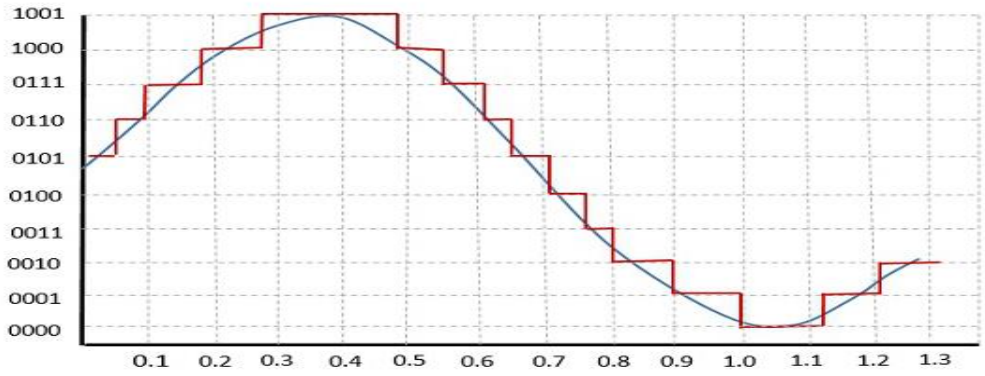
Στα συστήματα ψηφιακού ήχου, καθιερωμένες τιμές sampling rate είναι τα 44.1 kHz, που χρησιμοποιούνται κυρίως σε εφαρμογές μουσικής και CD audio, και τα 48 kHz, τα οποία προτιμώνται σε επαγγελματικές εφαρμογές ήχου και βίντεο [26]. Υψηλότεροι ρυθμοί δειγματοληψίας, όπως 88.2 kHz ή 96 kHz, προσφέρουν μεγαλύτερο περιθώριο επεξεργασίας και μειωμένο aliasing, αλλά απαιτούν αυξημένη υπολογιστική ισχύ και μνήμη.

Στην υλοποίηση ενός compressor, ο ρυθμός δειγματοληψίας επηρεάζει άμεσα τη χρονική συμπεριφορά βασικών παραμέτρων, όπως το attack και το release. Οι χρόνοι αυτοί εκφράζονται σε δευτερόλεπτα ή χιλιοστά του δευτερολέπτου, αλλά στην πράξη υλοποιούνται ως αριθμός δειγμάτων. Έτσι, για διαφορετικά sampling rates, ο ίδιος χρονικός συντελεστής μεταφράζεται σε διαφορετικό πλήθος δειγμάτων, γεγονός που καθιστά απαραίτητη την σωστή κλιμάκωση των παραμέτρων με βάση το sample rate του συστήματος.

3.4 Κβαντοποίηση & Bit Depth

Η κβαντοποίηση αποτελεί το στάδιο της ψηφιακής αναπαράστασης κατά το οποίο κάθε δείγμα του σήματος αντιστοιχίζεται σε μία από τις διαθέσιμες διακριτές στάθμες πλάτους. Σε αντίθεση με το αναλογικό σήμα, το οποίο μπορεί να λάβει άπειρο πλήθος τιμών, το ψηφιακό σύστημα περιορίζεται σε πεπερασμένο αριθμό επιπέδων, γεγονός που εισάγει σφάλμα προσέγγισης, γνωστό και ως θόρυβος κβαντοποίησης.

Ο θόρυβος κβαντοποίησης προκύπτει από τη διαφορά μεταξύ της πραγματικής αναλογικής τιμής του σήματος και της πλησιέστερης διαθέσιμης ψηφιακής στάθμης [27]. Σε χαμηλά bit depths, ο θόρυβος αυτός μπορεί να γίνει αντιληπτός, ιδιαίτερα σε σήματα χαμηλής στάθμης ή σε επαναλαμβανόμενες επεξεργασίες. Σε επαγγελματικά audio plugins, η χρήση εσωτερικής επεξεργασίας με floating-point ακρίβεια περιορίζει σημαντικά τις επιπτώσεις της κβαντοποίησης κατά την επεξεργασία.



Σχήμα 3.3: Διάγραμμα Κβαντοποίησης [28]

Το πλήθος των διαθέσιμων επιπέδων σε ένα ψηφιακό σύστημα καθορίζεται από το bit depth. Για ένα σύστημα με βάθος bit N, ο συνολικός αριθμός διακριτών επιπέδων πλάτους L δίνεται από τη σχέση (3.2) [29]:

$$L = 2^N \tag{3.2}$$

Στα συστήματα ψηφιακού ήχου, τα 16 bit χρησιμοποιούνται ευρέως σε εφαρμογές γενικής χρήσης, ενώ τα 24 bit αποτελούν πρότυπο σε επαγγελματικά περιβάλλοντα, λόγω του μεγαλύτερου δυναμικού εύρους που προσφέρουν.

Το bit depth επηρεάζει άμεσα το δυναμικό εύρος του συστήματος. Ο θεωρητικός λόγος σήματος προς θόρυβο (SNR) ενός ιδανικού N-bit κβαντιστή δίνεται από τη σχέση (3.3) [30]:

$$\mathrm{SNR} = 6.02N + 1.76\mathrm{dB} \tag{3.3}$$

Από αυτή τη σχέση προκύπτει ότι κάθε επιπλέον bit αυξάνει το δυναμικό εύρος κατά περίπου 6 dB. Έτσι, ένα σύστημα 16 bit προσφέρει δυναμικό εύρος περίπου 98 dB, ενώ ένα σύστημα 24 bit φτάνει τα 146 dB, ξεπερνώντας κατά πολύ τις δυνατότητες της ανθρώπινης ακοής και των περισσότερων αναλογικών κυκλωμάτων.

Στους ψηφιακούς compressors, η σωστή διαχείριση του δυναμικού εύρους είναι ιδιαίτερα κρίσιμη, καθώς η ίδια η λειτουργία της συμπίεσης στοχεύει στον έλεγχο και τη μείωση της δυναμικής διακύμανσης του σήματος. Εάν η κβαντοποίηση δεν ληφθεί υπόψη, η εφαρμογή μεγάλων βαθμών συμπίεσης ή έντονου make-up gain μπορεί να οδηγήσει σε φαινόμενα clipping στην έξοδο ενώ σε χαμηλές στάθμες, μπορεί να αναδείξει τον θόρυβο κβαντοποίησης. Για τον λόγο αυτό, ο compressor που υλοποιείται στην παρούσα εργασία έχει σχεδιαστεί ώστε να λειτουργεί εντός ασφαλών ορίων στάθμης, αξιοποιώντας την υψηλή ακρίβεια αριθμητικής κινητής υποδιαστολής που προσφέρει το περιβάλλον της JUCE.

3.5 Γραμμικοί και Μη-Γραμμικοί Επεξεργαστές Σήματος

Οι επεξεργαστές σήματος διακρίνονται σε γραμμικούς και μη γραμμικούς, ανάλογα με τη μαθηματική σχέση που συνδέει το σήμα εισόδου με το σήμα εξόδου. Ένα σύστημα χαρακτηρίζεται γραμμικό όταν ικανοποιεί την αρχή της υπέρθεσης, σύμφωνα με την οποία η απόκριση του συστήματος σε άθροισμα εισόδων ισούται με το άθροισμα των αποκρίσεων που θα παρήγαγε κάθε είσοδος ξεχωριστά [31].

Αν για ένα σύστημα ισχύει:

$$x_1(n) \rightarrow y_1(n)$$

$$x_2(n) \rightarrow y_2(n) \tag{3.4}$$

τότε για να είναι γραμμικό πρέπει να ισχύει:

$$x_1(n) + x_2(n) \rightarrow y_1(n) + y_2(n) \tag{3.5}$$

Η ιδιότητα αυτή συνεπάγεται ότι στους γραμμικούς επεξεργαστές, όπως τα φίλτρα και οι ισοσταθμιστές, η έξοδος μεταβάλλεται αναλογικά ως προς την είσοδο χωρίς να δημιουργούνται νέες φασματικές συνιστώσες.

Αντίθετα, ένα σύστημα χαρακτηρίζεται μη γραμμικό όταν δεν ικανοποιεί την αρχή της υπέρθεσης [31]. Ένα απλό παράδειγμα μη γραμμικού συστήματος δίνεται από τη σχέση (3.6):

$$y(n) = x^2(n) \tag{3.6}$$

Στην περίπτωση αυτή, η έξοδος που προκύπτει από άθροισμα εισόδων δεν ισούται με το άθροισμα των επιμέρους εξόδων. Η σχέση εισόδου-εξόδου δεν είναι αναλογική και οδηγεί σε δημιουργία νέων φασματικών συνιστωσών.

Οι compressors ανήκουν στην κατηγορία των μη γραμμικών επεξεργαστών, καθώς εφαρμόζουν μεταβαλλόμενη ενίσχυση στο σήμα ανάλογα με την στάθμη του. Όταν το σήμα υπερβαίνει ένα προκαθορισμένο όριο, η ενίσχυση μεταβάλλεται σύμφωνα με συγκεκριμένο λόγο συμπίεσης, με αποτέλεσμα η έξοδος να μην αποτελεί γραμμική κλιμάκωση της εισόδου. Η συμπεριφορά αυτή παραβιάζει την αρχή της υπέρθεσης και διαφοροποιεί τους compressors από τους γραμμικούς επεξεργαστές.

3.6 Ανάλυση Σήματος στο Πεδίο της Συχνότητας

Η ανάλυση σήματος στο πεδίο της συχνότητας αποτελεί βασικό εργαλείο της ψηφιακής επεξεργασίας σήματος, καθώς επιτρέπει την κατανόηση της φασματικής δομής ενός ηχητικού σήματος. Ενώ στο χρονικό πεδίο το σήμα περιγράφεται ως μεταβολή πλάτους στον χρόνο, στο συχνοτικό πεδίο αναλύεται ως άθροισμα ημιτονικών συνιστωσών διαφορετικών συχνοτήτων και πλατών. Η πλέον διαδεδομένη μέθοδος για τη μετάβαση από το χρονικό στο συχνοτικό πεδίο είναι ο Γρήγορος Μετασχηματισμός Fourier (Fast Fourier Transform – FFT), ο οποίος αποτελεί αποδοτική υπολογιστική υλοποίηση του Διακριτού Μετασχηματισμού Fourier (Discrete Fourier Transform – DFT).

Στην πράξη, το ηχητικό σήμα διαχωρίζεται σε διαδοχικά τμήματα σταθερού μήκους, τα οποία ονομάζονται πλαίσια (frames). Έπειτα, κάθε frame αναλύεται μέσω FFT, παράγοντας ένα σύνολο διακριτών bins. Κάθε bin αντιστοιχεί σε συγκεκριμένη συχνότητα, ενώ η τιμή του περιγράφει το πλάτος και τη φάση της συγκεκριμένης συχνοτικής συνιστώσας. Η πληροφορία αυτή μπορεί στη συνέχεια να χρησιμοποιηθεί για τη γραφική αναπαράσταση του φάσματος του σήματος, όπου οι τιμές των bins μετατρέπονται σε οπτικά στοιχεία στον άξονα της συχνότητας, επιτρέποντας την απεικόνιση της κατανομής της ενέργειας στο συχνοτικό πεδίο.

Το μέγεθος του FFT καθορίζει τη συχνοτική ανάλυση της επεξεργασίας, καθώς μεγαλύτερο μέγεθος FFT προσφέρει υψηλότερη ακρίβεια στον προσδιορισμό των συχνοτήτων, αλλά οδηγεί σε μεγαλύτερη χρονική καθυστέρηση στην ενημέρωση της ανάλυσης. Αντίθετα, μικρότερο μέγεθος FFT προσφέρει ταχύτερη χρονική απόκριση, εις βάρος όμως της συχνοτικής ανάλυσης.

Πριν την εφαρμογή του FFT χρησιμοποιείται συνήθως κάποιο παράθυρο (windowing function), όπως Hamming ή Hann window, με σκοπό τη μείωση των φαινομένων διαρροής φάσματος (spectral leakage) [32]. Η χρήση παραθύρου συμβάλλει σε πιο αξιόπιστη απεικόνιση του φάσματος, ιδιαίτερα σε σήματα που μεταβάλλονται δυναμικά, όπως ο ήχος μουσικών οργάνων ή η ανθρώπινη φωνή.

Παρότι ο compressor λειτουργεί κυρίως στο πεδίο του χρόνου και επηρεάζει τη δυναμική συμπεριφορά του σήματος και όχι άμεσα το φάσμα του, η ανάλυση συχνότητας παραμένει ιδιαίτερα χρήσιμη. Μέσω ενός spectrum analyzer, ο χρήστης μπορεί να παρακολουθεί την κατανομή της ενέργειας του σήματος πριν και μετά τη συμπίεση, εντοπίζοντας πιθανές μεταβολές στη φασματική ισορροπία ή ανεπιθύμητες παραμορφώσεις που ενδέχεται να προκύψουν έμμεσα από τη δυναμική επεξεργασία.

3.7 Real-Time DSP Απαιτήσεις σε Plugins

Τα audio plugins που λειτουργούν σε πραγματικό χρόνο υπόκεινται σε αυστηρούς περιορισμούς απόδοσης και χρονικής καθυστέρησης. Αυτό σημαίνει ότι η επεξεργασία κάθε block ήχου πρέπει να ολοκληρώνεται εντός ενός συγκεκριμένου χρονικού πλαισίου, ώστε να αποφεύγονται ακουστικά artifacts, dropouts ή διακοπές στην αναπαραγωγή.

Οι βασικές απαιτήσεις για real-time DSP σε audio plugins περιλαμβάνουν:

1. **Ποιότητα και Ακρίβεια Ήχου:**
   Η ποιότητα ήχου αποτελεί θεμελιώδη προϋπόθεση για κάθε επαγγελματικό plugin. Η επεξεργασία πρέπει να γίνεται χωρίς την προσθήκη ανεπιθύμητου θορύβου και ψηφιακών artifacts, ιδιαίτερα σε μη γραμμικές επεξεργασίες όπως η παραμόρφωση ή το saturation. Επιπλέον, η υποστήριξη υψηλών δειγματοληψιών είναι απαραίτητη για τη διατήρηση της ακεραιότητας του ήχου, ειδικά σε εφαρμογές mastering.

Στην πράξη, η έννοια της πιστότητας αναφέρεται στο πόσο πιστά αντιστοιχεί η έξοδος του συστήματος στην αρχική είσοδο δηλαδή, κατά πόσο ο επεξεργασμένος ήχος ακούγεται ίδιος ή αντιληπτά ισοδύναμος με το πρωτότυπο. Όσο υψηλότερη είναι η πιστότητα, τόσο λιγότερο αντιληπτές είναι οι αλλοιώσεις που εισάγει η επεξεργασία. Για τον λόγο αυτό, η πιστότητα αποτελεί σημαντικό παράγοντα κάθε συστήματος κωδικοποίησης καθώς καθορίζει το τελικό αποτέλεσμα που θα βιώσει ο ακροατής [33].

2. **Αποδοτικότητα και Επίδοση:**

   Η απόδοση ενός plugin σε πραγματικό χρόνο αποτελεί κρίσιμο παράγοντα για την επαγγελματική του χρήση και αξιολογείται μέσω τριών θεμελιωδών χαρακτηριστικών.

   Η χαμηλή κατανάλωση CPU είναι απαραίτητη για την ομαλή λειτουργία σε σύγχρονα παραγωγικά περιβάλλοντα. Ένα καλά σχεδιασμένο plugin ενσωματώνει βελτιστοποιημένους αλγορίθμους που επιτρέπουν την ταυτόχρονη λειτουργία πολλών instances χωρίς να υπερφορτώνουν τον επεξεργαστή, εξασφαλίζοντας έτσι τη δυνατότητα δημιουργίας πολύπλοκων συνθέσεων και μίξεων.

   Επιπλέον, η χαμηλή καθυστέρηση είναι ιδιαίτερα κρίσιμη για εφαρμογές όπως η ζωντανή ηχογράφηση και live συναυλίες όπου η άμεση απόκριση είναι απαραίτητη για τον μουσικό ή τον sound engineer. Στις περιπτώσεις αυτές, η καθυστέρηση θα πρέπει ιδανικά να είναι της τάξης των milliseconds.

   Τέλος, η σταθερότητα αποτελεί θεμελιώδη προϋπόθεση για κάθε επαγγελματικό plugin. Η αξιόπιστη λειτουργία χωρίς διακοπές, ειδικά κατά τη διάρκεια έντονης επεξεργασίας, είναι απαραίτητη σε επαγγελματικά περιβάλλοντα. Ένα ασταθές plugin μπορεί να προκαλέσει απώλεια δεδομένων ή να διακόψει κρίσιμες ηχογραφήσεις, καθιστώντας το ακατάλληλο για επαγγελματική χρήση.

3. **Διαχείριση Μνήμης και Πόρων:**

   Κατά τη λειτουργία σε πραγματικό χρόνο, η δυναμική δέσμευση μνήμης μπορεί να προκαλέσει απρόβλεπτες καθυστερήσεις. Η δέσμευση εξαρτάται από τον μηχανισμό διαχείρισης μνήμης του λειτουργικού συστήματος και από τον τρόπο με τον οποίο έχει σχεδιαστεί και υλοποιηθεί το plugin. Τέτοιες καθυστερήσεις μπορούν να διακόψουν την ομαλή ροή του audio thread και να οδηγήσουν σε ακουστικά artifacts. Για τον λόγο αυτό, η μνήμη δεσμεύεται εκ των προτέρων (pre-allocation) και επαναχρησιμοποιείται κατά την επεξεργασία.

4. **Προβλέψιμος Χρόνος Εκτέλεσης:**

   Κάθε block ήχου πρέπει να επεξεργάζεται εντός σταθερού χρονικού παραθύρου, δηλαδή εντός του χρονικού διαστήματος που αντιστοιχεί στην αναπαραγωγή του block πριν φτάσει το επόμενο. Το χρονικό αυτό παράθυρο καθορίζεται από το μέγεθος του buffer και τη συχνότητα δειγματοληψίας.

   Απρόβλεπτες καθυστερήσεις στον χρόνο εκτέλεσης μπορούν να οδηγήσουν σε glitches. Για τον λόγο αυτό στους αλγόριθμους αποφεύγονται λειτουργίες με μη σταθερό υπολογιστικό κόστος, όπως system calls ή βαριές αριθμητικές πράξεις χωρίς προβλέψιμη πολυπλοκότητα. Η υπολογιστική πολυπλοκότητα του αλγορίθμου πρέπει να είναι προβλέψιμη και σταθερή ως προς το μέγεθος των δεδομένων, ώστε ο χρόνος επεξεργασίας κάθε block να παραμένει εντός των χρονικών περιορισμών του συστήματος [33].

5. **Σταθερότητα και πρόληψη overflow:**

   Οι ψηφιακοί ήχοι μπορούν να αναπαρασταθούν με διαφορετικούς τρόπους, όπως floating-point, που επιτρέπει μεγάλο δυναμικό εύρος τιμών, ή fixed-point, όπου οι αριθμοί έχουν σταθερό σημείο και περιορισμένο εύρος. Κάθε μορφή έχει πλεονεκτήματα και περιορισμούς όσον αφορά την ακρίβεια και την ασφάλεια των αριθμητικών πράξεων.

   Οι αριθμητικές πράξεις σε πραγματικό χρόνο πρέπει να ελέγχονται ώστε να μην προκαλούν overflow, ειδικά σε επεξεργασίες δυναμικού εύρους. Σε συστήματα fixed-point, το περιορισμένο δυναμικό εύρος μπορεί να οδηγήσει σε υπερχείλιση όταν οι ενδιάμεσες μεταβλητές αποκτούν μεγάλες τιμές. Για την αντιμετώπιση του προβλήματος αυτού, έχουν χρησιμοποιηθεί τεχνικές όπως το block floating point, όπου ένα σύνολο δεδομένων μοιράζεται κοινό εκθέτη, επιτρέποντας αποτελεσματική διαχείριση του δυναμικού εύρους χωρίς απώλεια σταθερότητας [34].

   Παράλληλα, σε fixed-point αρχιτεκτονικές κάθε πράξη πολλαπλασιασμού μπορεί να εισάγει σφάλμα στρογγυλοποίησης, καθώς το γινόμενο δύο n-bit αριθμών απαιτεί διπλάσιο μήκος λέξης. Όταν το αποτέλεσμα περιορίζεται σε μικρότερο wordlength, τα λιγότερο σημαντικά bits αποκόπτονται ή στρογγυλοποιούνται, εισάγοντας θόρυβο στρογγυλοποίησης ο οποίος συσσωρεύεται κατά μήκος της αλυσίδας επεξεργασίας. Για τον λόγο αυτό, η χρήση μεγαλύτερου wordlength όπως 24-bit fixed-point ή 32-bit floating-point μειώνει σημαντικά την ισχύ του σφάλματος και συμβάλλει στη διατήρηση της ποιότητας ήχου [34].

3.8 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκαν οι θεμελιώδεις αρχές της ψηφιακής επεξεργασίας σήματος, εστιάζοντας στη μετατροπή αναλογικών σημάτων σε ψηφιακά μέσω δειγματοληψίας και κβαντοποίησης.

Αναλύθηκε ο ρόλος του bit depth και της ακρίβειας της αριθμητικής επεξεργασίας, καθώς και η διάκριση γραμμικών και μη γραμμικών συστημάτων. Επιπλέον, επισημάνθηκε η σημασία της ανάλυσης στο πεδίο της συχνότητας για την παρατήρηση φασματικών μεταβολών, καθώς και οι απαιτήσεις για αποδοτική υλοποίηση των audio plugins.

Με βάση τα παραπάνω, καθίσταται σαφής η σημασία της επεξεργασίας ήχου σε πραγματικό χρόνο, η οποία συνδέεται άμεσα με τη σχεδίαση και την αρχιτεκτονική των audio plugins, όπως παρουσιάζεται στο επόμενο κεφάλαιο.

Κεφάλαιο 4ο: Σχεδίαση και Αρχιτεκτονική Audio Plugins

4.1 Εισαγωγή

Η σχεδίαση και υλοποίηση ενός σύγχρονου audio plugin αποτελεί σύνθετο αντικείμενο, καθώς συνδυάζει απαιτήσεις ψηφιακής επεξεργασίας με την ανάγκη για σαφή και αξιόπιστη οπτική ανατροφοδότηση προς τον χρήστη. Τα audio plugins προορίζονται για χρήση σε ψηφιακά περιβάλλοντα επεξεργασίας ήχου και οφείλουν να λειτουργούν αξιόπιστα υπό αυστηρούς περιορισμούς καθυστέρησης και υπολογιστικής ισχύος.

Κεντρικός στόχος στον σχεδιασμό τέτοιων συστημάτων είναι η επίτευξη υψηλής ακρίβειας στην επεξεργασία, καθώς και στην αναπαράσταση των μετρήσεων και της δυναμικής συμπεριφοράς. Παράλληλα, ιδιαίτερη σημασία έχει ο σαφής διαχωρισμός μεταξύ του audio processing thread και του γραφικού περιβάλλοντος χρήστη, ώστε να διασφαλίζεται η σταθερότητα του συστήματος ακόμη και υπό έντονο φόρτο. Επιπλέον, ο σχεδιασμός των plugins στοχεύει στην επεκτασιμότητα, και στη συμβατότητα με διαφορετικά λειτουργικά συστήματα και audio formats, εξασφαλίζοντας με αυτόν τον τρόπο ευελιξία και αξιοπιστία του λογισμικού.

4.2 Ρύθμιση και Δομή Project

Στον χώρο της ψηφιακής επεξεργασίας, η επιλογή γλώσσας προγραμματισμού, framework και εργαλείων μεταγλώττισης επηρεάζει άμεσα την απόδοση και την σταθερότητα ενός audio plugin. Δεδομένου ότι τα plugins λειτουργούν σε περιβάλλοντα πραγματικού χρόνου, ακόμη και μικρές καθυστερήσεις ή μη αποδοτική διαχείριση πόρων μπορούν να οδηγήσουν σε σφάλματα και διακοπές.

Για τον λόγο αυτό, επιλέγονται γλώσσες προγραμματισμού που προσφέρουν υψηλή απόδοση και άμεσο έλεγχο της μνήμης και των υπολογιστικών πόρων. Η C++ αποτελεί την πιο διαδεδομένη επιλογή για εφαρμογές DSP σε πραγματικό χρόνο καθώς παρέχει ευελιξία και εκτεταμένο σύνολο βιβλιοθηκών για την ανάπτυξη audio plugins [35]. Εναλλακτικά, γλώσσες όπως Rust ή χαμηλού επιπέδου C χρησιμοποιούνται επίσης σε ορισμένα projects, ανάλογα με τις απαιτήσεις της απόδοσης και της ασφάλειας μνήμης.

Για τη διαχείριση και οργάνωση ενός plugin χρησιμοποιούνται ειδικά frameworks. Τα frameworks αποτελούν ολοκληρωμένες δομές λογισμικού που προσφέρουν την βασική αρχιτεκτονική σε εφαρμογές. Σε αντίθεση με τις βιβλιοθήκες, που παρέχουν συγκεκριμένες λειτουργίες, τα frameworks οργανώνουν την συνολική ροή και δομή της εφαρμογής, εφαρμόζοντας συγκεκριμένα πρότυπα σχεδίασης και οδηγίες κώδικα με στόχο την αποτελεσματική ανάπτυξη της [36].

Στην περίπτωση των audio plugins, τα frameworks αναλαμβάνουν τη διασύνδεση με τα πρότυπα όπως το VST και AU, την διαχείριση παραμέτρων, την επικοινωνία με τον host, καθώς και την υλοποίηση του γραφικού περιβάλλοντος και του μηχανισμού επεξεργασίας. Με αυτόν τον τρόπο, ο προγραμματιστής μπορεί να επικεντρωθεί στην υλοποίηση του αλγορίθμου DSP, χωρίς να χρειάζεται να διαχειρίζεται χαμηλού επιπέδου λεπτομέρειες.

Ενδεικτικά frameworks που χρησιμοποιούνται για τον σκοπό αυτό αποτελούν:

- **JUCE:** Διαδεδομένο framework σε C++, με έτοιμες κλάσεις για DSP, GUI, διαχείριση παραμέτρων και υποστήριξη προτύπων.
- **iPlug2:** Ελαφρύ και αποδοτικό framework σε C++ για την ανάπτυξη plugins σε μορφές VST, AU και AAX, με έμφαση στην απλότητα και την υψηλή απόδοση.
- **AudioKit:** Framework βασισμένο στη γλώσσα Swift, σχεδιασμένο κυρίως για εφαρμογές ήχου σε περιβάλλοντα iOS και macOS, με έμφαση στην ταχεία ανάπτυξη και τον εκπαιδευτικό χαρακτήρα.
- **RackAFX:** Framework σε C++ το οποίο προσφέρει έτοιμες DSP δομές και διευκολύνει την υλοποίηση και την δοκιμή αλγορίθμων επεξεργασίας ήχου.

Για την οργάνωση και αυτοματοποίηση της διαδικασίας μεταγλώττισης χρησιμοποιούνται εργαλεία build. Τα build systems αποτελούν λογισμικά που συλλέγουν όλες τις απαραίτητες πληροφορίες για τη δημιουργία μιας εφαρμογής, όπως εξαρτήσεις αρχείων, ρυθμίσεις compiler και κανόνες σύνδεσης [37]. Βασική λειτουργία τους είναι η διαχείριση του dependency graph του project και η υποστήριξη των incremental builds, δηλαδή η ανακατασκευή μόνο των τμημάτων του κώδικα που έχουν τροποποιηθεί. Αυτό είναι ιδιαίτερα σημαντικό σε μεγάλα projects ή σε εφαρμογές πραγματικού χρόνου, όπου απαιτείται ταχεία ανάπτυξη και δοκιμή.

Χαρακτηριστικά παραδείγματα αποτελούν:

- **CMake:** Cross-platform σύστημα build, που επιτρέπει τον καθορισμό της δομής του project και τον έλεγχο της διαδικασίας μεταγλώττισης σε διαφορετικά λειτουργικά συστήματα.
- **Ninja:** Γρήγορο και αποδοτικό build system, το οποίο χρησιμοποιείται συχνά μαζί με το CMake για την εκτέλεση των εντολών μεταγλώττισης.
- **Make**: Ένα από τα πρώτα εργαλεία build, χρησιμοποιεί Makefiles για τον καθορισμό των κανόνων μεταγλώττισης.
- **Bazel**: Αναπτύχθηκε από την Google και υποστηρίζει builds και tests σε πολλαπλές γλώσσες και πλατφόρμες, με έμφαση στην ταχύτητα και την κλιμάκωση.

Ο συνδυασμός γλώσσας προγραμματισμού υψηλής απόδοσης, κατάλληλου framework και αξιόπιστου συστήματος build παρέχει μια ευέλικτη και επεκτάσιμη βάση ανάπτυξης. Αυτή η προσέγγιση διευκολύνει την διαχείριση της πολυπλοκότητας των plugins, την υποστήριξη πολλαπλών πλατφορμών και την μελλοντική επέκταση των λειτουργιών τους.

4.3 Αρχιτεκτονική Συστήματος Ήχου

Η αρχιτεκτονική ενός συστήματος επεξεργασίας ήχου αποτελεί κρίσιμο παράγοντα για την λειτουργικότητα του. Σε περιβάλλοντα DAW, τα audio plugins καλούνται να επεξεργάζονται συνεχείς ροές δεδομένων υπό αυστηρούς χρονικούς περιορισμούς, γεγονός που απαιτεί συγκεκριμένες σχεδιαστικές επιλογές.

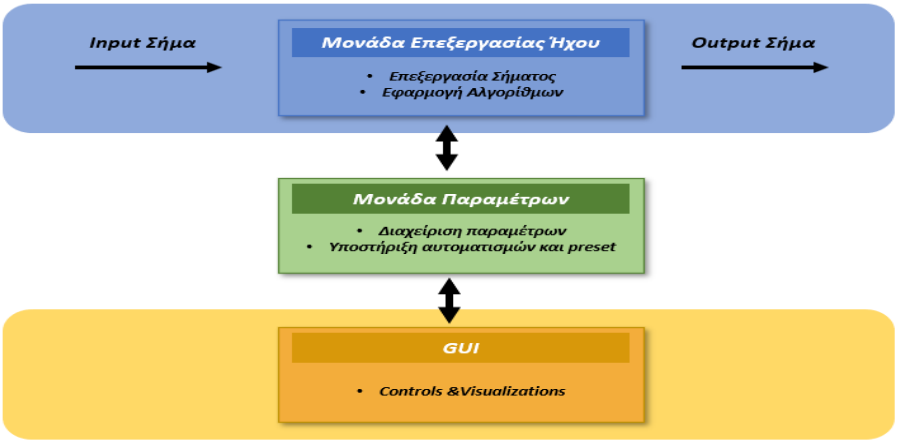
Κεντρική αρχή στη σχεδίαση τέτοιων συστημάτων είναι ο διαχωρισμός μεταξύ των λειτουργιών που σχετίζονται με την επεξεργασία του ηχητικού σήματος και αυτών που αφορούν την γραφική απεικόνιση και την αλληλεπίδραση με τον χρήστη. Οι σύγχρονες αρχιτεκτονικές βασίζονται συνήθως σε modular σχεδιασμό, όπου το σύστημα διαχωρίζεται σε διακριτές λειτουργικές μονάδες με συγκεκριμένους ρόλους. Στο JUCE αυτή η αρχιτεκτονική υλοποιείται μέσω των κλάσεων AudioProcessor και AudioProcessorEditor.

Σε ένα audio plugin, η επεξεργασία ήχου και το γραφικό περιβάλλον εκτελούνται ταυτόχρονα αλλά με διαφορετικές απαιτήσεις. Η επεξεργασία ήχου πραγματοποιείται σε αυστηρά χρονικά πλαίσια, καθώς πρέπει να ολοκληρώνεται εντός συγκεκριμένων προθεσμιών για κάθε block δεδομένων, ενώ το γραφικό περιβάλλον μπορεί να λειτουργεί με λιγότερους χρονικούς περιορισμούς. Για τον λόγο αυτό, οι δύο λειτουργίες οργανώνονται έτσι ώστε το γραφικό περιβάλλον να μην επηρεάζει την ομαλή επεξεργασία του ήχου.

Η αρχιτεκτονική ενός γενικού audio plugin περιλαμβάνει συνήθως τα εξής υποσυστήματα:

- **Μονάδα επεξεργασίας ήχου (Audio Processing Unit):** Λειτουργεί στο audio thread και είναι υπεύθυνη για την επεξεργασία του εισερχόμενου ηχητικού σήματος σε πραγματικό χρόνο. Σε αυτήν εφαρμόζονται οι αλγόριθμοι DSP, όπως η δυναμική συμπίεση, ενώ η επεξεργασία πραγματοποιείται σε blocks (buffers) με αυστηρούς χρονικούς περιορισμούς [38].
- **Μηχανισμός διαχείρισης παραμέτρων (Parameter Management):** Αποτελεί το ενδιάμεσο επίπεδο μεταξύ της επεξεργασίας ήχου και του γραφικού περιβάλλοντος. Συνδέει την επεξεργασία ήχου με το GUI, επιτρέποντας την ασφαλή ενημέρωση των τιμών χωρίς να επηρεάζεται η απόδοση του audio thread. Επιπλέον υποστηρίζει αυτοματισμούς από τον host.
- **Γραφικό περιβάλλον χρήστη (GUI):** Εκτελείται σε ανεξάρτητο νήμα και περιλαμβάνει στοιχεία ελέγχου και μονάδες οπτικής αναπαράστασης, όπως meters. Το GUI λαμβάνει δεδομένα παρακολούθησης (monitoring data) από την επεξεργασία ήχου, χωρίς να παρεμβαίνει άμεσα στον audio processing βρόχο.

Η ροή δεδομένων ακολουθεί μια σαφώς καθορισμένη κατεύθυνση. Το εισερχόμενο σήμα επεξεργάζεται από τα DSP modules, οδηγείται στην έξοδο του plugin και παράλληλα δημιουργούνται τα απαραίτητα δεδομένα για οπτική απεικόνιση και παρακολούθηση.



Σχήμα 4.1: Πρότυπο Αρχιτεκτονικής Audio Plugin

4.4 Μονάδες Επεξεργασίας Ήχου

Η μονάδα επεξεργασίας ήχου αποτελεί το βασικό λειτουργικό υποσύστημα ενός audio plugin και υλοποιείται συνήθως ως μια αλυσίδα διαδοχικών σταδίων επεξεργασίας, γνωστή και ως audio processing pipeline. Η ροή ξεκινά με την εισαγωγή του ηχητικού σήματος, το οποίο λαμβάνεται από το DAW ή από άλλες πηγές.

Η επεξεργασία πραγματοποιείται σε διαδοχικά στάδια, τα οποία μπορεί να περιλαμβάνουν:

- **Προσαρμογή στάθμης εισόδου:** Ρύθμιση της έντασης του εισερχόμενου σήματος ώστε να διασφαλίζεται σωστή λειτουργία των αλγορίθμων επεξεργασίας και αποφυγή παραμορφώσεων.
- **Ανάλυση σήματος:** Υπολογισμός χαρακτηριστικών του σήματος, όπως RMS, Peak ή άλλες μετρικές, που μπορούν να χρησιμοποιηθούν για τον έλεγχο των αλγορίθμων επεξεργασίας ή για την τροφοδότηση των οπτικών μετρητών.
- **Κύρια Επεξεργασία:** Εφαρμογή των κύριων DSP αλγορίθμων, όπως δυναμική επεξεργασία, χρήση φίλτρων ή άλλων ηχητικών τροποποιήσεων.
- **Έλεγχος στάθμης εξόδου και bypass:** Τελική ρύθμιση της έντασης του σήματος και δυνατότητα παράκαμψης της επεξεργασίας για σύγκριση του αρχικού και επεξεργασμένου σήματος.

Μια καλά σχεδιασμένη audio processing pipeline οφείλει να διαθέτει χαρακτηριστικά, όπως επεκτασιμότητα, ώστε να μπορεί να διαχειρίζεται αυξημένο αριθμό εισόδων χωρίς υποβάθμιση της απόδοσης. Επιπλέον απαιτείται αριθμητική ακρίβεια, προκειμένου οι αλγόριθμοι DSP να διατηρούν την φυσική μορφή του σήματος και αντιληπτική πιστότητα, ώστε το επεξεργασμένο σήμα να μην παρουσιάζει glitches ή ανεπιθύμητες αλλοιώσεις [39].

4.4.1 Ζώνες Επεξεργασίας του Ηχητικού Σήματος

Στην ψηφιακή επεξεργασία ήχου, το ηχητικό σήμα μπορεί να αντιμετωπιστεί είτε ως μία ενιαία ροή δεδομένων είτε ως σύνολο επιμέρους φασματικών περιοχών. Η δεύτερη προσέγγιση βασίζεται στην παρατήρηση ότι διαφορετικά τμήματα του φάσματος παρουσιάζουν συχνά διαφορετική ενεργειακή συμπεριφορά και διαφορετικές ανάγκες επεξεργασίας. Στους multi-band compressors, ο διαχωρισμός του φάσματος σε πολλαπλές ζώνες χρησιμοποιείται ώστε η δυναμική επεξεργασία να μπορεί να εφαρμοστεί με μεγαλύτερη ακρίβεια σε αυτές τις περιοχές [40].

Οι ζώνες επεξεργασίας αντιστοιχούν σε περιοχές του φάσματος, οι οποίες συχνά παρουσιάζουν διαφορετικά δυναμικά χαρακτηριστικά. Η έννοια των ζωνών αυτών είναι ιδιαίτερα σημαντική, καθώς επιτρέπει τη διαφοροποιημένη αντιμετώπιση των μεταβολών που εμφανίζονται σε διαφορετικά φασματικά τμήματα. Για παράδειγμα, οι χαμηλές συχνότητες μπορεί να απαιτούν διαφορετικό χειρισμό από τις υψηλές, τόσο ως προς τον βαθμό εφαρμογής της συμπίεσης όσο και ως προς τη χρονική συμπεριφορά της επεξεργασίας. Ωστόσο, ο διαχωρισμός του σήματος σε ζώνες αυξάνει την πολυπλοκότητα του συστήματος, καθώς απαιτεί τη χρήση φίλτρων διαχωρισμού, μηχανισμών ανεξάρτητης επεξεργασίας και διαδικασιών ανασύνθεσης του τελικού σήματος.

Η διαίρεση του φάσματος σε επιμέρους ζώνες δεν περιορίζεται μόνο σε εφαρμογές μουσικής παραγωγής ή audio plugins, αλλά συναντάται και σε άλλους τομείς της επεξεργασίας ήχου, όπως στις τηλεπικοινωνίες, στα συστήματα κωδικοποίησης και συμπίεσης ήχου καθώς και σε συστήματα ακουστικής υποβοήθησης. Ενδεικτικά, σε εφαρμογές που σχετίζονται με ακουστική υποβοήθηση, η πολυζωνική επεξεργασία έχει χρησιμοποιηθεί για την αντιμετώπιση φαινομένων όπως το spectral masking [41].

Παράλληλα, η απόδοση μιας πολυζωνικής μεθόδου επεξεργασίας μπορεί να επηρεάζεται σημαντικά από τον τρόπο διαχωρισμού του φάσματος, τον τύπο των ζωνών και τις παραμέτρους επεξεργασίας που εφαρμόζονται σε αυτές [42]. Η αύξηση του αριθμού των ζωνών μπορεί να προσφέρει μεγαλύτερη ακρίβεια ελέγχου, αλλά συνοδεύεται από αυξημένη υπολογιστική πολυπλοκότητα και πιθανές φασικές αποκλίσεις κατά τον συνδυασμό των επιμέρους ζωνών. Για τον λόγο αυτό, ο φασματικός διαχωρισμός δεν αποτελεί απλώς στάδιο οργάνωσης της επεξεργασίας, αλλά κρίσιμη σχεδιαστική επιλογή που μπορεί να επηρεάσει το τελικό ακουστικό αποτέλεσμα.

4.4.2 Single-band και Multi-band Μονάδες Επεξεργασίας

Με βάση τον τρόπο οργάνωσης του σήματος, οι μονάδες επεξεργασίας ήχου μπορούν να διακριθούν σε single-band και multi-band. Η διάκριση αυτή αποτελεί βασικό αρχιτεκτονικό χαρακτηριστικό των συστημάτων δυναμικής επεξεργασίας, καθώς επηρεάζει τόσο τη δομή της επεξεργασίας όσο και τον βαθμό ελέγχου που προσφέρεται στον χρήστη.

Στα single-band συστήματα, το συνολικό ηχητικό σήμα αντιμετωπίζεται ως μία ενιαία μονάδα και οδηγείται σε μία κοινή αλυσίδα επεξεργασίας. Η προσέγγιση αυτή χαρακτηρίζεται από απλούστερη υλοποίηση, περιορισμένη υπολογιστική πολυπλοκότητα και ευκολότερη διαχείριση των παραμέτρων. Για τον λόγο αυτό, χρησιμοποιείται ευρέως σε περιπτώσεις όπου επιδιώκεται συνολικός και άμεσος έλεγχος της δυναμικής συμπεριφοράς του σήματος.

Αντίθετα, στα multi-band συστήματα, το σήμα διαχωρίζεται σε περισσότερες φασματικές ζώνες, οι οποίες υποβάλλονται σε ανεξάρτητη επεξεργασία και στη συνέχεια αθροίζονται για την παραγωγή του τελικού σήματος εξόδου. Η αρχιτεκτονική αυτή επιτρέπει στοχευμένη επεξεργασία του ηχητικού υλικού, καθώς κάθε ζώνη μπορεί να διαθέτει διαφορετικές ρυθμίσεις και διαφορετική δυναμική συμπεριφορά. Ως αποτέλεσμα, τα multi-band συστήματα παρέχουν αυξημένο επίπεδο ελέγχου όταν το σήμα εμφανίζει ανομοιόμορφη φασματική κατανομή ή όταν απαιτείται επιλεκτική επεξεργασία συγκεκριμένων περιοχών.

Η δυνατότητα ανεξάρτητης επεξεργασίας των ζωνών καθιστά τα multi-band συστήματα κατάλληλα για εφαρμογές όπου απαιτείται προσαρμογή της επεξεργασίας σε φασματικές περιοχές που επηρεάζονται από θόρυβο ή masking. Σε τέτοιες συνθήκες, η πολυζωνική δυναμική επεξεργασία μπορεί να αξιοποιηθεί για τη βελτίωση του αντιληπτού λόγου σήματος προς θόρυβο, προσαρμόζοντας την επεξεργασία στις ιδιαιτερότητες του σήματος και του θορύβου [43].

Παρά τα πλεονεκτήματά τους, τα multi-band συστήματα συνοδεύονται από αυξημένες απαιτήσεις σχεδίασης και υλοποίησης. Η χρήση πολλαπλών ζωνών προϋποθέτει κατάλληλο διαχωρισμό του σήματος, ανεξάρτητη διαχείριση παραμέτρων ανά ζώνη, καθώς και προσεκτικό συνδυασμό των επιμέρους ζωνών, ώστε να διατηρείται η πιστότητα του τελικού αποτελέσματος. Συνεπώς, η επιλογή μεταξύ single-band και multi-band αρχιτεκτονικής εξαρτάται από τις απαιτήσεις της εφαρμογής, τη φύση του ηχητικού περιεχομένου και τον βαθμό ελέγχου που επιδιώκεται κατά την επεξεργασία.

4.5 Μονάδα Παραμέτρων

Η μονάδα παραμέτρων αποτελεί βασικό στοιχείο στη σχεδίαση ενός audio plugin, καθώς καθορίζει τον τρόπο με τον οποίο ο χρήστης αλληλεπιδρά με τους αλγορίθμους επεξεργασίας ήχου και επηρεάζει άμεσα τη λειτουργική συμπεριφορά του συστήματος. Σε περιβάλλοντα DAWs, οι παράμετροι λειτουργούν ως το κύριο μέσο ελέγχου της επεξεργασίας, επιτρέποντας την τροποποίηση κρίσιμων χαρακτηριστικών του ήχου.

Οι παράμετροι λειτουργούν ως διασύνδεση μεταξύ του χρήστη και του εσωτερικού συστήματος επεξεργασίας. Κάθε παράμετρος αποτελεί λογισμικό στοιχείο που αποθηκεύει τόσο την τρέχουσα τιμή όσο και τον τύπο της. Οι τιμές αυτές μπορεί να διαφέρουν ως προς τον τύπο τους ανάλογα με την λειτουργία που ελέγχουν [44]. Κατ' αυτόν τον τρόπο, οι παράμετροι μπορούν να αναπαριστούν συνεχή μεγέθη, όπως η ένταση που εκφράζεται ως πραγματικός αριθμός, διακριτά μεγέθη που αποδίδονται με ακέραιες τιμές, καθώς και δυαδικές καταστάσεις που περιγράφονται μέσω λογικών τιμών.

Πίνακας 4.1: Παράμετροι JUCE

Κατηγορία	Τύπος	Κλάση Παραμέτρου	Περιγραφή
Συνεχής	float	AudioParameterFloat	Παράμετρος με συνεχές εύρος τιμών
Διακριτή	int	AudioParameterInt	Παράμετρος ακέραιων τιμών με προκαθορισμένα βήματα
Δυαδική	bool	AudioParameterBool	Παράμετρος δύο καταστάσεων
Επιλογής	int (index)	AudioParameterChoice	Επιλογή από σύνολο προκαθορισμένων τιμών

Ένα σύγχρονο audio plugin οφείλει να υποστηρίζει αυτοματισμούς από το DAW. Οι παράμετροι πρέπει να μπορούν να αποθηκεύονται και να ανακαλούνται με συνέπεια, είτε μέσω presets είτε μέσω του project. Η αξιόπιστη διαχείριση των παραμέτρων διασφαλίζει την σωστή επαναφορά της συμπεριφοράς του plugin ανεξαρτήτως συστήματος ή χρονικής στιγμής.

4.6 Οπτικές Μονάδες

Το γραφικό περιβάλλον ενός audio plugin αποτελεί το κύριο μέσο αλληλεπίδρασης του χρήστη με το σύστημα. Μέσω του GUI, ο χρήστης μπορεί να ελέγξει παραμέτρους, να παρακολουθήσει την συμπεριφορά του σήματος και να αξιολογήσει τα αποτελέσματα των επεξεργασιών σε πραγματικό χρόνο. Λειτουργεί ως ενδιάμεσο μέσο ανάμεσα στον χρήστη και τη μονάδα επεξεργασίας ήχου, παρέχοντας άμεση οπτικοακουστική ανατροφοδότηση.

Μέσα στο GUI, οι οπτικές μονάδες αποτελούν ειδικά υποσυστήματα που αναπαριστούν σε πραγματικό χρόνο κρίσιμες πληροφορίες για το ηχητικό σήμα. Η κύρια λειτουργία τους είναι η απεικόνιση χαρακτηριστικών του σήματος, όπως επίπεδα έντασης, δυναμικές μεταβολές, ενεργειακή κατανομή σε συχνότητες, καθώς και άλλων σημαντικών μετρήσεων που βοηθούν στην αντίληψη του ήχου.

Τα εργαλεία απεικόνισης μπορούν να περιλαμβάνουν, μεταξύ άλλων:

- **Μετρητές στάθμης:** Παρέχουν ποσοτική πληροφόρηση σχετικά με την στιγμιαία και μέση ενεργειακή στάθμη του ηχητικού σήματος, όπως VU meters και Peak program meters (PPM) [45].
- **Waveform displays:** Εμφανίζουν την χρονική εξέλιξη του σήματος, επιτρέποντας την οπτική εκτίμηση της δυναμικής περιοχής και τον εντοπισμό πιθανών ανωμαλιών στο σήμα.
- **Προηγμένα γραφήματα και δείκτες:** Παρέχουν πληροφορίες για τη συμπεριφορά του σήματος μέσω αναπαραστάσεων όπως envelopes ή phase meters.

Οι οπτικές μονάδες λειτουργούν συμπληρωματικά προς την ακουστική αντίληψη. Για να διασφαλιστεί η λειτουργία του συστήματος, οι μονάδες αυτές συνήθως εκτελούνται σε ανεξάρτητα νήματα ή χρησιμοποιούν μηχανισμούς παρακολούθησης που δεν επηρεάζουν το κύριο audio processing thread. Με αυτόν τον τρόπο, εξασφαλίζεται αξιόπιστη η λειτουργία του συστήματος, ακόμη και σε περιβάλλοντα με αυξημένο φόρτο γραφικών ενημερώσεων.

4.7 Επίλογος

Στο κεφάλαιο αυτό αναλύθηκε η σχεδίαση και αρχιτεκτονική ενός σύγχρονου audio plugin, με έμφαση στον modular χαρακτήρα των συστημάτων, στη διαχωρισμένη εκτέλεση του audio processing thread και του GUI, καθώς και στη σωστή διαχείριση των παραμέτρων και των οπτικών μονάδων.

Επισημάνθηκε η σημασία της επιλογής γλώσσας προγραμματισμού, του κατάλληλου framework και αξιόπιστου συστήματος build για την επίτευξη real-time επεξεργασίας με υψηλή πιστότητα και σταθερότητα. Παράλληλα, παρουσιάστηκε η βασική λογική των ζωνών επεξεργασίας και η διάκριση μεταξύ single-band και multi-band αρχιτεκτονικών.

Οι αρχές αυτές αποτελούν τη θεωρητική και αρχιτεκτονική βάση για την πρακτική υλοποίηση του συστήματος, η οποία παρουσιάζεται στο επόμενο κεφάλαιο μέσα από την ανάλυση της δομής και των τεχνικών λεπτομερειών του συστήματος.

Κεφάλαιο 5ο: Υλοποίηση & Τεχνικές Λεπτομέρειες

5.1 Γενική Περιγραφή

Το παρόν κεφάλαιο επικεντρώνεται στην υλοποίηση και στις τεχνικές λεπτομέρειες του συστήματος δυναμικής επεξεργασίας ήχου τύπου compressor. Η αρχιτεκτονική του συστήματος βασίζεται στον διαχωρισμό μεταξύ της μονάδας επεξεργασίας ήχου και του γραφικού περιβάλλοντος χρήστη, ώστε να εξασφαλίζεται σαφής οργάνωση της λειτουργικότητας και αξιόπιστη επικοινωνία μεταξύ επεξεργασίας, οπτικοποίησης και ελέγχου παραμέτρων.

Το σύστημα αποτελείται από δύο βασικά υποσυστήματα όπου είναι η μονάδα επεξεργασίας ήχου και είναι υπεύθυνη για τη συμπίεση του σήματος και την διαχείριση των επιπέδων, και το γραφικό περιβάλλον χρήστη, το οποίο επιτρέπει την παραμετροποίηση και την παρακολούθηση της συμπεριφοράς του σήματος.

Η διεπαφή του συστήματος περιλαμβάνει περιστροφικούς επιλογείς για τη ρύθμιση των βασικών παραμέτρων και της στερεοφωνικής θέσης, καθώς και μετρητές εισόδου και εξόδου για την παρακολούθηση των επιπέδων σήματος. Επιπλέον, ενσωματώνονται μονάδες οπτικοποίησης, όπως waveform display, spectrum analyzer και VU meter, οι οποίες παρέχουν άμεση οπτική ανατροφοδότηση. Τέλος, η διεπαφή διαθέτει διακόπτη bypass για την παράκαμψη της επεξεργασίας, καθώς και επιπλέον κουμπιά για την εναλλαγή λειτουργιών, τη multi-band επεξεργασία, τη standalone λειτουργία και την επιλογή προκαθορισμένων ρυθμίσεων.

Σχήμα 5.1: Compressor Project

5.2 Εργαλεία και Τεχνολογίες Ανάπτυξης

Η ανάπτυξη του συγκεκριμένου compressor βασίστηκε σε σύγχρονες τεχνολογίες λογισμικού, οι οποίες επιλέχθηκαν με γνώμονα την απόδοση, τη σταθερότητα και τη συμβατότητα με σύγχρονα ψηφιακά περιβάλλοντα επεξεργασίας ήχου. Στο πλαίσιο αυτό, η υλοποίηση και η μεταγλώττιση πραγματοποιήθηκαν σε περιβάλλον Windows. Κεντρικός στόχος ήταν η δημιουργία ενός αξιόπιστου και επεκτάσιμου συστήματος, ικανού να λειτουργεί σε συνθήκες πραγματικού χρόνου με χαμηλή καθυστέρηση και σταθερή συμπεριφορά.

Το σύστημα υλοποιήθηκε στη γλώσσα προγραμματισμού C++, η οποία αποτελεί καθιερωμένο πρότυπο για εφαρμογές ψηφιακής επεξεργασίας ήχου και ανάπτυξης audio plugins. Η επιλογή της κρίθηκε κατάλληλη, καθώς επιτρέπει την ανάπτυξη αποδοτικών DSP αλγορίθμων και τον ακριβή έλεγχο της ροής επεξεργασίας στο audio thread, διασφαλίζοντας χαμηλή καθυστέρηση και αξιόπιστη λειτουργία.

Για την ανάπτυξη του συστήματος χρησιμοποιήθηκε το JUCE, το οποίο προσφέρει ένα ολοκληρωμένο περιβάλλον για audio processing και δημιουργία γραφικού περιβάλλοντος, με σαφή διαχωρισμό μεταξύ αυτών. Το συγκεκριμένο framework παρέχει έτοιμες δομές για τη διαχείριση παραμέτρων του compressor, την υποστήριξη αυτοματισμών από τα DAWs και εργαλεία για την υλοποίηση οπτικών μονάδων, μειώνοντας σημαντικά την πολυπλοκότητα του κώδικα.

Η ανάπτυξη και ο έλεγχος του compressor πραγματοποιήθηκαν στο περιβάλλον Visual Studio, το οποίο προσφέρει προηγμένα εργαλεία αποσφαλμάτωσης και ανάλυσης της εκτέλεσης του κώδικα. Με τη χρήση του περιβάλλοντος αυτού πραγματοποιήθηκε ο εντοπισμός σφαλμάτων καθ’ όλη την διάρκεια ανάπτυξης του κώδικα.

Για την διαδικασία της μεταγλώττισης χρησιμοποιήθηκε το CMake, ένα ευέλικτο σύστημα build που επιτρέπει την δημιουργία cross-platform builds και την διατήρηση ενιαίας δομής του project. Το CMake υποστήριξε την σύνδεση των απαραίτητων βιβλιοθηκών, την ρύθμιση των παραμέτρων του compiler και την δημιουργία των αρχείων build για διαφορετικά περιβάλλοντα ανάπτυξης.

Το plugin υλοποιήθηκε με υποστήριξη των μορφότυπων VST3 και AU, εξασφαλίζοντας συμβατότητα με ένα ευρύ σύνολο σύγχρονων DAWs. Το format VST3 επιλέχθηκε λόγω της εκτεταμένης χρήσης του και των δυνατοτήτων που προσφέρει, ενώ το Audio Unit προορίζεται για χρήση σε συστήματα macOS και DAWs όπως το Logic Pro, απαιτώντας ξεχωριστή μεταγλώττιση σε περιβάλλον macOS με χρήση Xcode και των αντίστοιχων Apple SDKs.

5.3 Δημιουργία Installer και Διανομή της Εφαρμογής

Για τη διανομή και εγκατάσταση της standalone εφαρμογής χρησιμοποιήθηκε το Inno Setup, ένα εργαλείο δημιουργίας installer για περιβάλλον Windows. Η χρήση του κρίθηκε απαραίτητη ώστε η εφαρμογή να μπορεί να διανεμηθεί και να εγκατασταθεί με οργανωμένο τρόπο, χωρίς να απαιτείται από τον χρήστη χειροκίνητη αντιγραφή αρχείων ή ξεχωριστή αναζήτηση των απαραίτητων runtime βιβλιοθηκών.

Στο πλαίσιο της διαδικασίας αυτής δημιουργήθηκε installer μέσω αρχείου script μορφής .iss, χρησιμοποιώντας τη σύνταξη του Inno Setup Script. Στο αρχείο αυτό ορίστηκαν τα βασικά στοιχεία της εφαρμογής, όπως το όνομα, η έκδοση, ο προτεινόμενος φάκελος εγκατάστασης, τα αρχεία που αντιγράφονται στο σύστημα και οι συντομεύσεις που δημιουργούνται. Μέσω του script, το εκτελέσιμο αρχείο της standalone εφαρμογής αντιγράφεται στον φάκελο που επιλέγει ο χρήστης, ενώ δημιουργείται και ο αντίστοιχος μηχανισμός απεγκατάστασης.

Κατά τις δοκιμές σε καθαρά περιβάλλοντα Windows διαπιστώθηκε ότι η εφαρμογή απαιτεί την ύπαρξη των Microsoft Visual C++ Runtime βιβλιοθηκών. Για τον λόγο αυτό, στον installer ενσωματώθηκαν τα αντίστοιχα Visual C++ Redistributable πακέτα. Με αυτόν τον τρόπο, ο χρήστης μπορεί να εγκαταστήσει τις απαραίτητες runtime βιβλιοθήκες κατά τη διαδικασία εγκατάστασης, αποφεύγοντας σφάλματα εκτέλεσης που σχετίζονται με αρχεία όπως msvcp140.dll, vcruntime140.dll και vcruntime140_1.dll.

Ο installer υποστηρίζει επιλογή αρχιτεκτονικής εγκατάστασης, περιλαμβάνοντας διαφορετικές εκδόσεις της εφαρμογής για x64, Win32/x86 και ARM64 συστήματα. Η x64 έκδοση αφορά σύγχρονα Windows συστήματα με Intel/AMD επεξεργαστές 64-bit, η Win32/x86 έκδοση αφορά παλαιότερα ή 32-bit Windows περιβάλλοντα, ενώ η ARM64 έκδοση αφορά Windows on ARM συσκευές. Ανάλογα με την επιλογή του χρήστη, εγκαθίσταται το αντίστοιχο εκτελέσιμο αρχείο και το αντίστοιχο Visual C++ Redistributable πακέτο.

Επιπλέον, στον installer προστέθηκε επιλογή για προαιρετική δημιουργία συντόμευσης στην επιφάνεια εργασίας. Με τον τρόπο αυτό, ο χρήστης μπορεί να επιλέξει αν επιθυμεί άμεση πρόσβαση στην εφαρμογή από το desktop, χωρίς η συντόμευση να δημιουργείται υποχρεωτικά. Παράλληλα, διατηρείται η δυνατότητα εκκίνησης της εφαρμογής από το Start Menu ή από τον φάκελο εγκατάστασης.

Η διαδικασία εγκατάστασης σχεδιάστηκε ώστε να είναι συμβατή κυρίως με σύγχρονα περιβάλλοντα Windows 10 και Windows 11. Κατά τη δοκιμή σε παλαιότερα περιβάλλοντα, όπως Windows 8.1, εμφανίστηκαν πρόσθετες εξαρτήσεις που σχετίζονται με το Universal C Runtime και απαιτούν ενημερώσεις του λειτουργικού συστήματος. Για τον λόγο αυτό, οι εκδόσεις Windows 10 και Windows 11 θεωρούνται ο βασικός στόχος υποστήριξης της παρούσας υλοποίησης, ενώ παλαιότερες εκδόσεις Windows δεν αποτέλεσαν κύριο στόχο συμβατότητας.

5.4 Audio Processing

Στο παρόν υποκεφάλαιο αναλύεται η διαδικασία επεξεργασίας ήχου που υλοποιεί το σύστημα, με έμφαση στην ροή του σήματος και στις τεχνικές ψηφιακής επεξεργασίας που εφαρμόζονται. Παρουσιάζεται η αρχιτεκτονική της επεξεργασίας, από την είσοδο του σήματος έως την έξοδό, καθώς και οι βασικές παράμετροι που επηρεάζουν τη λειτουργία της δυναμικής συμπίεσης.

5.4.1 Ροή Σήματος

Η ροή του σήματος αποτελεί βασικό στοιχείο της αρχιτεκτονικής επεξεργασίας ήχου, καθώς καθορίζει την σειρά και τον τρόπο με τον οποίο εφαρμόζονται τα επιμέρους στάδια επεξεργασίας. Στο παρόν σύστημα, η επεξεργασία του ήχου πραγματοποιείται σε blocks, σύμφωνα με τις απαιτήσεις επεξεργασίας σε πραγματικό χρόνο.

Κάθε block περιλαμβάνει συγκεκριμένο αριθμό δειγμάτων και διέρχεται από την ίδια ακολουθία επεξεργασίας. Η ακολουθία αυτή περιλαμβάνει στάδια προσαρμογής της στάθμης εισόδου, ανίχνευσης επιπέδων, δυναμικής συμπίεσης και ρύθμισης της στάθμης εξόδου. Η συνολική ροή του σήματος αποδίδεται ως εξής:

Είσοδος → Input Gain → Ανίχνευση Επιπέδων → Compression → Output Gain → Έξοδος

Στο στάδιο του input gain πραγματοποιείται αρχική ρύθμιση της στάθμης του σήματος πριν την είσοδο του στην δυναμική επεξεργασία. Στη συνέχεια ακολουθεί η ανίχνευση επιπέδων, κατά την οποία υπολογίζονται βασικά μεγέθη στάθμης που αξιοποιούνται τόσο για τη λειτουργία του compressor όσο και για την οπτική ανατροφοδότηση προς τον χρήστη. Στο στάδιο της δυναμικής συμπίεσης εφαρμόζεται ελεγχόμενη μείωση της δυναμικής περιοχής του σήματος, ενώ το output gain προσαρμόζει την στάθμη της εξόδου, διασφαλίζοντας το επιθυμητό τελικό επίπεδο σήματος.

5.4.2 Είσοδος

Η είσοδος του συστήματος αντιστοιχεί στο αρχικό audio buffer που εισέρχεται προς επεξεργασία. Το σήμα αυτό εισέρχεται στη ροή του compressor πριν από οποιαδήποτε επεξεργασία και αποτελεί το πρωτογενές σύνολο δειγμάτων πάνω στο οποίο βασίζεται η υπόλοιπη αλυσίδα επεξεργασίας. Στη λειτουργία plugin, το buffer εισόδου παρέχεται από το DAW μέσω του audio callback. Αντίστοιχα, στη standalone λειτουργία, το σήμα εισόδου προέρχεται από φορτωμένο αρχείο ήχου.

Στην περίπτωση φόρτωσης αρχείου, τα δεδομένα του ηχητικού σήματος ανακτώνται από το αντίστοιχο audio file, αποκωδικοποιούνται και μεταφέρονται σε κατάλληλη μορφή buffer ή πηγή αναπαραγωγής. Στη συνέχεια, το σήμα τροφοδοτεί την ίδια αλυσίδα επεξεργασίας με αυτή που χρησιμοποιείται και στη λειτουργία plugin, διασφαλίζοντας ενιαία λογική επεξεργασίας ανεξάρτητα από την προέλευση του σήματος.

5.4.3 Input και Output Gain

Ο compressor περιλαμβάνει στάδια input gain και output gain, τα οποία επιτρέπουν τον έλεγχο της στάθμης του σήματος πριν και μετά τη δυναμική συμπίεση. Τα στάδια αυτά παίζουν καθοριστικό ρόλο στη συνολική συμπεριφορά του και στη διαμόρφωση των επιπέδων του σήματος.

Το input gain εφαρμόζεται πριν την είσοδο του σήματος στον compressor και καθορίζει τη στάθμη με την οποία το σήμα φτάνει στο threshold. Με τον τρόπο αυτό επηρεάζεται έμμεσα ο βαθμός ενεργοποίησης της συμπίεσης, καθώς διαφορετικές ρυθμίσεις input gain μπορούν να οδηγήσουν σε εντονότερη ή ηπιότερη δυναμική επεξεργασία για το ίδιο εισερχόμενο σήμα.

Το output gain εφαρμόζεται μετά την ολοκλήρωση της συμπίεσης και χρησιμοποιείται για την αντιστάθμιση της απώλειας στάθμης που προκαλείται από το gain reduction. Στόχος του είναι η διατήρηση του επιθυμητού συνολικού επιπέδου του σήματος στην έξοδο, χωρίς να αλλοιώνεται ο χαρακτήρας της επεξεργασίας.

Στα δύο αυτά στάδια εφαρμόζεται μηχανισμός εξομάλυνσης στις μεταβολές του gain, ώστε να επιτυγχάνεται σταδιακή μεταβολή των επιπέδων ενίσχυσης. Η προσέγγιση αυτή αποτρέπει την εμφάνιση ανεπιθύμητων artifacts κατά την real-time επεξεργασία και διασφαλίζει ομαλή και φυσική συμπεριφορά της στάθμης.

5.4.4 Ανίχνευση Επιπέδων

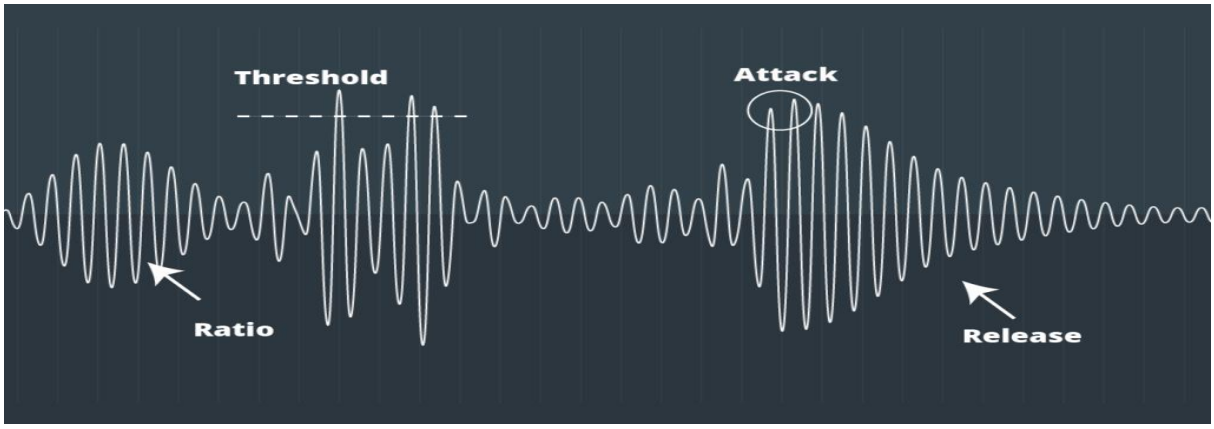
Για την αξιολόγηση της δυναμικής του σήματος, το σύστημα υπολογίζει Peak και RMS τιμές για κάθε κανάλι. Οι Peak τιμές αντιπροσωπεύουν τις στιγμιαίες κορυφές του σήματος και χρησιμοποιούνται ως βάση για τον υπολογισμό της συμπίεσης από τον compressor. Επιπλέον χρησιμοποιούνται για την ανίχνευση clipping και για την ένδειξη των LED meters, επιτρέποντας στον χρήστη να βλέπει τις απότομες μεταβολές της στάθμης σε πραγματικό χρόνο. Αντίθετα, οι RMS τιμές υπολογίζουν την ενεργό ένταση του σήματος, παρέχοντας μια πιο ρεαλιστική εκτίμηση της στάθμης. Αυτές οι τιμές χρησιμοποιούνται για την σταθερή απεικόνιση στο VU meter για την ένδειξη input και output.

Για την σταθεροποίηση της οπτικής αναπαράστασης των μετρήσεων, εφαρμόζεται χρονική εξομάλυνση στους υπολογισμούς των Peak και RMS τιμών. Η εξομάλυνση υλοποιείται μέσω συντελεστών, οι οποίοι καθορίζουν τον ρυθμό μεταβολής των ενδείξεων, χωρίς ωστόσο να επηρεάζουν την επεξεργασία του ίδιου του ηχητικού σήματος. Με τον τρόπο αυτό αποφεύγονται απότομες διακυμάνσεις στους μετρητές, επιτρέποντας στην οπτική πληροφορία να εμφανίζεται με ομαλό ρυθμό, καθιστώντας την ευκολότερα αντιληπτή από τον χρήστη και πιο συμβατή με την φυσική αντίληψη της στάθμης.

5.4.5 Δυναμική Συμπίεση

Στη δυναμική συμπίεση, ο πυρήνας της επεξεργασίας είναι ο compressor, ο οποίος μειώνει τη δυναμική περιοχή του σήματος όταν αυτό υπερβαίνει ένα προκαθορισμένο επίπεδο. Η λειτουργία του καθορίζεται από τις βασικές παραμέτρους [46]:

- **Threshold:** Ορίζει το επίπεδο πάνω από το οποίο ενεργοποιείται η συμπίεση. Τα σήματα που βρίσκονται κάτω από το threshold παραμένουν ανεπεξέργαστα.
- **Ratio:** Καθορίζει τον βαθμό μείωσης της στάθμης του σήματος για το τμήμα που υπερβαίνει το threshold. Εκφράζει το πόσο έντονα περιορίζεται η δυναμική περιοχή του σήματος, με μεγαλύτερες τιμές ratio να οδηγούν σε ισχυρότερη συμπίεση, ενώ μικρότερες τιμές να αντιστοιχούν σε ηπιότερη δυναμική επεξεργασία.
- **Attack:** Ορίζει τον χρόνο που απαιτείται για να εφαρμοστεί η συμπίεση από τον compressor, όταν η στάθμη του σήματος ξεπεράσει το threshold. Μικροί χρόνοι attack επιτρέπουν γρήγορη αντίδραση στις κορυφές, ενώ μεγαλύτεροι χρόνοι οδηγούν σε πιο ομαλή και φυσική απόκριση.
- **Release:** Καθορίζει τον χρόνο που απαιτείται για να σταματήσει η συμπίεση, όταν η στάθμη του σήματος πέσει κάτω από το threshold. Ένας κατάλληλος χρόνος release εξασφαλίζει ομαλή επαναφορά, αποφεύγοντας απότομες μεταβολές στην ένταση του ήχου.



Σχήμα 5.2: Παράμετροι Compressor [47]

Όταν το σήμα υπερβαίνει το threshold, ο compressor υπολογίζει το απαιτούμενο gain reduction και το εφαρμόζει σταδιακά, σύμφωνα με τους χρόνους attack και release. Η χρήση αυτών των παραμέτρων εξασφαλίζει ότι η μείωση της δυναμικής περιοχής δεν αλλοιώνει δραστικά τον χαρακτήρα του ήχου και διατηρείται η φυσικότητα του σήματος.

5.4.6 Gain Reduction

Η μείωση κέρδους αποτελεί το βασικό μέγεθος που εκφράζει τη λειτουργία της δυναμικής συμπίεσης, υπολογίζοντας τη διαφορά σε decibels (dB) μεταξύ της στάθμης εισόδου και της στάθμης εξόδου. Η μείωση του κέρδους καθορίζεται από την σχέση (5.1):

$$\text{Gain Reduction (dB)} = \text{Input Level (dB)} - \text{Output Level (dB)} \tag{5.1}$$

Η χρονική συμπεριφορά της gain reduction καθορίζεται από τις παραμέτρους attack και release, οι οποίες ορίζουν πόσο γρήγορα η μείωση ενεργοποιείται ή απενεργοποιείται, εξασφαλίζοντας ομαλή δυναμική επεξεργασία και προστασία του ήχου από απότομες μεταβολές.

Στον compressor, η gain reduction υπολογίζεται με βάση τις Peak τιμές του σήματος και καθορίζεται από την μέθοδο calculateGainReductionDb(). Αρχικά ελέγχεται αν η στάθμη του σήματος υπερβαίνει το προκαθορισμένο threshold. Αν η στάθμη είναι κάτω από το όριο, δεν εφαρμόζεται συμπίεση και η μείωση κέρδους παραμένει μηδενική.

Όταν το σήμα υπερβαίνει το threshold, υπολογίζεται η διαφορά στάθμης πάνω από το όριο (excess level):

$$\text{Excess(dB)} = \text{Input Level(dB)} - \text{Threshold(dB)} \tag{5.2}$$

Στη συνέχεια εφαρμόζεται ο λόγος συμπίεσης (ratio), που καθορίζει πόσο θα μειωθεί η στάθμη του σήματος πάνω από το threshold:

$$\text{Compression(dB)} = \text{Excess(dB)} \cdot \left(1 - \frac{1}{\text{Ratio}}\right) \tag{5.3}$$

Η τελική τιμή της gain reduction παίρνει αρνητικό πρόσημο, καθώς αντιπροσωπεύει μείωση στάθμης:

$$\text{Gain Reduction(dB)} = -\text{Compression(dB)} \tag{5.4}$$

Το αρνητικό πρόσημο χρησιμοποιείται στους υπολογισμούς και στους μετρητές για να εκφράζει με σαφήνεια ότι η στάθμη εξόδου μειώνεται σε σχέση με την είσοδο.

Η παρακολούθηση της gain reduction μέσω των VU meters και άλλων οπτικών ενδείξεων είναι κρίσιμη για τον χρήστη, καθώς επιτρέπει την κατανόηση του τρόπου με τον οποίο η δυναμική επεξεργασία εφαρμόζεται. Η οπτική ανατροφοδότηση βοηθά στην ακριβή ρύθμιση των παραμέτρων αποτρέποντας τόσο την υπερβολική συμπίεση όσο και την ανεπαρκή επεξεργασία.

5.4.7 Bypass

Η λειτουργία bypass επιτρέπει την απενεργοποίηση της δυναμικής επεξεργασίας, διατηρώντας αμετάβλητη τη ροή του σήματος από την είσοδο προς την έξοδο. Στο πλαίσιο του συστήματος, η λειτουργία αυτή ενεργοποιείται μέσω ειδικού switch, το οποίο παρέχει τη δυνατότητα άμεσης εναλλαγής μεταξύ επεξεργασμένου και μη επεξεργασμένου σήματος.

Σε κατάσταση bypass, το σήμα παρακάμπτει τον compressor, επιτρέποντας την άμεση σύγκριση μεταξύ ενεργής και ανενεργής κατάστασης. Η ύπαρξη της λειτουργίας παρέχει στον χρήστη τη δυνατότητα να εκτιμήσει αντικειμενικά την επίδραση της δυναμικής επεξεργασίας, χωρίς να επηρεάζεται από μεταβολές στη συνολική στάθμη ή στον χαρακτήρα του σήματος.

Η υλοποίηση του bypass έχει σχεδιαστεί με γνώμονα το real-time περιβάλλον λειτουργίας των DAWs. Για τον λόγο αυτό, η μετάβαση μεταξύ των καταστάσεων πραγματοποιείται ομαλά, αποφεύγοντας απότομες αλλαγές. Οι μονάδες οπτικοποίησης προσαρμόζουν την λειτουργία τους στην κατάσταση αυτή, απενεργοποιώντας τις σχετικές ενδείξεις τους με animations υποδεικνύοντας ότι ο compressor δεν επεμβαίνει στο σήμα. Με τον τρόπο αυτό καθίσταται σαφές ότι το σήμα διέρχεται χωρίς επεξεργασία, διευκολύνοντας τον χρήστη.

5.4.8 Έξοδος

Η έξοδος του συστήματος αντιστοιχεί στο τελικό audio buffer που παράγεται μετά την ολοκλήρωση της αλυσίδας επεξεργασίας. Το buffer αυτό εκφράζει το τελικό αποτέλεσμα της δυναμικής επεξεργασίας και αποτελεί τη μορφή με την οποία το σήμα αποδίδεται προς αναπαραγωγή ή περαιτέρω χρήση. Στη μορφή plugin, το επεξεργασμένο σήμα επιστρέφεται στο DAW, ενώ στη standalone λειτουργία μπορεί να αποδοθεί είτε στο σύστημα αναπαραγωγής είτε να εξαχθεί σε αρχείο ήχου.

5.5 Γραφικό Περιβάλλον

Το γραφικό περιβάλλον του compressor έχει σχεδιαστεί ώστε να παρέχει άμεση οπτική αναπαράσταση της επεξεργασίας του ήχου, διευκολύνοντας την αλληλεπίδραση του χρήστη με τις βασικές παραμέτρους. Η σχεδίαση δίνει έμφαση στη σαφήνεια της πληροφορίας και στην απόκριση σε πραγματικό χρόνο, επιτρέποντας την άμεση αξιολόγηση της επίδρασης των ρυθμίσεων στον ήχο.

Η διεπαφή του συστήματος οργανώνεται σε διακριτά οπτικά και διαδραστικά στοιχεία, τα οποία συμβάλλουν στον αποτελεσματικό έλεγχο της επεξεργασίας. Κάθε στοιχείο επιτελεί συγκεκριμένο ρόλο στη αλληλεπίδραση του χρήστη με το σύστημα. Η δομή και η λειτουργία αυτών των στοιχείων αναλύονται στις επόμενες υποενότητες.

5.5.1 VU Meter

Η κλάση ProgrammaticVUMeter υλοποιεί έναν ψηφιακό VU meter, ο οποίος αποτελεί μετρητή στάθμης με βελόνα. Απεικονίζει σε πραγματικό χρόνο τις RMS τιμές εισόδου και εξόδου του compressor σε κλίμακα dBFS, καθώς και τη μείωση κέρδους σε Peak. Η επιλογή της RMS μέτρησης βασίζεται στο γεγονός ότι τα αναλογικά VU meters αποτυπώνουν τη μέση ενεργειακή στάθμη, προσεγγίζοντας περισσότερο την ανθρώπινη αντίληψη της έντασης.

Η τιμή RMS υπολογίζεται από την σχέση (5.5) [48]:

$$\mathrm{RMS} = \sqrt{\frac{1}{\mathrm{N}} \sum_{\mathrm{n}=0}^{\mathrm{N}-1} \mathrm{x}[\mathrm{n}]^2} \tag{5.5}$$

όπου x[n] είναι τα δείγματα του σήματος στο buffer και N ο αριθμός των δειγμάτων που λαμβάνονται υπόψη. Η πράξη $\mathrm{x}[\mathrm{n}]^2$ παίρνει το τετράγωνο της στιγμιαίας τιμής, ώστε οι θετικές και αρνητικές τιμές να υπολογίζονται με τον ίδιο τρόπο στη μέση ενεργειακή στάθμη, ενώ η ρίζα τετραγώνου επαναφέρει τη μονάδα μέτρησης στην ίδια κλίμακα ώστε η τιμή να είναι συγκρίσιμη με τα αρχικά δείγματα.

Έπειτα, η τιμή RMS μετατρέπεται σε λογαριθμική κλίμακα dBFS (decibels relative to full scale) μέσω του τύπου (5.6) [49]:

$$\mathrm{L_{dBFS}} = 20 \cdot \log_{10}(\mathrm{RMS}) \tag{5.6}$$

Η κλίμακα dBFS χρησιμοποιείται στην ψηφιακή επεξεργασία ήχου για να εκφράσει την ένταση ενός σήματος σε σχέση με το μέγιστο δυνατό επίπεδο (full scale) που μπορεί να αναπαρασταθεί ψηφιακά, το οποίο ορίζεται ως 0 dBFS. Η μετατροπή αυτή είναι απαραίτητη ώστε η ένδειξη της στάθμης να μπορεί να χαρτογραφηθεί σωστά στο φόντο του VU meter.

Η κλάση του VU meter κληρονομεί από juce::Component και juce::Timer, επιτρέποντας τόσο την προγραμματισμένη σχεδίαση του γραφικού περιβάλλοντος όσο και την περιοδική ενημέρωση της ένδειξης. Η σχεδίαση υλοποιείται μέσω της μεθόδου paint() ενώ η ενημέρωση πραγματοποιείται μέσω της timerCallback(), η οποία εκτελείται στα 60 Hz, εξασφαλίζοντας ομαλή και σταθερή ανανέωση.

Η κλάση συνδυάζει ένα προγραμματισμένο GUI με βελόνα όπου σχεδιάζεται μέσω της μεθόδου drawRealisticNeedle(). Επιπλέον περιλαμβάνεται ένδειξη LED και text display τα οποία είναι όλα τοποθετημένα πάνω σε μια γραφιστικά επεξεργασμένη εικόνα.

Το VU meter υποστηρίζει δύο βασικές καταστάσεις λειτουργίας:

- **Κανονική λειτουργία:** Η εικόνα εμφανίζεται με αυξημένη φωτεινότητα, υποδεικνύοντας ότι ο μετρητής είναι ενεργός.
- **Bypass:** Η εικόνα σκουραίνει και η βελόνα κινείται με animation προς τα -20 dBFS, υποδεικνύοντας οπτικά ότι ο compressor δεν επηρεάζει το σήμα.

Επιπλέον το VU meter διαθέτει τρία buttons, IN, OUT και GR, τα οποία επιτρέπουν την επιλογή διαφορετικών λειτουργιών εμφάνισης και ορίζονται ως εξής:

- **IN:** Εμφανίζει τη στάθμη εισόδου RMS του compressor σε κλίμακα dBFS. Η ένδειξη ξεκινά από αρνητικές τιμές dBFS, που αντιστοιχούν στο αριστερό άκρο της κλίμακας, και μεταβάλλεται προς τα δεξιά καθώς αυξάνεται η στάθμη του σήματος πριν την επεξεργασία.
- **OUT:** Εμφανίζει τη στάθμη εξόδου RMS του compressor σε κλίμακα dBFS. Αντίστοιχα, η ένδειξη ξεκινά από αρνητικές τιμές dBFS και μετακινείται προς τα δεξιά, απεικονίζοντας το τελικό επίπεδο του σήματος μετά την εφαρμογή της συμπίεσης.
- **GR:** Εμφανίζει τη μείωση κέρδους σε Peak τιμές, η οποία εφαρμόζεται από τον compressor σε κλίμακα dB. Σε αυτή τη λειτουργία, η βελόνα ξεκινά από τα 0 dB και κινείται προς τα αρνητικά dB κάθε φορά που υπάρχει μείωση κέρδους. Στη multi-band λειτουργία, η ένδειξη gain reduction αντιστοιχεί στην επιλεγμένη ζώνη Low, Mid ή High.

Το GUI περιλαμβάνει επίσης text display για τις τιμές Current και Max. Η ένδειξη Current δείχνει την τρέχουσα τιμή της βελόνας, ενώ η Max εμφανίζει τη μέγιστη τιμή που έχει καταγραφεί καθ΄ όλη τη διάρκεια της μέτρησης. Οι τιμές ενημερώνονται με χαμηλότερη συχνότητα, ώστε να μειώνονται τα οπτικά άλματα της βελόνας και οι ενδείξεις να παραμένουν ευανάγνωστες. Επιπλέον μέσω της μεθόδου mouseDoubleclick() ο χρήστης έχει την δυνατότητα με χρήση διπλού κλικ, να επαναφέρει της ενδείξεις Current και Max στις αρχικές τους τιμές.

Σχήμα 5.3: VU Meter Buttons

Η θέση της βελόνας υπολογίζεται μέσω σημείων βαθμονόμησης (calibration points) που ορίζονται στη συνάρτηση setupDBCalibration(). Τα σημεία αυτά έχουν τοποθετηθεί πάνω στην εικόνα του μετρητή και αντιστοιχίζονται σε συγκεκριμένες τιμές dB, δημιουργώντας έναν πίνακα αναφοράς μεταξύ αριθμητικής τιμής στάθμης και γωνιακής θέσης της βελόνας. Η προσέγγιση αυτή επιτρέπει ακριβή χαρτογράφηση ακόμη και σε περιπτώσεις μη γραμμικής κλίμακας.

Η κίνηση της βελόνας πραγματοποιείται με ομαλό τρόπο μέσω δύο συναρτήσεων εξομάλυνσης:

- **IN/OUT Mode**: Χρησιμοποιείται η συνάρτηση applySmoothing(), με διαφορετικούς συντελεστές attack και release, ώστε η βελόνα να ακολουθεί ομαλά τις αυξομειώσεις του σήματος.
- **GR Mode**: Χρησιμοποιείται η συνάρτηση applySmoothingGR(), η οποία περιορίζει τις απότομες αλλαγές λαμβάνοντας υπόψη τον τρόπο μεταβολής της μείωσης κέρδους.

Για τον έλεγχο της ορθότητας της βαθμονόμησης υλοποιήθηκε η βοηθητική συνάρτηση drawCalibrationPointsDebug(), η οποία προβάλλει τα σημεία βαθμονόμησης και το σημείο περιστροφής της βελόνας.

Σχήμα 5.4: Debug για Calibration Points

Στο VU meter υπάρχει οπτική ένδειξη clipping, η οποία ενεργοποιείται όταν η στάθμη του σήματος υπερβαίνει τα 0 dBFS. Στην περίπτωση αυτή, το αντίστοιχο LED ανάβει, ενώ οι ενδείξεις Current και Max του text display αλλάζουν σε χρώμα κόκκινο, προσφέροντας άμεση και ευδιάκριτη πληροφόρηση για την κατάσταση υπερφόρτωσης του σήματος. Τα γραφικά του LED βασίζονται σε sprite δύο καταστάσεων το οποίο φορτώνεται μέσω της μεθόδου loadLedFrames().

Η λειτουργία του LED υλοποιείται μέσω της συνάρτησης updateLedState(), η οποία ελέγχει συνεχώς την τρέχουσα τιμή του μετρητή currentDisplayDb και ενημερώνει την κατάσταση του LED.

Η συνάρτηση εφαρμόζει τις εξής λογικές:

- Αν ο compressor βρίσκεται σε bypass, το LED σβήνει άμεσα, ανεξάρτητα από την τιμή του σήματος, για να υποδηλώνει ότι ο compressor δεν επεμβαίνει.
- Αν ο compressor είναι ενεργός, η συνάρτηση συγκρίνει την τρέχουσα τιμή currentDisplayDb με τα όρια ledOnThreshold και ledOffThreshold. Αν η στάθμη υπερβαίνει το όριο ενεργοποίησης, το LED ανάβει ενώ αν πέσει κάτω από το όριο απενεργοποίησης, το LED σβήνει.

Σχήμα 5.5: Κατάσταση Clipping VU Meter

Πίνακας 5.1: Μέθοδοι VU Meter

Κατηγορίες	Μέθοδοι	Λειτουργίες
Κατασκευή	ProgrammaticVUMeter(), ~ProgrammaticVUMeter(), setProcessor()	Αρχικοποίηση και σύνδεση με processor
Διαχείριση Κατάστασης	setVUMode(), setBypassMode(), resetMeterDisplay(), resetSmoothingForMode()	Διαχείριση bypass, mode και επαναφοράς μετρητή
Υπολογισμοί & Μετατροπές	getTargetDbForMode(), getRealMeterDb(), calculateNeedlePosition(), setupDBCalibration()	Υπολογισμός και βαθμονόμηση ενδείξεων μετρητή
Απεικόνιση	paint(), drawRealisticNeedle(), drawBypassNeedle(), drawLedIndicator(), drawCalibrationPointsDebug()	Σχεδίαση background,βελόνας, LED, text display και Debug calibraion

LED & Clipping	updateLedState(), loadLedFrames(), drawLedIndicator()	Έλεγχος κατάστασης LED, φόρτωση sprite
Smoothing	applySmoothing(), applySmoothingGR()	Ομαλή κίνηση βελόνας και ενδείξεων
Διαδραστικότητα	mouseDoubleClick()	Επαναφορά μετρητή με διπλό κλικ
Περιοδική ανανέωση	timerCallback()	Ανανέωση τιμών

5.5.2 LED Meter

Η κλάση LED VU Meter του compressor έχει σχεδιαστεί ως στερεοφωνικό σύστημα οπτικής απεικόνισης της στιγμιαίας στάθμης του σήματος, αποτελούμενο από δύο ανεξάρτητες κατακόρυφες στήλες που αντιστοιχούν στο αριστερό και στο δεξί κανάλι. Κάθε στήλη λειτουργεί αυτόνομα ως προς την μέτρηση και την ένδειξη Peak, διατηρώντας ωστόσο κοινή κλίμακα και ενιαία οπτική λογική, ώστε να επιτρέπεται η άμεση σύγκριση μεταξύ των δύο καναλιών. Η ενημέρωση των οπτικών ενδείξεων ανανεώνεται μέσω Timer στα 60 Hz ενώ η συνολική σχεδίαση του meter πραγματοποιείται στην συνάρτηση paint(), η οποία αναλαμβάνει την τελική απεικόνιση των επιμέρους στοιχείων.

Η οπτική ένδειξη του LED meter βασίζεται στη μέτρηση της στιγμιαίας στάθμης του ψηφιακού σήματος. Τα δείγματα ήχου που λαμβάνονται από το audio buffer βρίσκονται σε κανονικοποιημένη μορφή κινητής υποδιαστολής στο εύρος (5.7):

$$-1.0 \leq x[n] \leq 1.0 \tag{5.7}$$

Οι θετικές τιμές αντιστοιχούν στη θετική ημιπερίοδο του σήματος, δηλαδή σε αυξημένη πίεση ή ένταση του ήχου προς μία κατεύθυνση, ενώ οι αρνητικές τιμές αντιπροσωπεύουν την αρνητική ημιπερίοδο, δηλαδή μειωμένη πίεση ή ένταση προς την αντίθετη κατεύθυνση. Η τιμή 0 αντιστοιχεί σε μηδενικό πλάτος, δηλαδή στην απουσία στιγμιαίας έντασης του σήματος.

Αρχικά υπολογίζεται η απόλυτη τιμή του δείγματος (5.8) ώστε να ληφθεί το πλάτος του σήματος ανεξάρτητα από την πολικότητα καθώς η αντίληψη της έντασης δεν εξαρτάται από αυτήν.

$$A = |x[n]| \tag{5.8}$$

Στη συνέχεια η τιμή αυτή μετατρέπεται σε στάθμη dBFS χρησιμοποιώντας τον λογαριθμικό μετασχηματισμό από την σχέση (5.6). Μετά τον υπολογισμό της στάθμης, η τιμή περιορίζεται στο εύρος λειτουργίας του meter.

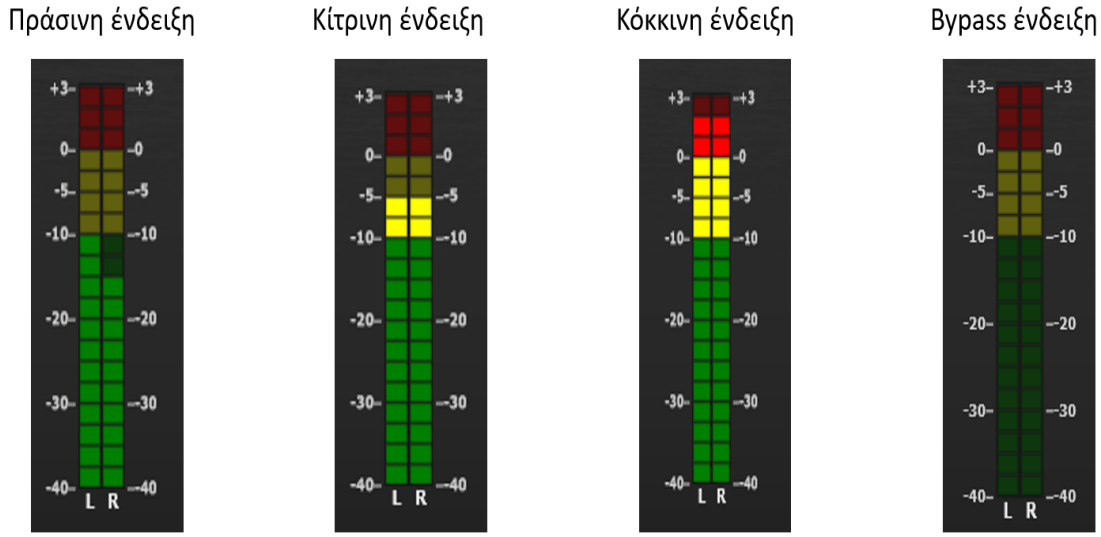
$$-40 \leq L_{dBFS} \leq +3 \tag{5.9}$$

Η συνολική δυναμική περιοχή του meter εκτείνεται από −40 dBFS έως +3 dBFS και αποδίδεται μέσω 19 διακριτών τμημάτων LED. Τιμές μικρότερες από −40 dBFS θεωρούνται χαμηλές στάθμες, ενώ τιμές που υπερβαίνουν το όριο 0 dBFS αντιστοιχούν σε περιοχή υπερφόρτωσης (clipping). Για λόγους οπτικής ανατροφοδότησης, η κλίμακα επεκτείνεται έως +3 dBFS, επισημαίνοντας την εμφάνιση υπερφόρτωσης του σήματος.

Η κατανομή των LED δεν είναι γραμμική ως προς το εύρος των dBFS, αλλά οργανώνεται σε segments με διαφορετική χρωματική κωδικοποίηση. Στις στάθμες από −40 dBFS έως −10 dBFS χρησιμοποιούνται πράσινα LED, τα οποία υποδηλώνουν ασφαλή επίπεδα σήματος. Στην περιοχή από -10 dBFS μέχρι 0 dBFS χρησιμοποιούνται κίτρινα LED, τα οποία λειτουργούν ως προειδοποίηση για αυξημένη στάθμη, ενώ για τιμές από 0 dBFS έως +3 dBFS χρησιμοποιούνται κόκκινα LED, τα οποία υποδηλώνουν clipping και ενδεχόμενη παραμόρφωση. Η συγκεκριμένη κατανομή και η αντιστοίχιση στάθμης-χρώματος υλοποιούνται εσωτερικά μέσω πινάκων αναφοράς στην συνάρτηση setupLedPositions().

Η αντιστοίχιση των LED με τις τιμές dBFS υλοποιείται μέσω πίνακα αναφοράς ledDbValues, όπου κάθε LED αντιστοιχεί σε συγκεκριμένο επίπεδο στάθμης. Με βάση την τρέχουσα τιμή dBFS του σήματος, υπολογίζεται ο αριθμός των LED που πρέπει να ενεργοποιηθούν, ξεκινώντας από τη βάση της στήλης προς την κορυφή. Τα ανενεργά LED αποδίδονται με χαμηλότερη φωτεινότητα, ώστε να διατηρείται η συνοχή της κλίμακας.

Παράλληλα με τις στήλες των LED, εμφανίζεται αριθμητική κλίμακα dBFS τοποθετημένη στις εξωτερικές πλευρές του meter. Τα labels περιλαμβάνουν επιλεγμένες τιμές αναφοράς −40, −30, −20, −10, −5, 0 και +3 dBFS. Η χαρτογράφηση και η κατακόρυφη τοποθέτηση των αριθμητικών ενδείξεων υλοποιούνται μέσω της συνάρτησης setupDbLabels(), ενώ η ακριβής θέση κάθε label υπολογίζεται δυναμικά με τη συνάρτηση getLabelYPosition(). Η σχεδίαση της κλίμακας πραγματοποιείται μέσω των συναρτήσεων drawDbScaleLeft() και drawDbScaleRight(), αντίστοιχα για το αριστερό και δεξί κανάλι.



Σχήμα 5.6: Ενδείξεις LED Meter

Η χρονική συμπεριφορά του meter έχει σχεδιαστεί ώστε να προσεγγίζει την λειτουργία επαγγελματικών Peak Meters. Οι αυξήσεις της στάθμης εμφανίζονται άμεσα μέσω fast attack λειτουργίας, ώστε να αποδίδονται με ακρίβεια τα transients του σήματος, ενώ οι μειώσεις πραγματοποιούνται με αργό ρυθμό slow release, προκειμένου η ένδειξη να παραμένει ευανάγνωστη και να αποφεύγεται το απότομο σβήσιμο των LED. Η συμπεριφορά αυτή καθιστά την ένδειξη εύκολα αντιληπτή και χρήσιμη κατά την μίξη ή επεξεργασία ήχου.

Επιπλέον, υλοποιείται μηχανισμός Peak hold, ο οποίος καταγράφει την μέγιστη στιγμιαία τιμή στάθμης για κάθε κανάλι και την διατηρεί ορατή για συγκεκριμένο χρονικό διάστημα πριν αρχίσει να μειώνεται σταδιακά. Με τον τρόπο αυτό, ο χρήστης μπορεί να εντοπίσει αιχμές που ενδέχεται να μην είναι άμεσα αντιληπτές κατά την συνεχή ροή του σήματος. Η πρόσβαση στις τιμές Peak hold πραγματοποιείται μέσω των μεθόδων getLeftPeakHoldDb() και getRightPeakHoldDb(), οι οποίες επιστρέφουν τις αντίστοιχες τιμές για το αριστερό και δεξί κανάλι.

Τέλος, η λειτουργία bypass ενσωματώνεται μέσω των συναρτήσεων setBypass() και getBypass(). Όταν ο compressor βρίσκεται σε κατάσταση bypass, το meter τροφοδοτείται με τιμές στάθμης κάτω από το κατώφλι της κλίμακας, με αποτέλεσμα όλα τα LED να απενεργοποιούνται οπτικά, παρέχοντας σαφή και άμεση ένδειξη αδρανούς κατάστασης.

Πίνακας 5.2: Μέθοδοι LED Meter

Κατηγορίες	Μέθοδοι	Λειτουργίες
Κατασκευή	LedVuMeter(), ~LedVuMeter()	Δημιουργία component, αρχικοποίηση και ρύθμιση του LED Meter
Διαχείριση Δεδομένων Στάθμης	setLevelsDb(), setLevels(), setLeftLevel(), setRightLevel()	Εισαγωγή τιμών dBFS ή normalized levels
Παραμετροποίηση	setBypass(), getBypass()	Ενεργοποίηση/απενεργοποίηση bypass
Ανάκτηση Δεδομένων	getLeftPeakHoldDb(), getRightPeakHoldDb()	Επιστροφή τιμών Peak Hold
Περιοδική ανανέωση	timerCallback()	Ανανέωση Τιμών
Απεικόνιση	paint(), drawSingleColumn(), drawDbScaleLeft(), drawDbScaleRight(), drawChannelLabels()	Σχεδίαση LED columns, dBFS κλίμακες και ετικέτες καναλιών
Βαθμονόμηση & Ρύθμιση	setupLedPositions(), setupDbLabels(), dbToLedLevel(), getLabelYPosition()	Ορισμός τιμών και χρωμάτων LED, θέσεις ετικετών dBFS
Ενημέρωση	updatePeakDb()	Ενημέρωση Peak hold τιμών με smoothing

5.5.3 Spectrum Analyzer

Η κλάση SpectrumAnalyzer υλοποιεί έναν φασματικό αναλυτή, ο οποίος απεικονίζει σε πραγματικό χρόνο τη σχέση μεταξύ συχνότητας και έντασης του σήματος ήχου. Σχεδιάζεται με την χρήση του framework και χρησιμοποιεί FFT υψηλής ανάλυσης για να παρέχει λεπτομερή ακρίβεια. Ο σχεδιασμός του επικεντρώνεται στην οπτική σαφήνεια, την ομαλή απεικόνιση και τη δυνατότητα παραμετροποίησης.

Η κλάση περιλαμβάνει εσωτερικούς buffers για την αποθήκευση των δειγμάτων ήχου, των δεδομένων FFT και των αποτελεσμάτων του μετασχηματισμού. Τα δείγματα ήχου εισάγονται στον αναλυτή μέσω της συνάρτησης pushBuffer(), η οποία συλλέγει και συγχωνεύει τα κανάλια του εισερχόμενου audio buffer σε ένα μονοφωνικό σήμα. Η διαδικασία αυτή πραγματοποιείται με τον υπολογισμό του μέσου όρου των καναλιών, σύμφωνα με τη σχέση (5.10):

$$x_{\text{mono}}[n] = \frac{1}{C}\sum_{c=1}^{C} x_c\,[n] \tag{5.10}$$

όπου $x_c[n]$ είναι το δείγμα του καναλιού c στη χρονική στιγμή n, C ο συνολικός αριθμός καναλιών και $x_{\text{mono}}[n]$ το μονοφωνικό αποτέλεσμα. Με τον τρόπο αυτό εξασφαλίζεται ότι όλα τα κανάλια συμμετέχουν ισότιμα στον υπολογισμό.

Η επαναφορά της εσωτερικής κατάστασης του αναλυτή, όπως των buffers, πραγματοποιείται μέσω της συνάρτησης clear().

Πριν τον φασματικό μετασχηματισμό FFT, εφαρμόζεται παράθυρο Hamming στα δεδομένα εισόδου, το οποίο μειώνει τα φαινόμενα φασματικής διαρροής και προσφέρει βελτιωμένη ανάλυση στις συχνότητες [50].

$$w[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \tag{5.11}$$

Η παραμετρική τιμή N αντιστοιχεί στο μέγεθος του FFT. Έπειτα κάθε block πολλαπλασιάζεται με την αντίστοιχη τιμή του παραθύρου (5.12):

$$x_{\text{windowed}}[n] = x[n] \times w[n] \tag{5.12}$$

Ο φασματικός μετασχηματισμός υλοποιείται στην συνάρτηση processFFT(). Ο αναλυτής χρησιμοποιεί FFT μεγέθους 8192 σημείων (FFT order 13), προσφέροντας υψηλή ανάλυση σε όλο το ακουστικό φάσμα. Ο μετασχηματισμός αυτός βασίζεται στην μαθηματική σχέση DFT (5.13) [51]:

$$X(k) = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi kn}{N}} \tag{5.13}$$

Μετά την εφαρμογή του μετασχηματισμού FFT, το αποτέλεσμα για κάθε bin είναι ένας μιγαδικός αριθμός που περιλαμβάνει πληροφορία για το πλάτος-ένταση της αντίστοιχης συχνότητας. Για την απεικόνιση του φάσματος υπολογίζεται το μέτρο του μιγαδικού αριθμού σύμφωνα με την σχέση (5.14):

$$|\,X(k)\,| = \sqrt{\mathrm{Re}(k)^2 + \mathrm{Im}(k)^2} \tag{5.14}$$

όπου $\mathrm{Re}(k)$ και $\mathrm{Im}(k)$ είναι αντίστοιχα το πραγματικό και το φανταστικό μέρος του αποτελέσματος της FFT. Στη συνέχεια, το πλάτος κάθε συχνότητας μετατρέπεται σε κλίμακα dBFS, χρησιμοποιώντας τον τύπο (5.6), προκειμένου η απεικόνιση του φάσματος να ανταποκρίνεται στη λογαριθμική κλίμακα.

Παράλληλα, εφαρμόζεται zero-padding, ώστε να αυξηθεί η ακρίβεια του φάσματος χωρίς την ανάγκη μεγαλύτερης υπολογιστικής ισχύος. Έπειτα εφαρμόζεται overlap processing με βαθμό επικάλυψης 25% του μεγέθους FFT δηλαδή κάθε νέο frame μετατοπίζεται κατά fftSize/4 δείγματα σε σχέση με το προηγούμενο. Η χρήση αυτή αυξάνει την χρονική πυκνότητα της ανάλυσης και συμβάλλει σε ομαλότερη οπτική μετάβαση μεταξύ διαδοχικών ενημερώσεων.

Η αντιστοίχιση των FFT bins σε πραγματικές συχνότητες πραγματοποιείται με βάση το sample rate που καθορίζεται από το DAW, το οποίο ορίζεται δυναμικά μέσω της συνάρτησης setSampleRate(). Η τιμή αυτή ορίζει το εύρος συχνοτήτων που μπορεί να απεικονιστεί, καθώς και την απόσταση μεταξύ των διαδοχικών συχνοτικών σημείων και βασίζεται στην σχέση (5.15):

$$f_k = \frac{k \times f_s}{N} \tag{5.15}$$

όπου f_k είναι η συχνότητα που αντιστοιχεί στο bin k, f_s το sample rate και N το μέγεθος του FFT που είναι $2^{13} = 8192$ δείγματα. Με τον τρόπο αυτό κάθε σημείο του φάσματος αντιστοιχεί σε συγκεκριμένο εύρος συχνοτήτων.

Μετά τον υπολογισμό του φάσματος, οι τιμές των συχνοτήτων και των επιπέδων έντασης χαρτογραφούνται σε συντεταγμένες της οθόνης ώστε να σχεδιαστεί η γραφική απεικόνιση του φάσματος. Ο οριζόντιος άξονας της απεικόνισης αντιστοιχεί στην συχνότητα, ενώ ο κατακόρυφος άξονας αντιστοιχεί στο επίπεδο έντασης σε dBFS.

Η χαρτογράφηση των συχνοτήτων πραγματοποιείται σε λογαριθμική κλίμακα. Για τον λόγο αυτό χρησιμοποιείται ο λογαριθμικός μετασχηματισμός (5.16) [52]:

$$x_{norm} = \frac{\log(f_k / f_{min})}{\log(f_{max} / f_{min})} \tag{5.16}$$

όπου f_k είναι η συχνότητα του bin, f_{min} και f_{max} τα κατώτερα και ανώτερα όρια συχνοτήτων που εμφανίζονται στο γράφημα. Η σχέση αυτή μετατρέπει τη συχνότητα σε μια κανονικοποιημένη τιμή στο διάστημα [0,1], η οποία χρησιμοποιείται για τον υπολογισμό της οριζόντιας συντεταγμένης της συχνότητας στο γράφημα.

Αντίστοιχα, οι τιμές στάθμης σε dBFS χαρτογραφούνται στον κατακόρυφο άξονα μέσω γραμμικής κλίμακας μεταξύ των ελάχιστων και μέγιστων επιπέδων που εμφανίζονται στο γράφημα. Η χαρτογράφηση αυτή πραγματοποιείται σύμφωνα με την σχέση (5.17):

$$y = \mathrm{map}(L_{dB}, L_{min}, L_{max}) \tag{5.17}$$

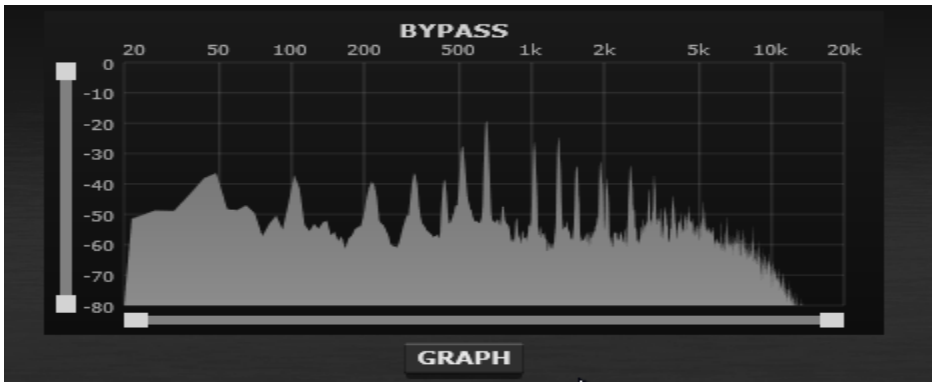
όπου L_{dB} είναι η στάθμη της συχνότητας σε dBFS και L_{min}, L_{max} τα όρια της δυναμικής περιοχής του αναλυτή.

Με τον τρόπο αυτό κάθε συχνοτική συνιστώσα του σήματος μετατρέπεται σε σημείο στην οθόνη και τα σημεία αυτά συνδέονται ώστε να σχηματίσουν την καμπύλη του φάσματος. Η διαδικασία αυτή επαναλαμβάνεται συνεχώς σε πραγματικό χρόνο, επιτρέποντας την δυναμική παρακολούθηση της φασματικής κατανομής του σήματος.

Η γραφική αναπαράσταση περιλαμβάνει:

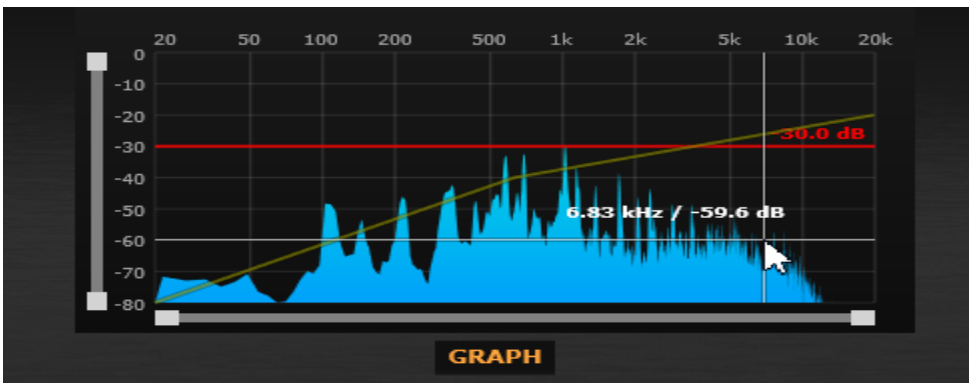
- Δυναμική ομαλή ενημέρωση με temporal smoothing, ώστε οι κορυφές του φάσματος να κινούνται φυσικά και χωρίς απότομα jumps.
- Psychoacoustic tilt, που ενισχύει τις υψηλές συχνότητες και εφαρμόζει προσαρμογή χαμηλών, ώστε η απεικόνιση να ανταποκρίνεται καλύτερα στην ανθρώπινη ακοή.
- Low frequency cut και smoothing για καλύτερη ανάγνωση των χαμηλών συχνοτήτων χωρίς οπτική παραμόρφωση.

Το component υποστηρίζει color gradients για το φάσμα, δίνοντας έναν σύγχρονο και αισθητικό χαρακτήρα. Το gradient αυτό ενημερώνεται δυναμικά μέσω της συνάρτησης updateGradient(). Σε κατάσταση bypass, χρησιμοποιείται ξεχωριστό gradient και εμφανίζεται ένα έντονο overlay με την ένδειξη "BYPASS", ώστε ο χρήστης να γνωρίζει ότι το σήμα δεν υποβάλλεται σε επεξεργασία. Παράλληλα, στην οθόνη εμφανίζεται το φάσμα συχνοτήτων του raw input σήματος με γκρι χρώμα, χωρίς να εφαρμόζονται οι ρυθμίσεις του compressor.



Σχήμα 5.7: Λειτουργία Bypass Spectrum Analyzer

Ο αναλυτής υποστηρίζει επίσης την απεικόνιση της γραμμής ratio και της γραμμής threshold. Η γραμμή λογού συμπίεσης έχει κίτρινο χρώμα και υπολογίζεται μέσω της συνάρτησης setCompressionCurve(), ενώ η σχεδίασή της πραγματοποιείται στην συνάρτηση drawCurve(). Η γραμμή threshold εμφανίζεται με κόκκινο χρώμα και υποδεικνύει το επίπεδο threshold του compressor στο φάσμα. Η εμφάνιση και η απόκρυψη των γραφικών αυτών στοιχείων λειτουργεί με ένα GRAPH button και ελέγχεται μέσω των μεθόδων showCompressionCurve(), hideCompressionCurve() και setShowCurve().



Σχήμα 5.8: Κανονική Λειτουργία Spectrum Analyzer

Για τη βελτίωση της αλληλεπίδρασης με τον χρήστη, ο αναλυτής υποστηρίζει δυναμικά tooltips και crosshair. Κατά την μετακίνηση του δείκτη του ποντικιού στο πλαίσιο του αναλυτή φάσματος, εμφανίζονται οι τιμές συχνότητας και dBFS. Η θέση του δείκτη παρακολουθείται μέσω των συναρτήσεων mouseMove(), mouseDrag() και mouseExit(), ενώ η λογική υπολογισμού των τιμών αυτών υλοποιείται στην συνάρτηση updateTooltip(). Η γραφική απεικόνιση του tooltip πραγματοποιείται μέσω της μεθόδου drawTooltip(), ενώ η απεικόνιση του crosshair υλοποιείται με την drawCrosshair(). Το crosshair λειτουργεί ως οπτικός οδηγός πάνω στο φάσμα, βοηθώντας τον χρήστη να εντοπίσει με μεγαλύτερη ακρίβεια και ευκολία το σημείο το οποίο έχει επιλέξει.

Επιπρόσθετα, ο αναλυτής διαθέτει RangeSlider, ένα custom γραφικό στοιχείο σχεδιασμένο για την επιλογή εύρους τιμών. Ο RangeSlider χρησιμοποιείται για τον έλεγχο του zoom τόσο στον οριζόντιο άξονα της συχνότητας όσο και στον κατακόρυφο άξονα των dBFS. Ο σχεδιασμός του είναι orientation-agnostic, καθώς ο προσανατολισμός του δεν καθορίζεται ρητά, αλλά προκύπτει δυναμικά από τις διαστάσεις του component. Με τον τρόπο αυτό το ίδιο στοιχείο μπορεί να λειτουργήσει είτε οριζόντια είτε κατακόρυφα, χωρίς αλλαγή στη βασική του λογική.

Η γραφική του απεικόνιση περιλαμβάνει ένα κεντρικό track σταθερού πάχους, πάνω στο οποίο προβάλλεται το επιλεγμένο εύρος τιμών. Τα όρια του εύρους αναπαρίστανται μέσω δύο ανεξάρτητων knobs, τα οποία μπορούν να μετακινηθούν με το ποντίκι, επιτρέποντας στον χρήστη να ορίσει δυναμικά τα επιθυμητά όρια. Στον άξονα των dBFS, το εύρος επιλογής εκτείνεται από –80 έως 0 dBFS, ενώ στον άξονα της συχνότητας από 20 Hz έως 20 kHz.

Ιδιαίτερη σημασία έχει η υλοποίηση μηχανισμού ελάχιστης απόστασης μεταξύ των δύο ορίων. Το ελάχιστο αυτό όριο ορίζεται ως ποσοστό του συνολικού εύρους και μπορεί να διαφοροποιείται ανάλογα με τον προσανατολισμό του slider. Με τον τρόπο αυτό αποτρέπεται η πλήρης σύμπτωση των δύο τιμών, διασφαλίζοντας λειτουργική και σταθερή επιλογή περιοχής προβολής.

Πίνακας 5.3: Μέθοδοι Spectrum Analyzer

Κατηγορίες	Μέθοδοι	Λειτουργίες
Κατασκευή & Διάταξη	SpectrumAnalyzer(), ~SpectrumAnalyzer(), resized()	Δημιουργία αντικειμένου, ορισμός μεγέθους FFT, buffers και τοποθέτηση components
Διαχείριση Gradient	updateGradient()	Ενημέρωση χρωματικής διαβάθμισης για κανονική λειτουργία και bypass
Επεξεργασία Σήματος	processFFT(), pushBuffer(), clear()	Επεξεργασία FFT, εισαγωγή audio δεδομένων και εκκαθάριση buffers
Παραμετροποίηση	setBypass(), setColorMode(), setSampleRate()	Ρύθμιση λειτουργικών παραμέτρων

Καμπύλη Συμπίεσης	setCompressionCurve(), showCompressionCurve(), hideCompressionCurve(), setShowCurve()	Υπολογισμός και έλεγχος εμφάνισης καμπύλης
Περιοδική ανανέωση	timerCallback()	Περιοδική ενεργοποίηση FFT και ενημέρωση
Αλληλεπίδραση Χρήστη	mouseMove(), mouseDrag(), mouseExit(), updateTooltip()	Παρακολούθηση ποντικιού, υπολογισμός τιμών, ενημέρωση tooltip
Απεικόνιση	paint(), drawGrid(), drawFrame(), drawCurve(), drawTooltip(), drawCrosshair()	Σχεδίαση όλων των οπτικών στοιχείων

5.5.4 Signal Display

Η κλάση SimpleWaveformDisplay υλοποιεί ένα γραφικό στοιχείο, το οποίο απεικονίζει σε πραγματικό χρόνο τις κυματομορφές εισόδου και εξόδου του compressor, καθώς και την οπτική ένδειξη της gain reduction. Η απεικόνιση πραγματοποιείται μέσω της συνάρτησης paint(), ενώ η ενημέρωση των δεδομένων γίνεται περιοδικά μέσω μηχανισμού timer και της timerCallback(), εξασφαλίζοντας ομαλή και σταθερή ανανέωση.

Για την αποθήκευση και διαχείριση των δεδομένων, η κλάση χρησιμοποιεί δύο κυκλικά buffers, ένα για το σήμα εισόδου και ένα για το σήμα εξόδου μετά τη συμπίεση. Τα buffers αυτά διατηρούν ένα χρονικό παράθυρο του σήματος, επιτρέποντας την απεικόνιση της ιστορικής εξέλιξης της κυματομορφής.

Τα νέα δείγματα εισάγονται συνεχώς μέσω της συνάρτησης pushAudioBuffers(), με μηχανισμό κυκλικής εγγραφής, ώστε να διατηρείται σταθερό το μέγεθος του αποθηκευμένου ιστορικού. Σε περίπτωση επαναφοράς της ένδειξης, η συνάρτηση clear() μηδενίζει τα buffers. Τα δεδομένα του σήματος που αποθηκεύονται στα buffers βασίζονται στην ίδια κλίμακα που χρησιμοποιήθηκε στις προηγούμενες υλοποιήσεις, με τιμές αναφοράς [-1.0, 1.0].

Το μέγεθος των buffers ορίζεται σταθερά σε 44100 δείγματα, διατηρώντας ένα συνεχώς ενημερωμένο ιστορικό τιμών του σήματος. Η χρονική έκταση της κυματομορφής που εμφανίζεται στην οθόνη δεν είναι απόλυτα σταθερή, αλλά εξαρτάται από το πλήθος των διαθέσιμων δειγμάτων στον buffer και από τη χαρτογράφησή τους στο πλάτος της περιοχής σχεδίασης.

Για την επιβράδυνση και την αποδοτική σχεδίαση της κυματομορφής στην οθόνη εφαρμόζεται διαδικασία μείωσης δειγμάτων (downsampling). Δεδομένου ότι ο αριθμός των δειγμάτων στο buffer μπορεί να είναι πολύ μεγαλύτερος από τον αριθμό των pixel της οθόνης, τα δείγματα ομαδοποιούνται σε μικρά τμήματα. Για κάθε τμήμα υπολογίζονται οι μέγιστες και ελάχιστες τιμές πλάτους (5.18):

$$x_{max} = \max(x[n])$$
$$x_{min} = \min(x[n]) \quad (5.18)$$

Οι τιμές αυτές χρησιμοποιούνται για τον καθορισμό των άνω και κάτω ορίων της κυματομορφής στο αντίστοιχο pixel της οθόνης. Με τον τρόπο αυτό αποτυπώνονται οι κορυφές του σήματος χωρίς απώλεια σημαντικής πληροφορίας, ακόμη και όταν ο αριθμός των δειγμάτων είναι μεγάλος.

Η μετατροπή των τιμών πλάτους σε συντεταγμένες οθόνης πραγματοποιείται μέσω γραμμικής χαρτογράφησης (5.19):

$$y = y_{center} - x[n] \cdot H \tag{5.19}$$

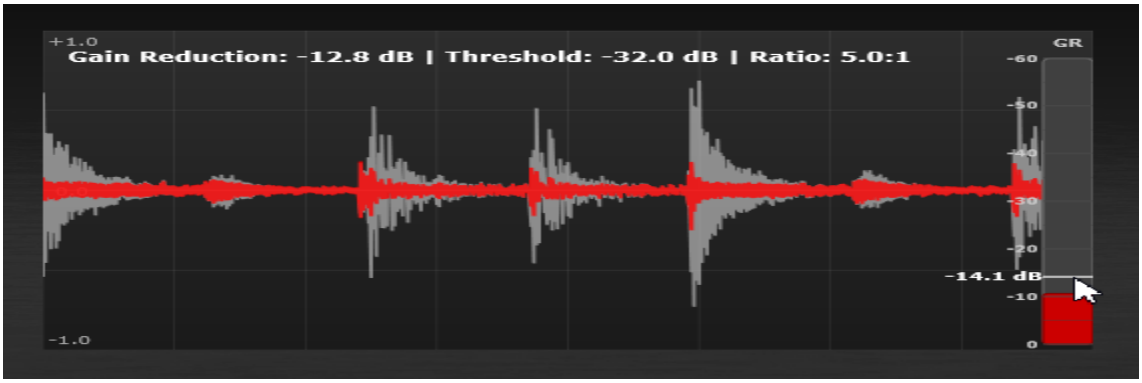
όπου y_{center} είναι το κέντρο της περιοχής σχεδίασης και H ένας συντελεστής κλιμάκωσης που καθορίζει το ύψος της κυματομορφής.

Η απεικόνιση των κυματομορφών πραγματοποιείται με χρήση δύο διακριτών χρωμάτων και υλοποιείται μέσω της συνάρτησης drawWaveform(). Η κυματομορφή εισόδου εμφανίζεται με ανοιχτό γκρι χρώμα και λειτουργεί ως σημείο αναφοράς για το αρχικό ανεπεξέργαστο σήμα, ενώ η κυματομορφή εξόδου αποδίδεται με κόκκινο χρώμα, επισημαίνοντας το επεξεργασμένο σήμα. Η σχεδίαση βασίζεται στον υπολογισμό των μέγιστων και ελάχιστων τιμών των δειγμάτων σε κάθε οριζόντια θέση, προσφέροντας μια σαφή και αντιπροσωπευτική απεικόνιση της δυναμικής του σήματος.

Στο υπόβαθρο του component σχεδιάζεται πλέγμα αναφοράς μέσω της συνάρτησης drawGrid(), το οποίο περιλαμβάνει οριζόντιες και κατακόρυφες βοηθητικές γραμμές καθώς και βασικές ενδείξεις πλάτους +1.0, 0.0, -1.0, οι οποίες αντιστοιχούν στο κανονικοποιημένο εύρος τιμών του ψηφιακού σήματος. Το πλέγμα διευκολύνει την οπτική εκτίμηση του πλάτους και της συμμετρίας του σήματος.

Η ένδειξη της gain reduction παρουσιάζεται μέσω κατακόρυφης μπάρας στη δεξιά πλευρά του component και σχεδιάζεται από τη συνάρτηση drawGainReductionMeter(). Η κίνηση της μπάρας ακολουθεί μηχανισμό fast attack – slow release, ώστε οι μειώσεις στάθμης λόγω συμπίεσης να αποτυπώνονται άμεσα, ενώ η επιστροφή στην αρχική κατάσταση να πραγματοποιείται πιο ομαλά. Παράλληλα, η αριθμητική ένδειξη της gain reduction ενημερώνεται σε χαμηλότερη συχνότητα προκειμένου να παραμένει ευανάγνωστη και οπτικά σταθερή.

Κατά την μετακίνηση του δείκτη πάνω στη μπάρα gain reduction, εμφανίζεται μια οριζόντια γραμμή που ακολουθεί την θέση του ποντικιού και προβάλλεται η αντίστοιχη τιμή σε dB. Όταν ο δείκτης εξέλθει από το πεδίο της μπάρας, η ένδειξη αποκρύπτεται. Η λειτουργία αυτή βασίζεται στις μεθόδους mouseMove() και mouseExit().



Σχήμα 5.9: Κανονική Λειτουργία Signal Display

Στο επάνω μέρος του component εμφανίζονται πληροφορίες για την κατάσταση του compressor, όπως η τρέχουσα τιμή gain reduction, το threshold και ο λόγος συμπίεσης μέσω της συνάρτησης drawCompressionInfo(). Η ενημέρωση των παραμέτρων αυτών, πραγματοποιείται μέσω της συνάρτησης setCompressionParameters(), η οποία αποθηκεύει τις νέες τιμές, ώστε να χρησιμοποιηθούν στην επόμενη ανανέωση της απεικόνισης. Με τον τρόπο αυτό διασφαλίζεται ο συγχρονισμός μεταξύ της επεξεργασίας ήχου και της γραφικής αναπαράστασης.

Σε περίπτωση που ο compressor βρίσκεται σε κατάσταση bypass, οι παραπάνω πληροφορίες δεν εμφανίζονται. Αντίθετα, εμφανίζεται το raw σήμα εισόδου με γκρι χρώμα και ειδικό overlay με την ένδειξη “BYPASS”, το οποίο υλοποιείται μέσω της συνάρτησης drawBypassOverlay().

Σχήμα 5.10: Bypass Λειτουργία Signal Display

Πίνακας 5.4: Μέθοδοι SimpleWaveformDisplay

Κατηγορίες	Μέθοδοι	Λειτουργίες
Κατασκευή	SimpleWaveformDisplay()	Δημιουργία circular buffers
Διαχείριση Δεδομένων	pushAudioBuffers(), pushInputBufferOnly(), clear(), clearOutputOnly()	Ενημέρωση buffers με νέα δείγματα, εκκαθάριση δεδομένων
Παραμετροποίηση	setCompressionParameters(), setBypass(), setPlayhead()	Ρύθμιση threshold, ratio, GR, bypass και playhead
Περιοδική ανανέωση	timerCallback()	Περιοδική ενημέρωση ενδείξεων
Απεικόνιση	paint(), drawGrid(), drawWaveform(), drawGainReductionMeter(), drawBypassOverlay(), drawCompressionInfo()	Σχεδίαση των οπτικών στοιχείων
Αλληλεπίδραση Χρήστη	mouseMove(), mouseExit()	Παρακολούθηση ποντικιού για tooltip

5.5.5 Knobs

Η κλάση CustomRotarySlider υλοποιεί ένα προσαρμοσμένο rotary slider που κληρονομεί από το juce::Slider. Στόχος του component είναι η αντικατάσταση του τυπικού rotary knob του JUCE, προσφέροντας βελτιωμένη οπτική αναπαράσταση και πιο ακριβή έλεγχο των παραμέτρων του compressor.

Κατά την αρχικοποίηση του slider, εισάγεται εικόνα τύπου image strip, όπου κάθε καρέ αντιστοιχεί σε μία θέση του knob. Η εικόνα εισάγεται είτε από τους ενσωματωμένους πόρους της εφαρμογής είτε από το τοπικό σύστημα αρχείων. Εφόσον η εικόνα είναι διαθέσιμη, υπολογίζεται ο συνολικός αριθμός των καρέ και το ύψος κάθε καρέ, ώστε η περιστροφή του knob να αντιστοιχεί σωστά στις τιμές της παραμέτρου. Σε περίπτωση που δεν είναι δυνατή η φόρτωση εικόνας, το slider εξακολουθεί να λειτουργεί χρησιμοποιώντας την προεπιλεγμένη εμφάνιση του JUCE.

Το component υποστηρίζει διαφορετικούς τύπους παραμέτρων, όπως input και output σε dB, attack και release σε milliseconds, threshold σε dB και ratio για τον λόγο συμπίεσης. Η σχεδίαση του knob πραγματοποιείται μέσω της μεθόδου paint(). Για τις περισσότερες παραμέτρους εφαρμόζεται γραμμική χαρτογράφηση μεταξύ τιμής και γωνίας περιστροφής, ενώ για την παράμετρο ratio χρησιμοποιείται ειδική χαρτογράφηση μέσω της μεθόδου mapRatioValueToAngle().

Πάνω από το knob προβάλλεται η ονομασία της παραμέτρου με έντονη γραμματοσειρά, προσφέροντας σαφή ένδειξη της λειτουργίας κάθε ελέγχου. Ο οπτικός σχεδιασμός του knob περιλαμβάνει tick marks γύρω από την περιφέρεια του κύκλου, με τα major ticks να αποδίδονται πιο έντονα και τα minor ticks λεπτότερα, παρέχοντας σαφή ένδειξη των βασικών θέσεων. Κάθε major tick συνοδεύεται από ετικέτα τιμής, η οποία προσαρμόζεται δυναμικά ανάλογα με τον τύπο της παραμέτρου.

Επιπλέον, το component υποστηρίζει την εισαγωγή τιμών μέσω text box, επιτρέποντας στον χρήστη να πληκτρολογεί απευθείας την επιθυμητή τιμή. Η μετατροπή μεταξύ αριθμητικής τιμής και κειμένου υλοποιείται μέσω των μεθόδων getTextFromValue() και getValueFromText(), λαμβάνοντας υπόψη την μονάδα μέτρησης και τις ιδιαιτερότητες κάθε παραμέτρου. Η διάταξη του text box ελέγχεται από τη μέθοδο resized(), η οποία διασφαλίζει ότι παραμένει σταθερά τοποθετημένο και κεντραρισμένο, ανεξάρτητα από τις διαστάσεις του slider.

Σχήμα 5.11: Compressor Knob [53]

Πίνακας 5.5: Μέθοδοι CustomRotarySlider

Μέθοδοι	Περιγραφή
Paint()	Σχεδίαση knob
getTextFromValue()	Μετατροπή τιμής σε κείμενο
getValueFromText()	Μετατροπή κειμένου σε τιμή
mapRatioValueToAngle()	Χαρτογράφηση ratio–γωνίας
Resized()	Διαχείριση διάταξης διεπαφής

Για την εμφάνιση του component χρησιμοποιείται προσαρμοσμένο BoldLookAndFeel, το οποίο υλοποιείται ως εσωτερικό struct της κλάσης. Το BoldLookAndFeel ρυθμίζει τα χρώματα, την γραμματοσειρά και τις διαστάσεις των labels και του text box, διατηρώντας μία καθαρή και minimal αισθητική. Παράλληλα, αναλαμβάνει την σωστή εμφάνιση του label κατά την επεξεργασία τιμών, εξασφαλίζοντας ομοιομορφία.

Μέσω των μεθόδων fillTextEditorBackground() και drawTextEditorOutline() διασφαλίζεται ότι το text box έχει διαφανές φόντο και προσαρμοσμένο περίγραμμα, επιτρέποντας την ομαλή ενσωμάτωσή του στην οπτική σχεδίαση, ενώ η getLabelBorderSize() μηδενίζει το περιθώριο γύρω από το label. Η getLabelFont() ορίζει έντονη γραμματοσειρά για τα labels και η drawLabel() αναλαμβάνει την απεικόνισή και το μέγεθος τους με κατάλληλες διαστάσεις.

Η περιστροφή του knob πραγματοποιείται μέσω drag του ποντικιού. Επιπλέον, υποστηρίζεται και μεταβολή της τιμής μέσω της ροδέλας, επιτρέποντας πιο λεπτομερή και ελεγχόμενη ρύθμιση της παραμέτρου.

Πίνακας 5.6: Μέθοδοι Struct BoldLookAndFeel

Μέδοθοι	Περιγραφή
getLabelFont()	Διαμόρφωση έντονης γραμματοσειράς
drawLabel()	Εμφάνιση ετικετών με προσαρμοσμένο layout
fillTextEditorBackground()	Δημιουργία διαφανούς υποβάθρου για text editor
drawTextEditorOutline()	Αφαίρεση περιγραμμάτων
getLabelBorderSize()	Μηδενισμός περιθωρίων για μέγιστη χωρητικότητα

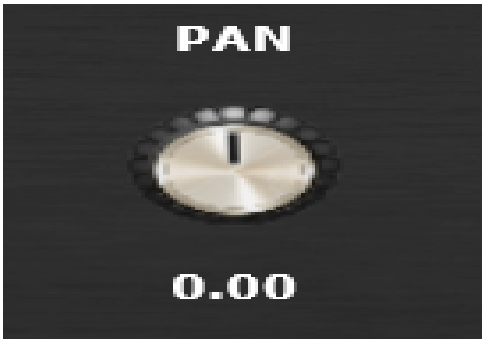
Για την παράμετρο panning χρησιμοποιείται ξεχωριστή κλάση, η PanKnob, η οποία επίσης κληρονομεί από την juce::Slider, αλλά υλοποιείται ανεξάρτητα από την κλάση CustomRotarySlider, καθώς εξυπηρετεί διαφορετικές λειτουργικές και οπτικές απαιτήσεις. Η κλάση αυτή έχει σχεδιαστεί ειδικά για τον έλεγχο της στερεοφωνικής θέσης του σήματος και χρησιμοποιεί επίσης image strip απεικόνιση για την οπτική αναπαράσταση της θέσης του επιλογέα.

Το PanKnob λειτουργεί σε εύρος τιμών από -1.0 έως 1.0, όπου οι ακραίες τιμές αντιστοιχούν στην πλήρη μετατόπιση προς το αριστερό ή το δεξί κανάλι, ενώ η μηδενική τιμή αντιστοιχεί στην κεντρική θέση. Η σχεδίασή του περιλαμβάνει τίτλο, προσαρμοσμένο text box για την εμφάνιση της τιμής, καθώς και custom LookAndFeel, το οποίο εξασφαλίζει ομοιογενή εμφάνιση με τα υπόλοιπα στοιχεία του γραφικού περιβάλλοντος. Με τον τρόπο αυτό, το PanKnob ενσωματώνεται λειτουργικά και αισθητικά στο σύστημα, διατηρώντας ταυτόχρονα την απαραίτητη αυτονομία ως ξεχωριστό component.

Πίνακας 5.7: Μέθοδοι PanKnob

Μέθοδοι	Περιγραφή
Paint()	Σχεδίαση του knob και του τίτλου του component
getTextBoxBounds()	Υπολογισμός της περιοχής εμφάνισης του text box
setStripImage()	Ορισμός της εικόνας και του αριθμού των frames
setStripImageFromMemory()	Φόρτωση εικόνας strip από μνήμη

Για την οπτική διαμόρφωση του text box και των labels, η κλάση PanKnob χρησιμοποιεί εσωτερικό BoldLookAndFeel, το οποίο ακολουθεί την ίδια σχεδιαστική λογική με το αντίστοιχο BoldLookAndFeel της κλάσης CustomRotarySlider, προσαρμοσμένο στις ανάγκες του συγκεκριμένου component.



Σχήμα 5.12: Pan Knob [53]

5.5.6 Buttons και Switches

Τα κουμπιά και οι διακόπτες του γραφικού περιβάλλοντος έχουν υλοποιηθεί με στόχο να προσφέρουν σαφή πληροφόρηση, άμεση οπτική ανάδραση και αισθητική συνοχή με το υπόλοιπο interface του compressor. Για τον σκοπό αυτό, χρησιμοποιούνται custom στοιχεία διεπαφής βασισμένα σε μηχανισμό toggle, επιτρέποντας την εναλλαγή μεταξύ ενεργής και ανενεργής κατάστασης.

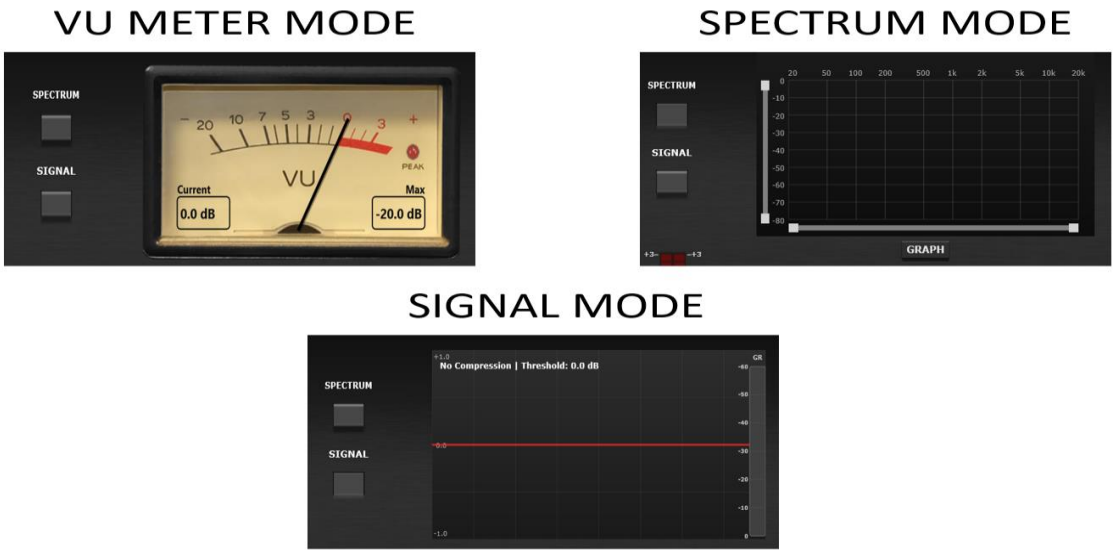
Στην περίπτωση της κλάσης LogicProStyleButton, τα κουμπιά IN, OUT, GR και GRAPH που εμφανίζονται στο VU meter και στο Spectrum Analyzer λειτουργούν ως διακόπτες επιλογής λειτουργίας. Η ίδια σχεδίαση εφαρμόζεται και στα standalone buttons καθώς και στο button ενεργοποίησης της multi-band λειτουργίας. Η ενεργή κατάσταση κάθε κουμπιού καθορίζεται μέσω της μεθόδου getToggleState(), η οποία επιστρέφει true όταν το κουμπί είναι ενεργοποιημένο και false όταν είναι απενεργοποιημένο. Αυτή η τιμή χρησιμοποιείται στη μέθοδο paint() όπου είναι υπεύθυνη για τις γραφικές αλλαγές, όπως μεταβολή μεγέθους, αλλαγή χρώματος και προσθήκη σκίασης.

Όταν ένα κουμπί ενεργοποιείται, εμφανίζεται ελαφρώς μεγεθυμένο μέσω κεντρικού scaling, δημιουργώντας την αίσθηση φυσικού πατήματος, ενώ το φόντο του γίνεται πιο σκούρο και το χρώμα του κειμένου πιο έντονο. Αντίθετα, στην ανενεργή κατάσταση, τα κουμπιά διατηρούν πιο ουδέτερη εμφάνιση, με διακριτικό highlight στο επάνω μέρος και σκιά στο κάτω, ενισχύοντας το τρισδιάστατο οπτικό αποτέλεσμα. Η χρήση σκιάσεων, περιγραμμάτων και ελαφρών μετατοπίσεων κατά την ενεργοποίηση προσδίδει βάθος και ρεαλισμό στη διεπαφή, προσομοιώνοντας φυσικά hardware κουμπιά.

Σχήμα 5.13: Buttons κλάσης LogicProStyleButton

Το interface περιλαμβάνει επίσης διακόπτες τύπου PNGToggleButton, οι οποίοι βασίζονται σε strip images και χρησιμοποιούνται για τις λειτουργίες spectrum analyzer και signal viewer. Κάθε toggle button εναλλάσσεται μεταξύ δύο frames, όπου το πρώτο αντιστοιχεί στην ανενεργή και το δεύτερο στην ενεργή κατάσταση. Η μέθοδος paint() επιλέγει το κατάλληλο frame με βάση την τρέχουσα τιμή toggle μέσω της μεθόδου getToggleState(), εξασφαλίζοντας σαφή οπτική ένδειξη της επιλογής του χρήστη. Η φόρτωση των εικόνων γίνεται μέσω των μεθόδων setStripImage() και setStripImageFromMemory(), που επιτρέπουν τον καθορισμό της εικόνας strip και του πλήθους των frames.

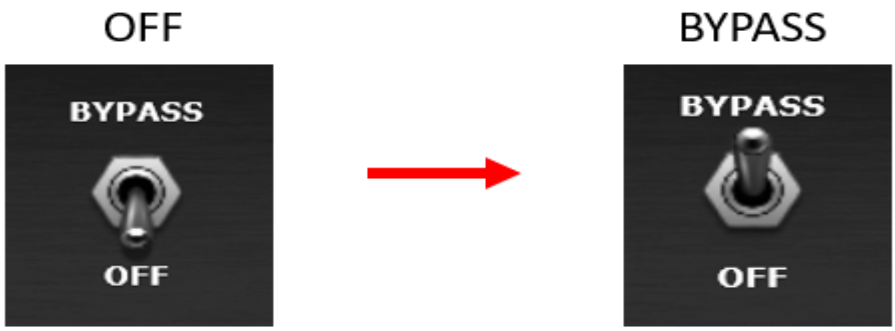
Η προεπιλεγμένη κατάσταση του συστήματος είναι η εμφάνιση του VU meter, το οποίο παραμένει ενεργό όταν δεν έχει επιλεχθεί κάποια άλλη λειτουργία. Με την ενεργοποίηση ενός εκ των PNG toggle buttons, η διεπαφή μεταβαίνει στην αντίστοιχη προβολή spectrum analyzer ή signal viewer. Η λογική λειτουργίας των διακοπτών είναι mutually exclusive, διασφαλίζοντας ότι μόνο μία λειτουργία απεικόνισης μπορεί να είναι ενεργή κάθε χρονική στιγμή. Έτσι, η επιλογή ενός mode απενεργοποιεί αυτόματα οποιαδήποτε προηγούμενη επιλογή.



Σχήμα 5.14: PNGToggleButton Modes [53]

Το Bypass Switch αποτελεί στοιχείο διεπαφής, το οποίο επιτρέπει στον χρήστη να ενεργοποιεί ή να απενεργοποιεί άμεσα την επεξεργασία του compressor. Η λειτουργία του βασίζεται σε μηχανισμό toggle, προσφέροντας δύο σαφείς και διακριτές καταστάσεις όπου είναι η ενεργή επεξεργασία και η παράκαμψη. Με αυτόν τον τρόπο, ο χρήστης μπορεί να συγκρίνει εύκολα το επεξεργασμένο και το μη επεξεργασμένο σήμα, χωρίς να διακόπτεται η ροή εργασίας.

Σε επίπεδο γραφικής υλοποίησης, ο διακόπτης χρησιμοποιεί εικόνα τύπου strip, η οποία περιλαμβάνει δυο frames, καθένα από τα οποία αντιστοιχεί σε διαφορετική λειτουργική κατάσταση. Συγκεκριμένα, το πρώτο frame απεικονίζει την κατάσταση bypass, ενώ το δεύτερο frame αντιστοιχεί στην κανονική λειτουργία του compressor. Ανάλογα με την κατάσταση του διακόπτη, προβάλλεται το αντίστοιχο τμήμα της εικόνας, εξασφαλίζοντας άμεση οπτική ένδειξη της τρέχουσας κατάστασης.



Σχήμα 5.15: Bypass Switch [53]

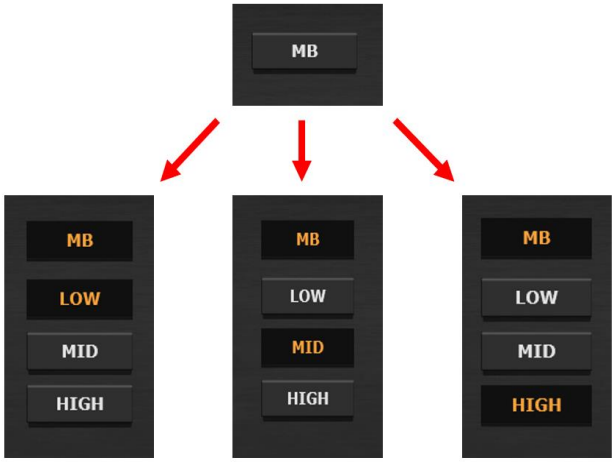
Πίνακας 5.8: Μέθοδοι Buttons

Κλάση	Μέθοδοι
LogicProStyleButton	paint(), getToggleState()
PNGToggleButton	paint(), setStripImage(), getToggleState(), setStripImageFromMemory()
BypassButton	paint(), setStripImage(), setStripImageFromMemory(), setStripImageFromFile(), getImage(), getFrameHeight(), getToggleState()

5.5.7 Multi-band Λειτουργία

Η υποστήριξη της multi-band λειτουργίας στο γραφικό περιβάλλον του συστήματος απαιτεί μία πιο σύνθετη διεπαφή ελέγχου σε σχέση με την single-band επεξεργασία, καθώς ο χρήστης πρέπει να έχει τη δυνατότητα ανεξάρτητης ρύθμισης πολλαπλών ζωνών συχνοτήτων. Για τον σκοπό αυτό υλοποιήθηκε η κλάση MultiBandEditorComponent, η οποία λειτουργεί ως εξειδικευμένο γραφικό component για τη διαχείριση των παραμέτρων της multi-band επεξεργασίας. Η κλάση κληρονομεί από juce::Component και juce::Timer, επιτρέποντας τόσο τη σχεδίαση του περιβάλλοντος όσο και την περιοδική ανανέωσή του.

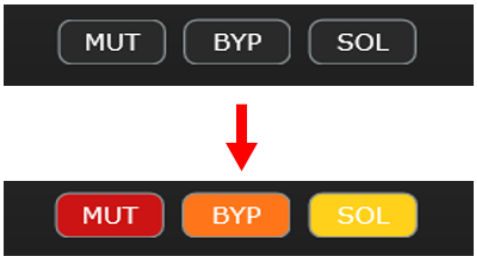
Η ενεργοποίηση της multi-band διεπαφής πραγματοποιείται μέσω ειδικού button επιλογής λειτουργίας. Όταν το button αυτό βρίσκεται σε ενεργή κατάσταση, το σύστημα μεταβαίνει σε multi-band mode και εμφανίζεται το MultiBandEditorComponent, μέσω του οποίου παρέχεται ανεξάρτητος έλεγχος των επιμέρους ζωνών συχνοτήτων. Με την ενεργοποίηση αυτή, εμφανίζονται buttons για τις τρεις διακριτές ζώνες Low, Mid και High. Αντίθετα, όταν η multi-band λειτουργία δεν είναι ενεργή, το σύστημα παραμένει σε single-band mode και χρησιμοποιείται η βασική διεπαφή παραμετροποίησης του compressor.



Σχήμα 5.16: Multi-band λειτουργίες

Το component οργανώνει τη διεπαφή σε τρεις διακριτές περιοχές, οι οποίες αντιστοιχούν στις ζώνες χαμηλών, μεσαίων και υψηλών συχνοτήτων. Για κάθε ζώνη παρέχονται ανεξάρτητοι περιστροφικοί επιλογείς για τις παραμέτρους threshold, attack, release και ratio, επιτρέποντας στον χρήστη να καθορίζει τη δυναμική συμπεριφορά κάθε band με αυτόνομο τρόπο.

Παράλληλα, για κάθε ζώνη διατίθενται τρία κουμπιά ελέγχου, BYP, MUT και SOL, τα οποία χρησιμοποιούνται αντίστοιχα για την bypass, mute και solo λειτουργία της συγκεκριμένης περιοχής συχνοτήτων. Η λειτουργία bypass επιτρέπει την παράκαμψη της συμπίεσης για τη συγκεκριμένη ζώνη, η λειτουργία mute απενεργοποιεί τον ήχο της ζώνης, ενώ η λειτουργία solo απομονώνει τη συγκεκριμένη ζώνη ώστε ο χρήστης να μπορεί να την ελέγξει ανεξάρτητα από τις υπόλοιπες. Με τον τρόπο αυτό, η διεπαφή προσφέρει άμεσο και στοχευμένο έλεγχο της κατάστασης κάθε band.



Σχήμα 5.17: Κουμπιά Bypass, Mute και Solo

Επιπλέον, το component περιλαμβάνει δύο γραμμικούς sliders για τον καθορισμό των συχνοτήτων crossover, δηλαδή των σημείων διαχωρισμού μεταξύ χαμηλής, μεσαίας και υψηλής ζώνης συχνοτήτων. Ο πρώτος slider ορίζει το όριο μεταξύ χαμηλής και μεσαίας μπάντας και ρυθμίζεται στο εύρος 20 Hz έως 1000 Hz, ενώ ο δεύτερος το όριο μεταξύ μεσαίας και υψηλής μπάντας και ρυθμίζεται στο εύρος 1000 Hz έως 20000 Hz. Με τον τρόπο αυτό, ο χρήστης μπορεί να ρυθμίζει δυναμικά τον φασματικό διαχωρισμό του σήματος και να προσαρμόζει τη λειτουργία της multi-band επεξεργασίας ανάλογα με τις απαιτήσεις.

Για την οπτική διαμόρφωση των συγκεκριμένων sliders χρησιμοποιείται το προσαρμοσμένο FrequencySliderLookAndFeel, το οποίο κληρονομεί από την κλάση juce::LookAndFeel_V4. Το struct αυτό καθορίζει την εμφάνιση των labels και του text box των sliders, χρησιμοποιώντας έντονη γραμματοσειρά, ώστε τα στοιχεία αυτά να ενσωματώνονται ομοιόμορφα στη συνολική αισθητική της διεπαφής.

Σχήμα 5.18: Sliders Συχνοτήτων

Πίνακας 5.9: Μέθοδοι FrequencySliderLookAndFeel

Μέθοδοι	Περιγραφή
FrequencySliderLookAndFeel()	Αρχικοποίηση χρωμάτων του text box και του text editor
getLabelFont()	Ορισμός έντονης γραμματοσειράς για τα labels
drawLabel()	Προσαρμοσμένη σχεδίαση των labels
fillTextEditorBackground()	Δημιουργία υποβάθρου για τον text editor
drawTextEditorOutline()	Κατάργηση του περιγράμματος του text editor
getLabelBorderSize()	Μηδενισμός των περιθωρίων γύρω από το label

Η αρχικοποίηση και η σύνδεση όλων των στοιχείων πραγματοποιείται στον constructor MultiBandEditorComponent(). Σε αυτό το στάδιο καθορίζονται τα εύρη τιμών των sliders, οι αρχικές τους τιμές, καθώς και τα χαρακτηριστικά των κουμπιών και των crossover controls. Παράλληλα, μέσω των SliderAttachment και ButtonAttachment, όλα τα στοιχεία της διεπαφής συνδέονται με τις αντίστοιχες παραμέτρους του AudioProcessorValueTreeState, διασφαλίζοντας αμφίδρομο συγχρονισμό μεταξύ του GUI και της μονάδας επεξεργασίας ήχου. Ο destructor ~MultiBandEditorComponent() εξασφαλίζει την ασφαλή αποδέσμευση των πόρων του component, ενώ η λειτουργία του timer διακόπτεται αυτόματα κατά την καταστροφή του αντικειμένου.

Η μέθοδος timerCallback() χρησιμοποιείται για την περιοδική ανανέωση του component, όταν το σύστημα λειτουργεί σε multi-band mode. Η προσέγγιση αυτή εξασφαλίζει ότι η οπτική κατάσταση της διεπαφής παραμένει ενημερωμένη και συμβατή με την τρέχουσα κατάσταση του processor. Παράλληλα, η μέθοδος paint() είναι υπεύθυνη για τη σχεδίαση του γραφικού περιβάλλοντος του component. Πιο συγκεκριμένα, σχεδιάζει το πλαίσιο του component, τα περιγράμματα, καθώς και τις σχετικές ενδείξεις των bands και των crossover περιοχών.

Η διάταξη όλων των στοιχείων ελέγχου πραγματοποιείται μέσω της μεθόδου resized(). Η μέθοδος αυτή υπολογίζει δυναμικά τις θέσεις και τις διαστάσεις των rotary sliders, των κουμπιών και των crossover sliders, με βάση το συνολικό μέγεθος του component. Τα στοιχεία κατανέμονται ανά ζώνη με οργανωμένο τρόπο, εξασφαλίζοντας ομοιομορφία και ευκολία στον χειρισμό της διεπαφής.

Τέλος, οι βοηθητικές μέθοδοι getAllSliders() και getAllButtons() χρησιμοποιούνται για τη συλλογή των αντίστοιχων controls σε ενιαίες δομές δεδομένων, διευκολύνοντας την αρχικοποίηση και την διαχείριση της διεπαφής. Η χρήση αυτών των βοηθητικών συναρτήσεων συμβάλλει στη βελτίωση της οργάνωσης του κώδικα και στη μείωση της επαναληψιμότητας.

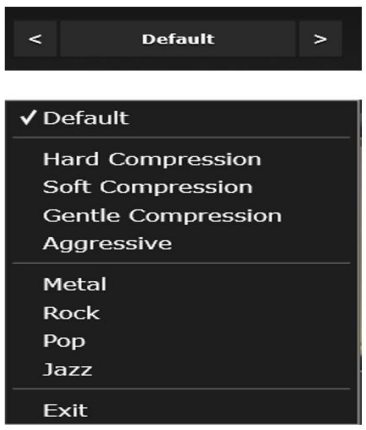
Πίνακας 5.10: Μέθοδοι MultiBandEditorComponent

Μέθοδοι	Περιγραφή
MultiBandEditorComponent(), ~MultiBandEditorComponent()	Αρχικοποίηση του component, δημιουργία controls
timerCallback()	Ανανέωση τιμών της διεπαφής
paint()	Σχεδίαση background και των οπτικών στοιχείων
resized()	Τοποθέτηση και διάταξη sliders, buttons
getAllSliders(), getAllButtons()	Συλλογή των controls για αρχικοποίηση και εμφάνιση

5.5.8 Σύστημα Presets

Το σύστημα προκαθορισμένων ρυθμίσεων-presets αποτελεί βασικό λειτουργικό τμήμα του compressor, καθώς επιτρέπει την ταχεία εφαρμογή των παραμέτρων που αντιστοιχούν σε συγκεκριμένα χαρακτηριστικά δυναμικής επεξεργασίας. Μέσω των presets, ο χρήστης μπορεί να επιλέξει άμεσα μία προκαθορισμένη συμπεριφορά του compressor, χωρίς να απαιτείται η χειροκίνητη ρύθμιση κάθε παραμέτρου ξεχωριστά, καλύπτοντας διαφορετικά σενάρια δυναμικής επεξεργασίας, από ήπια έως έντονη συμπίεση. Η προσέγγιση αυτή διευκολύνει σημαντικά τη χρήση του συστήματος, ιδιαίτερα σε περιπτώσεις όπου απαιτείται γρήγορη εναλλαγή μεταξύ διαφορετικών ρυθμίσεων.

Το σύστημα υλοποιείται μέσω της κλάσης PresetButton, η οποία λειτουργεί ως προσαρμοσμένο στοιχείο διεπαφής για την επιλογή και την εναλλαγή presets. Το component αποτελείται από τρία επιμέρους κουμπιά, ένα για μετάβαση στο προηγούμενο preset, ένα κεντρικό κουμπί που εμφανίζει το όνομα του τρέχοντος preset και ένα για μετάβαση στο επόμενο preset. Η εναλλαγή πραγματοποιείται είτε μέσω των πλευρικών κουμπιών είτε μέσω αναδυόμενου μενού που εμφανίζεται από το κεντρικό κουμπί, επιτρέποντας άμεση επιλογή από το σύνολο των διαθέσιμων presets.



Σχήμα 5.19: Preset Λειτουργία

Η κλάση διατηρεί εσωτερικά το currentPresetId και το currentPresetName, εξασφαλίζοντας ότι το στοιχείο διεπαφής παραμένει συγχρονισμένο με την τρέχουσα κατάσταση του συστήματος. Η επιλογή ενός preset ενεργοποιεί callback προς το editor, ώστε να εφαρμοστούν αυτόματα οι αντίστοιχες ρυθμίσεις του compressor.

Η εφαρμογή των presets υλοποιείται μέσω της μεθόδου applyPreset(int presetId), η οποία λειτουργεί ως το κεντρικό σημείο εφαρμογής των προκαθορισμένων ρυθμίσεων. Η μέθοδος λαμβάνει ως όρισμα το presetId και καλεί τις μεθόδους applySingleBandPreset() και applyMultiBandPreset(). Με τον τρόπο αυτό, το ίδιο preset εφαρμόζεται τόσο στη single-band όσο και στη multi-band λειτουργία, ενώ οι τιμές κατανέμονται στις αντίστοιχες παραμέτρους ανάλογα με τον τρόπο επεξεργασίας.

Στη single-band λειτουργία, οι τιμές του preset εφαρμόζονται στις βασικές παραμέτρους του compressor, όπως input, output, threshold, ratio, attack και release. Η διαδικασία υλοποιείται μέσω της applySingleBandPreset(), όπου κάθε preset αντιστοιχεί σε διαφορετική περίπτωση δομής switch και οι αντίστοιχες τιμές ανατίθενται στους sliders του compressor.

Στη multi-band λειτουργία, οι αντίστοιχες τιμές εφαρμόζονται ταυτόχρονα και στις τρεις ζώνες επεξεργασίας, χωρίς να μεταβάλλονται οι καταστάσεις bypass, mute και solo των bands κατά την αλλαγή preset. Η βοηθητική συνάρτηση setAllBands() εφαρμόζει τις ίδιες τιμές threshold, ratio, attack και release στις ζώνες Low, Mid και High, εξασφαλίζοντας κοινή δυναμική συμπεριφορά για όλα τα bands του επιλεγμένου preset. Η διαδικασία αυτή υλοποιείται στη applyMultiBandPreset(), όπου οι βοηθητικές συναρτήσεις setFloatParam() και setChoiceParam() ενημερώνουν τις αντίστοιχες παραμέτρους του AudioProcessorValueTreeState.

Ιδιαίτερο χαρακτηριστικό της υλοποίησης είναι ότι τα presets δεν τροποποιούν τα σημεία διαχωρισμού των crossover συχνοτήτων, γεγονός που επιτρέπει τον άμεσο έλεγχο των σημείων από τον χρήστη. Έτσι, το σύστημα presets επηρεάζει αποκλειστικά τη δυναμική συμπεριφορά της επεξεργασίας, ενώ ο φασματικός διαχωρισμός της multi-band λειτουργίας διατηρείται ανεξάρτητος.

Πίνακας 5.11: Μέθοδοι PresetButton

Μέθοδοι	Περιγραφή
PresetButton(),~PresetButton()	Αρχικοποίηση του component και αποδέσμευση
Paint()	Σχεδίαση των γραφικών
resized()	Διάταξη των κουμπιών
setPresetCallback()	Ορισμός callback για την εφαρμογή του επιλεγμένου preset
setCurrentPreset()	Ενημέρωση του τρέχοντος preset και του εμφανιζόμενου ονόματος
getPresetName()	Αντιστοίχιση αναγνωριστικού preset σε όνομα
applyPresetById()	Εφαρμογή preset με βάση το αναγνωριστικό
stepPreset()	Μετάβαση στο προηγούμενο ή στο επόμενο preset

showMenu()	Εμφάνιση αναδυόμενου μενού επιλογής preset

5.5.9 Standalone Λειτουργία

Στη standalone λειτουργία, το γραφικό περιβάλλον του συστήματος επεκτείνεται με επιπλέον στοιχεία ελέγχου, τα οποία υποστηρίζουν τη διαχείριση και την αναπαραγωγή αρχείων ήχου. Παρότι διατηρείται η βασική δομή του interface του compressor, προστίθενται ειδικά controls που επιτρέπουν στον χρήστη να αλληλεπιδρά με το ηχητικό υλικό χωρίς την ανάγκη host ή DAW.

Η διεπαφή περιλαμβάνει κουμπιά όπως Load, Play, Stop, Clear, Rewind, Forward και Export, τα οποία οργανώνονται ως λειτουργικά στοιχεία διαχείρισης αρχείων ήχου. Σε επίπεδο υλοποίησης, κάθε κουμπί συνδέεται με την αντίστοιχη λειτουργία μέσω του μηχανισμού onClick, ο οποίος εκτελεί συγκεκριμένη ενέργεια όταν ο χρήστης πατήσει το αντίστοιχο κουμπί.

Το Load χρησιμοποιείται για την επιλογή και φόρτωση αρχείου ήχου στο σύστημα σε μορφή WAV ή MP3 μέσω της μεθόδου loadFileToPlayer(), ενώ το Play καλεί τη μέθοδο playPauseFile() για την έναρξη ή παύση της αναπαραγωγής. Το Stop καλεί τη μέθοδο stopAndRewindFile(), διακόπτοντας την αναπαραγωγή και επαναφέροντας τη θέση του αρχείου στην αρχή. Τα Rewind και Forward επιτρέπουν τη χρονική μετακίνηση μέσα στο αρχείο με την μέθοδο seekFileBySeconds(), μεταβάλλοντας την τρέχουσα θέση αναπαραγωγής ανά 5 δευτερόλεπτα, ενώ το Clear καλεί τη μέθοδο clearLoadedFile(), αφαιρώντας το τρέχον αρχείο από τη standalone εφαρμογή. Τέλος, το Export επιτρέπει την αποθήκευση του επεξεργασμένου αποτελέσματος σε νέο αρχείο ήχου σε μορφή WAV μέσω της μεθόδου exportFile(). Με τον τρόπο αυτό, η standalone διεπαφή δεν περιορίζεται μόνο στην παραμετροποίηση του compressor, αλλά υποστηρίζει και τον συνολικό χειρισμό του ηχητικού υλικού.

Σχήμα 5.20: Standalone Buttons

Μετά τη φόρτωση αρχείου, εμφανίζεται ένδειξη χρόνου αναπαραγωγής, η οποία παρουσιάζει την τρέχουσα χρονική τιμή και τη συνολική διάρκεια του αρχείου. Η ένδειξη αυτή ενημερώνεται περιοδικά από το γραφικό περιβάλλον, ώστε να παραμένει συγχρονισμένη με την αναπαραγωγή του ηχητικού υλικού.

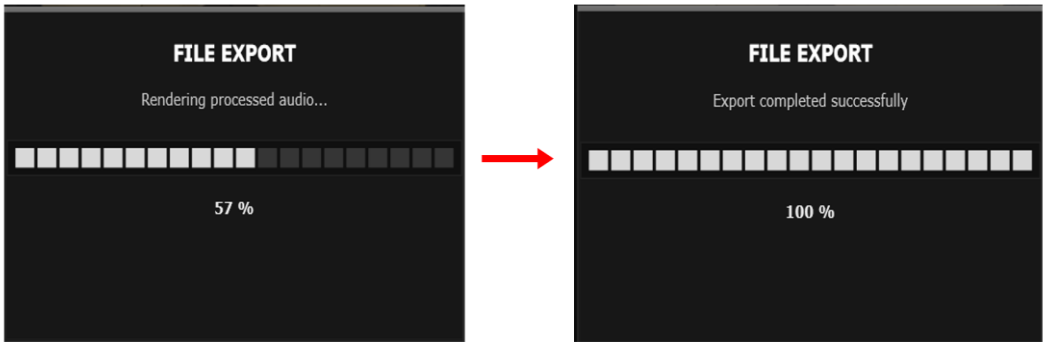
Για τον υπολογισμό της τρέχουσας χρονικής τιμής αξιοποιείται η μέθοδος getPlaybackPosition(), ενώ η συνολική διάρκεια προκύπτει από το φορτωμένο αρχείο ήχου. Οι τιμές χρόνου μετατρέπονται σε μορφή λεπτών και δευτερολέπτων, ώστε να εμφανίζονται με ευανάγνωστο τρόπο στον χρήστη. Με αυτόν τον τρόπο, ο χρήστης έχει άμεση χρονική αναφορά κατά την αναπαραγωγή και μπορεί να παρακολουθεί με μεγαλύτερη ακρίβεια την διάρκεια του ήχου.

Σχήμα 5.21: Ένδειξη time label

Η προσθήκη των συγκεκριμένων στοιχείων ελέγχου ενσωματώνεται στο υπάρχον γραφικό περιβάλλον με τρόπο που διατηρεί την οπτική συνοχή του συστήματος. Τα νέα κουμπιά ακολουθούν την ίδια σχεδιαστική λογική με τα υπόλοιπα buttons του interface, ώστε η standalone λειτουργία να αποτελεί φυσική επέκταση της βασικής διεπαφής και όχι ξεχωριστό περιβάλλον χρήσης. Έτσι, ο χρήστης μπορεί να μεταβαίνει από την παραμετροποίηση της επεξεργασίας στον χειρισμό της αναπαραγωγής και της εξαγωγής χωρίς να αλλάζει το συνολικό πλαίσιο αλληλεπίδρασης.

Στη standalone λειτουργία, η διαδικασία εξαγωγής του επεξεργασμένου ηχητικού σήματος περιλαμβάνει ειδική οπτική ένδειξη προόδου, ώστε ο χρήστης να ενημερώνεται άμεσα για την κατάσταση της λειτουργίας export. Η εξαγωγή ενεργοποιείται μέσω της μεθόδου exportFile(), ενώ η επεξεργασία του ηχητικού υλικού κατά την εξαγωγή πραγματοποιείται τμηματικά μέσω της processExportBlock(). Για την οπτική απεικόνιση της διαδικασίας χρησιμοποιείται το ExportOverlayComponent, το οποίο εμφανίζεται ως overlay πάνω από το κύριο γραφικό περιβάλλον.

Το ExportOverlayComponent περιλαμβάνει τίτλο, λεκτική ένδειξη κατάστασης, ποσοστιαία τιμή προόδου και γραφική μπάρα προόδου. Η οπτική απεικόνιση της προόδου υλοποιείται μέσω του βοηθητικού component SegmentedBar, το οποίο αποδίδει την πρόοδο με χρήση διακριτών τμημάτων, παρέχοντας σαφή και ευανάγνωστη ένδειξη της εξέλιξης της διαδικασίας. Η ενημέρωση της προόδου πραγματοποιείται περιοδικά μέσω Timer, ενώ η τρέχουσα τιμή λαμβάνεται από atomic μεταβλητή, εξασφαλίζοντας ασφαλή επικοινωνία μεταξύ της διαδικασίας εξαγωγής και του γραφικού περιβάλλοντος.



Σχήμα 5.22: Export Απεικόνιση

Πίνακας 5.12: Μέθοδοι ExportOverlayComponent

Μέθοδοι	Περιγραφή
ExportOverlayComponent()	Αρχικοποίηση του overlay, των labels και της μπάρας προόδου
setExportingState()	Ενημέρωση της διεπαφής για διαδικασία export
setCompletedState()	Ενημέρωση της διεπαφής για επιτυχημένη ολοκλήρωση export
setFailedState()	Ενημέρωση της διεπαφής για αποτυχία export
paint()	Σχεδίαση του overlay και του περιβάλλοντος ένδειξης προόδου
resized()	Τοποθέτηση των labels και της μπάρας προόδου στο overlay
timerCallback()	Ενημέρωση και επανασχεδίαση του component

5.5.10 Background

Το γραφικό περιβάλλον του συστήματος χρησιμοποιεί προσαρμοσμένο background με σκοπό τη δημιουργία μιας επαγγελματικής αισθητικής, εμπνευσμένης από αναλογικό audio hardware. Για τον λόγο αυτό, επιλέχθηκε εικόνα με μεταλλική υφή, η οποία έχει υποστεί επεξεργασία ώστε να εμφανίζεται σε πιο σκοτεινή απόχρωση. Η επεξεργασία της εικόνας περιλαμβάνει μείωση της φωτεινότητας και της αντίθεσης, καθώς και ήπια εξομάλυνση των λεπτομερειών, ώστε να αποφεύγεται η οπτική υπερφόρτωση.

Η σκοτεινή χρωματική παλέτα επιλέχθηκε σκόπιμα, καθώς συμβάλλει στην ανάδειξη των βασικών γραφικών στοιχείων, όπως το VU meter, το spectrum analyzer και τα buttons, διατηρώντας παράλληλα μια καθαρή και ισορροπημένη αισθητική διεπαφής. Επιπλέον, η μεταλλική υφή προσδίδει βάθος και αίσθηση υλικότητας στη διεπαφή, αποφεύγοντας την επίπεδη και μονότονη εμφάνιση.

Σε επίπεδο υλοποίησης, η εικόνα του background φορτώνεται από τους πόρους του project και σχεδιάζεται στη διεπαφή μέσω της μεθόδου paint(), αποτελώντας σταθερό οπτικό υπόβαθρο για όλα τα επιμέρους components.

5.6 Αρχιτεκτονική του Συστήματος

Στο παρόν κεφάλαιο παρουσιάζεται η αρχιτεκτονική του συστήματος και αναλύεται ο τρόπος με τον οποίο δομούνται και αλληλεπιδρούν τα βασικά υποσυστήματα. Η αρχιτεκτονική βασίζεται στον διαχωρισμό μεταξύ DSP, διαχείρισης παραμέτρων και GUI, σύμφωνα με τις αρχές ανάπτυξης real-time audio εφαρμογών.

Στο σύστημα αυτό, τα τέσσερα βασικά αρχεία έχουν διαφορετικούς αλλά αλληλοσυμπληρούμενους ρόλους. Το PluginProcessor.h και PluginProcessor.cpp αποτελούν τον πυρήνα της επεξεργασίας ήχου, με το header (PluginProcessor.h) να δηλώνει τη δομή, τις κλάσεις και τις μεθόδους, ενώ το source (PluginProcessor.cpp) υλοποιεί την πλήρη DSP λογική και τη διαχείριση των παραμέτρων. Αντίστοιχα, το PluginEditor.h και PluginEditor.cpp διαχειρίζονται το γραφικό περιβάλλον του συστήματος, με το header (PluginEditor.h) να ορίζει την κλάση του editor και τα μέλη της και το source (PluginEditor.cpp) να υλοποιεί τη δημιουργία, τη διάταξη και τη δυναμική ενημέρωση όλων των GUI components.

Στις επόμενες ενότητες παρουσιάζεται αναλυτικά η δομή και η λειτουργία αυτών των τεσσάρων αρχείων, με στόχο την πλήρη κατανόηση της συνολικής αρχιτεκτονικής.

5.6.1 Ανάλυση PluginProcessor.cpp

Το αρχείο PluginProcessor.cpp αποτελεί τον κεντρικό πυρήνα λειτουργίας του συστήματος, καθώς είναι υπεύθυνο για την επεξεργασία του ηχητικού σήματος σε πραγματικό χρόνο και για τον συντονισμό της επικοινωνίας μεταξύ του audio thread, της γραφικής διεπαφής χρήστη και του host. Στο πλαίσιο αυτό, η κλάση PtyxiakhCompressorAudioProcessor, που κληρονομεί από την juce::AudioProcessor, συγκεντρώνει το σύνολο των βασικών λειτουργιών του συστήματος, υλοποιώντας τόσο τη ροή επεξεργασίας του σήματος όσο και τη διαχείριση των παραμέτρων και της κατάστασης του plugin.

Ο constructor της κλάσης είναι υπεύθυνος για την αρχικοποίηση της δομής του συστήματος. Κατά τη δημιουργία του αντικειμένου, ορίζονται οι παράμετροι του compressor μέσω του μηχανισμού AudioProcessorValueTreeState και της μεθόδου createParameterLayout(). Οι παράμετροι αυτές ορίζονται με συγκεκριμένα εύρη τιμών και κατάλληλους τύπους, επιτρέποντας ακριβή έλεγχο, συμβατότητα με το host και συγχρονισμό με το γραφικό περιβάλλον.

Οι παράμετροι που ορίζονται στη createParameterLayout() οργανώνονται σε βασικές κατηγορίες. Αρχικά, υπάρχουν οι γενικές παράμετροι επεξεργασίας, όπως Input, Output, InputPan και Bypassed, οι οποίες επηρεάζουν τη συνολική ροή του σήματος ανεξάρτητα από τον τρόπο λειτουργίας. Στη συνέχεια, ορίζονται οι βασικές παράμετροι της single-band συμπίεσης, όπως Threshold, Attack, Release και Ratio. Παράλληλα, περιλαμβάνονται παράμετροι που σχετίζονται με την κατάσταση και την απεικόνιση του γραφικού περιβάλλοντος, όπως ViewMode, WaveformView, VUMeterMode και MultiBandMode.

Για τη multi-band λειτουργία ορίζεται ξεχωριστό σύνολο παραμέτρων, ώστε κάθε ζώνη συχνοτήτων να μπορεί να ελέγχεται ανεξάρτητα. Οι παράμετροι LowMidCrossover και MidHighCrossover καθορίζουν τα σημεία διαχωρισμού του σήματος σε χαμηλή, μεσαία και υψηλή ζώνη, ενώ για κάθε ζώνη υπάρχουν ανεξάρτητες ρυθμίσεις Threshold, Attack, Release και Ratio που καθορίζουν τη συμπεριφορά της συμπίεσης. Επιπλέον, οι παράμετροι Bypassed, Mute και Solo επιτρέπουν την παράκαμψη, τη σίγαση ή την απομόνωση κάθε ζώνης κατά την επεξεργασία.

Πίνακας 5.13: Μέθοδοι CreateParameterLayout()

ID	Τύπος	Εύρος Τιμών
Input, Output	AudioParameterFloat	-24dB έως +24dB
Threshold	AudioParameterFloat	-60dB έως 0dB
Attack	AudioParameterFloat	5ms έως 500ms
Release	AudioParameterFloat	5ms έως 500ms
Ratio	AudioParameterChoice	1:1 έως 20:1

InputPan	AudioParameterFloat	-1.0 έως +1.0
LowMidCrossover	AudioParameterFloat	20 Hz έως 1000 Hz
MidHighCrossover	AudioParameterFloat	1000 Hz έως 20000 Hz
Bypassed, Mute, Solo	AudioParameterBool	true/false
ViewMode, WaveformView, MultiBandMode	AudioParameterBool	true/false
VuMeterMode	AudioParameterChoice	GR / IN / OUT

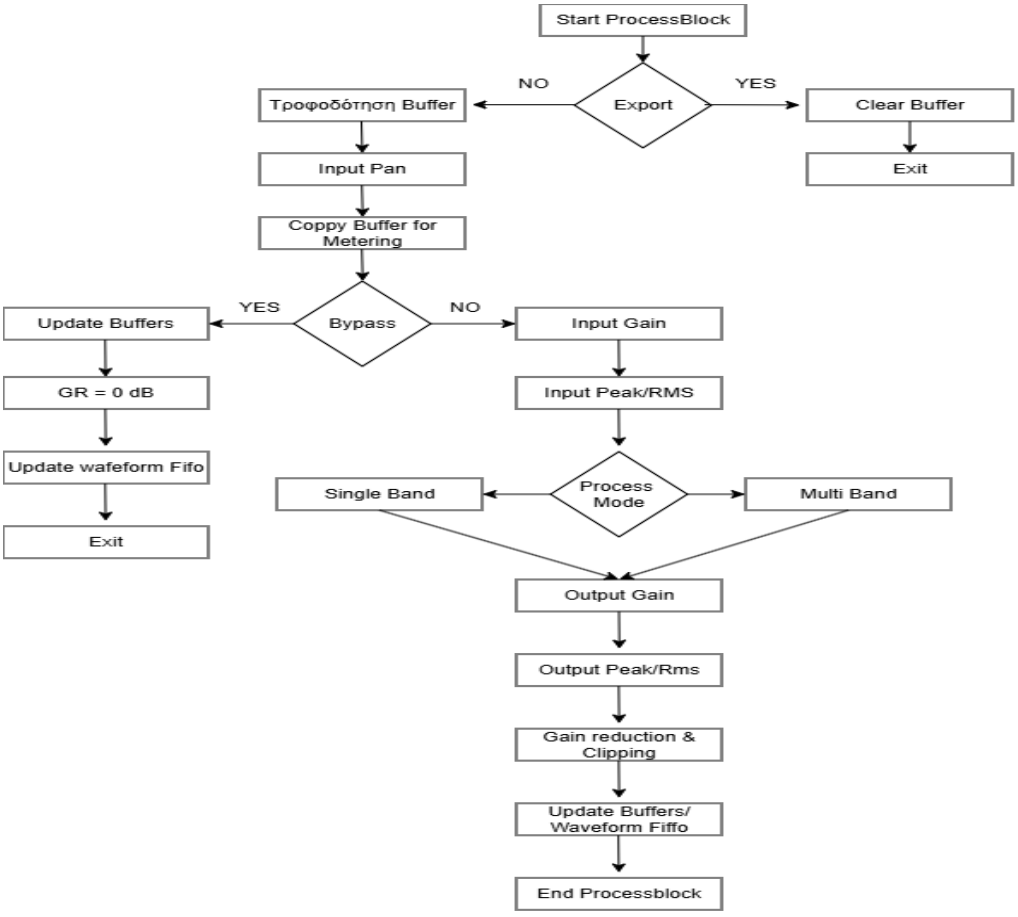
Η μέθοδος prepareToPlay() καλείται πριν από την έναρξη της αναπαραγωγής και χρησιμοποιείται για την προετοιμασία των DSP μονάδων. Σε αυτό το στάδιο αρχικοποιούνται ο single-band compressor, η multi-band μονάδα επεξεργασίας, τα στάδια Input Gain και Output Gain, καθώς και ο μηχανισμός oversampling για True Peak επεξεργασία, σύμφωνα με το sample rate, το μέγεθος του buffer και τον αριθμό καναλιών του συστήματος.

Παράλληλα, ενεργοποιείται μηχανισμός smoothing των gain τιμών, ώστε να αποφεύγονται ανεπιθύμητοι θόρυβοι κατά τη δυναμική αλλαγή παραμέτρων. Επιπλέον, υπολογίζεται ο συντελεστής εξομάλυνσης των RMS μετρήσεων και προετοιμάζεται το transportSource, το οποίο χρησιμοποιείται για την αναπαραγωγή αρχείων ήχου στη standalone λειτουργία. Η μέθοδος releaseResources() καλείται κατά τη διακοπή της επεξεργασίας και χρησιμοποιείται για την απελευθέρωση πόρων, όπως το transportSource.

Η βασική επεξεργασία υλοποιείται στη μέθοδο processBlock(), η οποία εκτελείται για κάθε block δειγμάτων. Στη λειτουργία plugin, το buffer εισόδου παρέχεται από το host ή το DAW, ενώ στη standalone λειτουργία μπορεί να τροφοδοτηθεί από το transportSource, όταν έχει φορτωθεί αρχείο ήχου. Η ροή επεξεργασίας έχει τα εξής στάδια:

1. **Έλεγχος κατάστασης export:** Ελέγχεται αν βρίσκεται σε εξέλιξη διαδικασία export ή αν το audio πρέπει να παγώσει προσωρινά λόγω του export UI. Σε αυτή την περίπτωση το buffer καθαρίζεται και η επεξεργασία σταματά για το συγκεκριμένο block.
2. **Τροφοδότηση buffer:** Στη λειτουργία plugin το buffer παρέχεται από το host, ενώ στη standalone λειτουργία μπορεί να τροφοδοτηθεί από το transportSource, όταν υπάρχει ενεργή αναπαραγωγή αρχείου.
3. **Εφαρμογή Input Pan:** Αν η παράμετρος InputPan είναι διαφορετική από την κεντρική θέση, εφαρμόζεται μεταβολή της σχετικής στάθμης του αριστερού και του δεξιού καναλιού.
4. **Αντιγραφή buffer για metering:** Δημιουργείται αντίγραφο του εισερχόμενου σήματος, το οποίο χρησιμοποιείται για τις ενδείξεις εισόδου και για την οπτική ανατροφοδότηση.
5. **Έλεγχος bypass:** Αν το bypass είναι ενεργό, παρακάμπτεται η δυναμική επεξεργασία, η gain reduction μηδενίζεται και ενημερώνονται οι μετρήσεις, τα clipping flags και οι buffers οπτικοποίησης.
6. **Εφαρμογή Input Gain:** Εφόσον το bypass δεν είναι ενεργό, εφαρμόζεται η ρύθμιση input gain στο κύριο buffer και στο buffer που χρησιμοποιείται για metering.

7. **Υπολογισμός Input Peak και RMS:** Υπολογίζονται οι τιμές εισόδου για τα meters και τις οπτικές ενδείξεις.
8. **Ενημέρωση circular buffers:** Τα δεδομένα εισόδου και εξόδου αποθηκεύονται σε buffers κυκλικής μνήμης, ώστε το GUI να έχει πρόσβαση σε πρόσφατο ιστορικό του σήματος για τις οπτικές ενδείξεις.
9. **Επιλογή τρόπου συμπίεσης:** Αν το σύστημα βρίσκεται σε single-band mode εφαρμόζεται ο βασικός compressor, ενώ σε multi-band mode καλείται ο multi-band compressor.
10. **Εφαρμογή Output Gain:** Ρυθμίζεται η τελική στάθμη εξόδου μετά τη συμπίεση.
11. **Υπολογισμός Output Peak, RMS και True Peak:** Υπολογίζονται οι τελικές μετρήσεις εξόδου και γίνεται έλεγχος κορυφών μέσω oversampling.
12. **Υπολογισμός Gain Reduction:** Υπολογίζεται η μείωση κέρδους και αποθηκεύεται σε atomic μεταβλητή για ασφαλή ανάγνωση από το GUI.
13. **Έλεγχος clipping:** Ενεργοποιούνται τα αντίστοιχα flags αν ανιχνευθεί υπέρβαση στάθμης.
14. **Ενημέρωση waveform FIFO:** Το FIFO buffer ενημερώνεται με δεδομένα για την απεικόνιση της κυματομορφής. Σε κανονική λειτουργία αποθηκεύονται το σήμα εισόδου και το σήμα εξόδου, ενώ σε κατάσταση bypass αποθηκεύεται μόνο το σήμα εισόδου ως οπτική αναφορά. Η χρήση του εξασφαλίζει ασφαλή μεταφορά δεδομένων από το audio thread προς το GUI, αποτρέποντας προβλήματα συγχρονισμού και ανεπιθύμητα glitches, clicks και dropouts.



Σχήμα 5.23: Διάγραμμα Ροής Processblock

Το αρχείο περιλαμβάνει επίσης λειτουργίες για τη standalone διαχείριση αρχείων ήχου. Οι μέθοδοι loadFile() και loadFileToPlayer() χρησιμοποιούνται για την ανάγνωση αρχείων, την αποθήκευση των δεδομένων στο fullFileBuffer και τη δημιουργία ενός AudioFormatReaderSource που συνδέεται με το transportSource. Ο fullFileBuffer χρησιμοποιείται για την αποθήκευση του ηχητικού αρχείου στη μνήμη, επιτρέποντας άμεση πρόσβαση για λειτουργίες ανάλυσης και οπτικοποίησης. Παράλληλα, δημιουργείται previewBuffer για την οπτική προεπισκόπηση του φορτωμένου αρχείου.

Οι μέθοδοι playFile(), playPauseFile(), stopFile(), stopAndRewindFile(), seekFileBySeconds(), clearLoadedFile() και isFilePlaying() υλοποιούν τον βασικό έλεγχο αναπαραγωγής και διαχείρισης του ηχητικού υλικού.

Η εξαγωγή του επεξεργασμένου σήματος υλοποιείται μέσω της μεθόδου exportFile(). Κατά τη διαδικασία αυτή, το φορτωμένο αρχείο αντιγράφεται σε προσωρινό buffer και επεξεργάζεται τμηματικά σε blocks σταθερού μεγέθους. Κάθε block περνά από τη μέθοδο processExportBlock(), η οποία εφαρμόζει την ίδια αλυσίδα επεξεργασίας με τη real-time λειτουργία, δηλαδή input panning, input gain, single-band ή multi-band compression και output gain. Στο τέλος της διαδικασίας, το επεξεργασμένο σήμα γράφεται σε αρχείο WAV μέσω AudioFormatWriter, ενώ η πρόοδος της διαδικασίας ενημερώνεται μέσω callback ώστε να μπορεί να απεικονιστεί στο γραφικό περιβάλλον.

Οι μέθοδοι getNumPrograms(), getCurrentProgram(), setCurrentProgram(), getProgramName() και changeProgramName() αποτελούν μέρος του βασικού interface της κλάσης juce::AudioProcessor και παρέχουν συμβατότητα με τα προγράμματα του host. Στην παρούσα υλοποίηση χρησιμοποιείται μόνο το προεπιλεγμένο πρόγραμμα του host, καθώς το σύστημα presets του compressor έχει υλοποιηθεί ανεξάρτητα μέσω του γραφικού περιβάλλοντος.

Η αποθήκευση και ανάκτηση της κατάστασης του plugin υλοποιείται μέσω των getStateInformation() και setStateInformation(), επιτρέποντας στο DAW να αποθηκεύει presets και να επαναφέρει πλήρως τις παραμέτρους. Επιπλέον, μέθοδοι του JUCE framework, όπως getName(), acceptsMidi(), producesMidi(), isMidiEffect(), getTailLengthSeconds() και isBusesLayoutSupported(), διασφαλίζουν τη συμβατότητα του plugin με διαφορετικά host περιβάλλοντα και καθορίζουν τη δυνατότητα υποστήριξης MIDI. Τέλος, η σύνδεση με το γραφικό περιβάλλον υλοποιείται μέσω των hasEditor() και createEditor(), που επιτρέπουν τη δημιουργία και προβολή της διεπαφής χρήστη.

Πίνακας 5.14: Μέθοδοι PluginProcessor.cpp

Κατηγορίες	Μέθοδοι	Λειτουργίες
Αρχικοποίηση	Constructor,Destructor, prepareToPlay(), releaseResources()	Διαχείριση πόρων και προετοιμασία
Επεξεργασία Ήχου	processBlock(),processExportBlock()	Επεξεργασία σήματος
Παράμετροι	createParameterLayout(),getStateInformation(), setStateInformation()	Διαχείριση παραμέτρων
Standalone	loadFile(), loadFileToPlayer(), playFile(), playPauseFile(), stopFile(), stopAndRewindFile(), seekFileBySeconds(), clearLoadedFile(), isFilePlaying()	Υποστήριξη λειτουργιών standalone αναπαραγωγής

Export	exportFile()	Εξαγωγή σήματος σε αρχείο ήχου
Επικοινωνία με Host	getName(), acceptsMidi(),producesMidi(),isMidiEffect(), getTailLengthSeconds(),isBusesLayoutSupported(), getNumPrograms(),getCurrentProgram(),setCurrentProgram(), getProgramName(),changeProgramName()	Διασύνδεση και συμβατότητα με το DAW
UI / Editor	hasEditor(),createEditor()	Σύνδεση με γραφικό περιβάλλον
Πρόσβαση Δεδομένων	getInputBufferCopy(), getOutputBufferCopy()	Παροχή δεδομένων για οπτικοποίηση σημάτων

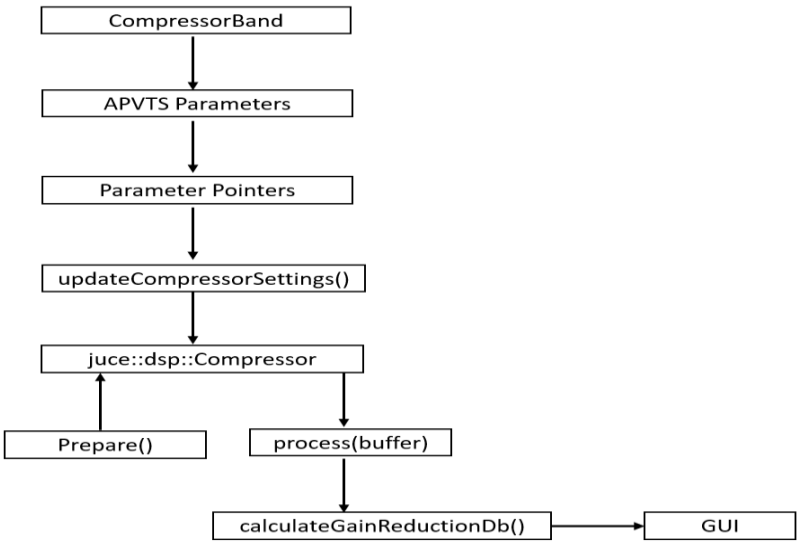
5.6.2 Ανάλυση PluginProcessor.h

Το αρχείο PluginProcessor.h καθορίζει τη βασική δομή και τα κύρια δομικά στοιχεία του συστήματος, παρέχοντας τη δήλωση για όλες τις λειτουργίες επεξεργασίας ήχου, την επικοινωνία με το GUI και τον host. Ο βασικός του ρόλος είναι να δηλώσει τις κλάσεις, τις δομές και τις μεθόδους που θα χρησιμοποιηθούν από τον compressor, χωρίς να υλοποιεί την πλήρη DSP λογική, η οποία περιλαμβάνεται στο αντίστοιχο αρχείο Processor.cpp. Με αυτόν τον τρόπο, το Processor.h λειτουργεί ως συνδετικός μηχανισμός ανάμεσα στα modules, επιτρέποντας στον processor να λαμβάνει τιμές παραμέτρων από το GUI και τον host, να τις εφαρμόζει στην DSP επεξεργασία και να επιστρέφει πληροφορίες πίσω στο GUI για την οπτική αναπαράσταση.

Μέρος του header αποτελεί η δομή CompressorBand, η οποία αναπαριστά μία αυτόνομη μονάδα συμπίεσης. Η δομή αυτή περιλαμβάνει δείκτες σε παραμέτρους όπως attack, release, threshold, ratio και bypassed, καθώς και σε παραμέτρους input και output gain. Οι παράμετροι αυτές διαχειρίζονται από το AudioProcessorValueTreeState (APVTS), το οποίο αποτελεί τον κεντρικό μηχανισμό διαχείρισης και αποθήκευσης των παραμέτρων του compressor. Μέσω της χρήσης δεικτών, η δομή επιτρέπει την άμεση πρόσβαση στις τρέχουσες τιμές των παραμέτρων χωρίς να απαιτείται αντιγραφή ή αποθήκευση ενδιάμεσων τιμών.

Η CompressorBand δηλώνει επίσης τις ακόλουθες μεθόδους:

- **prepare():** Προετοιμάζει το αντικείμενο juce::dsp::Compressor για επεξεργασία σύμφωνα με το ProcessSpec του host.
- **updateCompressorSettings():** Συγχρονίζει τις παραμέτρους του compressor με τις τιμές του APVTS.
- **process():** Εφαρμόζει τη συμπίεση σε ένα audio buffer, λαμβάνοντας υπόψη την κατάσταση bypass.
- **calculateGainReductionDb():** Υπολογίζει τη μείωση κέρδους σε dB με βάση τις τρέχουσες παραμέτρους.



Σχήμα 5.24: Διάγραμμα Ροής CompressorBand

Στο header δηλώνεται επίσης το enum class ProcessingMode, το οποίο καθορίζει τον τρόπο επεξεργασίας του σήματος. Οι διαθέσιμες καταστάσεις είναι single-band και multi-band, επιτρέποντας στον processor να εναλλάσσεται μεταξύ ενιαίας συμπίεσης του συνολικού σήματος και πολυζωνικής επεξεργασίας.

Η δομή MultiBandCompressor υλοποιεί τη βασική οργάνωση της multi-band επεξεργασίας. Περιλαμβάνει τρεις ανεξάρτητες μονάδες CompressorBand, οι οποίες αντιστοιχούν στη χαμηλή, μεσαία και υψηλή ζώνη συχνοτήτων. Επιπλέον, περιλαμβάνει τις παραμέτρους crossover, τα αντίστοιχα audio buffers για κάθε ζώνη και τέσσερα φίλτρα τύπου juce::dsp::LinkwitzRileyFilter<float>, τα οποία χρησιμοποιούνται για τον διαχωρισμό του σήματος.

Για κάθε crossover σημείο χρησιμοποιείται ένα ζεύγος φίλτρων low-pass και high-pass, ώστε το σήμα να διαχωρίζεται σε δύο συμπληρωματικές περιοχές γύρω από τη συχνότητα διαχωρισμού. Το low-pass φίλτρο αφήνει κυρίως το τμήμα του σήματος κάτω από τη συχνότητα crossover, ενώ το high-pass φίλτρο αφήνει κυρίως το τμήμα πάνω από αυτή.

Στην παρούσα υλοποίηση, το πρώτο crossover σημείο ορίζει το όριο μεταξύ χαμηλής και μεσαίας ζώνης, ενώ το δεύτερο ορίζει το όριο μεταξύ μεσαίας και υψηλής ζώνης. Έτσι, η χαμηλή ζώνη προκύπτει από το low-pass φίλτρο του πρώτου crossover και η υψηλή ζώνη από το high-pass φίλτρο του δεύτερου crossover, ενώ η μεσαία ζώνη δημιουργείται από τον συνδυασμό high-pass φίλτρου στο πρώτο crossover και low-pass φίλτρου στο δεύτερο crossover.

Η συγκεκριμένη κλάση φίλτρου του JUCE framework αντιστοιχεί σε Linkwitz-Riley φίλτρα τέταρτης τάξης με κλίση 24 dB/octave, επομένως η μετάβαση μεταξύ των ζωνών είναι σχετικά απότομη αλλά όχι ιδανικά κάθετη [54]. Η δομή υποστηρίζει επίσης λειτουργίες Mute, Bypass και Solo για κάθε ζώνη, καθώς και διαδικασία συνδυασμού των επιμέρους ζωνών για την παραγωγή του τελικού σήματος μετά την ανεξάρτητη επεξεργασία.

Στη συνέχεια, η κλάση PtyxiakhCompressorAudioProcessor οργανώνει τις λειτουργίες του plugin και δηλώνει όλα τα απαραίτητα μέλη για την επικοινωνία με τον host, τη διαχείριση παραμέτρων και την παροχή δεδομένων για το GUI. Ο constructor και ο destructor δηλώνονται για να καθορίσουν τη δημιουργία και καταστροφή των αντικειμένων, ενώ οι μέθοδοι prepareToPlay() και releaseResources() προσφέρουν τη δυνατότητα προετοιμασίας του DSP και απελευθέρωσης πόρων.

Η κύρια μέθοδος επεξεργασίας ήχου, processBlock(), δηλώνεται για να εκτελεί την κύρια DSP λογική. Παράλληλα, δηλώνονται οι μέθοδοι για τη διαχείριση παραμέτρων και κατάστασης, όπως createParameterLayout(), getStateInformation(), setStateInformation(), καθώς και οι getNumPrograms(), getCurrentProgram(), setCurrentProgram(), getProgramName() και changeProgramName(), που παρέχουν πρόσβαση σε προγράμματα χρήστη.

Η κλάση περιλαμβάνει επίσης δηλώσεις μεθόδων που καθορίζουν τη συμβατότητα με τον host και τις δυνατότητες MIDI, όπως getName(), acceptsMidi(), producesMidi(), isMidiEffect(), getTailLengthSeconds() και isBusesLayoutSupported(), καθώς και μεθόδων για το γραφικό περιβάλλον, όπως την hasEditor() και την createEditor(). Επιπλέον, δηλώνονται μέθοδοι για την πρόσβαση σε δεδομένα audio και την παρακολούθηση των επιπέδων ήχου, όπως getInputBufferCopy(), getOutputBufferCopy(), getAudioBufferCopy(), getLastAudioBuffer() και getPlaybackPosition(), καθώς και μέθοδοι για την ενημέρωση της κατάστασης του compressor, το metering και την ένδειξη clipping.

Στο ίδιο πλαίσιο, το header περιλαμβάνει και μεθόδους που σχετίζονται με τη standalone λειτουργία και τη διαχείριση αρχείων ήχου, όπως loadFile(), loadFileToPlayer(), playFile(), stopFile(), playPauseFile(), stopAndRewindFile(), seekFileBySeconds(), clearLoadedFile() και isFilePlaying(). Οι μέθοδοι αυτές καλύπτουν βασικές λειτουργίες όπως φόρτωση αρχείου, έλεγχο αναπαραγωγής, παύση, διακοπή, χρονική μετακίνηση και εκκαθάριση του φορτωμένου ηχητικού υλικού.

Για τη διαχείριση της αναπαραγωγής στη standalone λειτουργία χρησιμοποιείται η κλάση juce::AudioTransportSource, η οποία λειτουργεί ως ενδιάμεσος μηχανισμός μεταφοράς του ηχητικού σήματος προς την αλυσίδα επεξεργασίας και υλοποιεί τις βασικές λειτουργίες ελέγχου της αναπαραγωγής.

Η διαδικασία εξαγωγής δηλώνεται μέσω της exportFile() και της processExportBlock(). Η πρώτη αναλαμβάνει την εξαγωγή του τελικού επεξεργασμένου σήματος σε αρχείο ήχου, ενώ η δεύτερη εφαρμόζει την αλυσίδα επεξεργασίας σε blocks κατά τη διαδικασία offline export. Επιπλέον, η βοηθητική μέθοδος applyGain() χρησιμοποιείται για την εφαρμογή gain σε audio buffers.

Τέλος, το PluginProcessor.h ολοκληρώνεται με τις δηλώσεις των βασικών μελών της κλάσης, όπως audio buffers, μεταβλητές εξομάλυνσης, τιμές Peak και RMS, καθώς και atomic flags για την ανίχνευση clipping, διαμορφώνοντας ένα ολοκληρωμένο interface για την επεξεργασία σήματος, τη διεπαφή χρήστη και την επικοινωνία με το host.

Πίνακας 5.15: Μέθοδοι PluginProcessor.h

Κατηγορίες	Μέθοδοι	Λειτουργίες
Αρχικοποίηση	Constructor, Destructor, prepareToPlay(), releaseResources()	Αρχικοποίηση, αποδέσμευση πόρων και προετοιμασία DSP
Επεξεργασία Ήχου	processBlock(),processExportBlock(), applyGain()	Επεξεργασία και διαμόρφωση σήματος
Παράμετροι	createParameterLayout(), getStateInformation(), setStateInformation()	Διαχείριση και αποθήκευση παραμέτρων
Processing Mode	setProcessingMode(), getProcessingMode(), isMultiBandMode(), isSingleBandMode()	Εναλλαγή μεταξύ single-band και multi-band λειτουργίας
Επιλογή Ζώνης	setCurrentSelectedBand(), getCurrentSelectedBand()	Επιλογή ενεργής ζώνης στη multi-band λειτουργία
Επικοινωνία με Host	getName(), acceptsMidi(), producesMidi(), isMidiEffect(), getTailLengthSeconds(), isBusesLayoutSupported(), getNumPrograms(), getCurrentProgram(), setCurrentProgram(), getProgramName(), changeProgramName()	Συμβατότητα και επικοινωνία με DAW
UI / Editor	hasEditor(),createEditor()	Διαχείριση γραφικού περιβάλλοντος
Standalone / Playback	loadFile(), loadFileToPlayer(), playFile(), stopFile(), playPauseFile(), stopAndRewindFile(), clearLoadedFile(), seekFileBySeconds(), isFilePlaying(), play(), stop(), setPosition()	Διαχείριση αρχείων και standalone αναπαραγωγής
Πρόσβαση Δεδομένων	getInputBufferCopy(), getOutputBufferCopy(), getAudioBufferCopy(),getLastAudioBuffer(),	Παροχή δεδομένων για οπτικοποίηση
Analyzer / Waveform	getAnalyzerInputBuffer(),getAnalyzerOutputBuffer(), getWaveformInputBuffer(),getWaveformOutputBuffer()	Παροχή buffers για spectrum analyzer και waveform display
Bypass	isBypassed()	Έλεγχος κατάστασης λειτουργίας bypass
VU Meter Mode	setCurrentVUMode(), getCurrentVUMode()	Διαχείριση λειτουργίας VU meter

Μέτρηση Gain Reduction	getCurrentGainReductionDb(), getGainReductionForMeter(), getGainReductionDb()	Μέτρηση και απεικόνιση μείωσης κέρδους
Peak Μετρήσεις	getInputPeakL(), getInputPeakR(), getOutputPeakL(), getOutputPeakR(), getInputLevelDbL(), getInputLevelDbR(), getOutputLevelDbL(), getOutputLevelDbR()	Μέτρηση Peak επιπέδων
RMS Μετρήσεις	getInputRms(),getOutputRms(), getInputRmsDb(), getOutputRmsDb()	Υπολογισμός στάθμης RMS
Clipping	isClipping(),isInputClipping(), getInputClippingLevel(), getSmoothedClippingLevel(),getClippingIndicatorLevel()	Ανίχνευση και απεικόνιση clipping
Normalized Levels	getInputLevelNormalized(), getOutputLevelNormalized()	Κανονικοποιημένη απεικόνιση επιπέδου
Playback Visualization	getPlaybackPosition(), getCurrentPosition(), getTotalLength(), isPlayerPlaying(), isTransportPlaying()	Πληροφορία χρονικής θέσης και κατάστασης αναπαραγωγής

5.6.3 Ανάλυση PluginEditor.cpp

Το αρχείο PluginEditor.cpp διαχειρίζεται το γραφικό περιβάλλον του συστήματος και λειτουργεί ως γέφυρα επικοινωνίας μεταξύ του χρήστη και της μονάδας DSP, επιτρέποντας την αλληλεπίδραση με τις παραμέτρους επεξεργασίας. Η κλάση PtyxiakhCompressorAudioProcessorEditor, η οποία κληρονομεί από την juce::AudioProcessorEditor, αναλαμβάνει τη δημιουργία, την τοποθέτηση και τη διαχείριση των οπτικών στοιχείων της διεπαφής, όπως sliders, buttons, meters, waveform display και spectrum analyzer.

Ο constructor της κλάσης είναι υπεύθυνος για την αρχικοποίηση όλων των GUI components και τη σύνδεσή τους με τον APVTS του processor. Κατά την εκτέλεσή του, κάθε slider και button συνδέεται με την αντίστοιχη παράμετρο μέσω των attachments, εξασφαλίζοντας ότι η αλλαγή μιας παραμέτρου από τον χρήστη ενημερώνει άμεσα το DSP module και αντίστροφα. Επιπλέον, ο constructor αναλαμβάνει την τοποθέτηση των εικόνων για τα buttons, την οριστικοποίηση των χρωμάτων και των fonts για τα labels, καθώς και τη ρύθμιση των αρχικών toggle states για τα view mode buttons.

Στο ίδιο στάδιο, αρχικοποιούνται και τα στοιχεία της standalone λειτουργίας, όπως τα κουμπιά load, export, play, stop, clear, rewind και forward, καθώς και το time label για την προβολή της χρονικής θέσης αναπαραγωγής. Παράλληλα, δημιουργείται το ExportOverlayComponent, το οποίο εμφανίζεται κατά τη διαδικασία εξαγωγής και ενημερώνει τον χρήστη για την πρόοδο μέσω του γραφικού στοιχείου SegmentedBar και της ποσοστιαίας ένδειξης.

Πριν από την εκκίνηση της διαδικασίας εξαγωγής, ελέγχεται αν υπάρχει φορτωμένο αρχείο ήχου μέσω της hasLoadedFile(). Αν δεν έχει φορτωθεί αρχείο, εμφανίζεται προειδοποιητικό μήνυμα μέσω AlertWindow και η διαδικασία εξαγωγής διακόπτεται πριν δημιουργηθεί FileChooser ή κληθεί η exportFile(). Στο ίδιο στάδιο αρχικοποιούνται το PresetButton, το InputPan slider και τα στοιχεία της multi-band λειτουργίας, όπως τα κουμπιά επιλογής ζώνης low, mid και high, οι multi-band παράμετροι compression, τα κουμπιά bypass, mute και solo, καθώς και οι crossover sliders.

Παράλληλα, όλα τα components προστίθενται στην οθόνη μέσω της μεθόδου addAndMakeVisible(), ενώ ορίζεται το αρχικό μέγεθος του παραθύρου του compressor και ξεκινά ένας timer για περιοδική ενημέρωση των δυναμικών στοιχείων. Ο destructor της κλάσης είναι υπεύθυνος για την καταστροφή των αντικειμένων, διασφαλίζοντας την αυτόματη απελευθέρωση των πόρων που αυτά δεσμεύουν.

Η μέθοδος paint() είναι υπεύθυνη για την οπτική αναπαράσταση του φόντου της διεπαφής. Σε αυτήν εφαρμόζεται το μεταλλικό φόντο στο background εφόσον αυτό είναι διαθέσιμο. Σε περίπτωση που η εικόνα δεν είναι έγκυρη, χρησιμοποιείται ένα εναλλακτικό gradient φόντο για να διατηρείται η αισθητική συνοχή του interface. Επιπλέον, εφαρμόζεται ένα ήπιο overlay φωτισμού και σκίασης, το οποίο ενισχύει την οπτική αντίθεση της διεπαφής.

Η μέθοδος resized() καθορίζει δυναμικά τις θέσεις και διαστάσεις όλων των components ανάλογα με το μέγεθος του παραθύρου. Η μέθοδος αυτή τοποθετεί τα view mode toggle buttons με τρόπο που εξασφαλίζει ότι μόνο ένα mode είναι ενεργό κάθε στιγμή και οργανώνει τα radio groups των VU meter buttons ώστε να διατηρείται η σωστή επιλογή των modes. Επιπλέον, η resized() ορίζει τα bounds για τα οπτικά στοιχεία, προσφέροντας μια ομοιόμορφη και λειτουργική διάταξη.

Οι μέθοδοι setSelectedMultiBand() και syncMultiBandUIFromState() διαχειρίζονται την εμφάνιση και τον συγχρονισμό των στοιχείων της multi-band λειτουργίας. Η setSelectedMultiBand() καθορίζει ποια ζώνη συχνοτήτων είναι ενεργή και ανανεώνει τα attachments των sliders και των buttons ώστε να συνδέονται με τις παραμέτρους της αντίστοιχης ζώνης. Η syncMultiBandUIFromState() ελέγχει την κατάσταση της παραμέτρου MultiBandMode και εμφανίζει είτε τα single-band controls είτε τα multi-band controls, εξασφαλίζοντας σωστή εναλλαγή μεταξύ των δύο τρόπων λειτουργίας.

Η timerCallback() εκτελείται περιοδικά και αναλαμβάνει τη συγχρονισμένη ενημέρωση των δυναμικών στοιχείων του GUI με τα δεδομένα του processor. Μέσα σε αυτήν ελέγχεται η κατάσταση bypass, οι επιλογές view mode, ενημερώνονται τα meters, καθώς και τα displays. Παράλληλα, μεταφέρονται οι τιμές των παραμέτρων όπως threshold, ratio και gain reduction από τον processor προς τα οπτικά components. Με τον τρόπο αυτό, η μέθοδος εξασφαλίζει ότι ο χρήστης βλέπει σε πραγματικό χρόνο τις αλλαγές στο σήμα και στις παραμέτρους, χωρίς καθυστερήσεις.

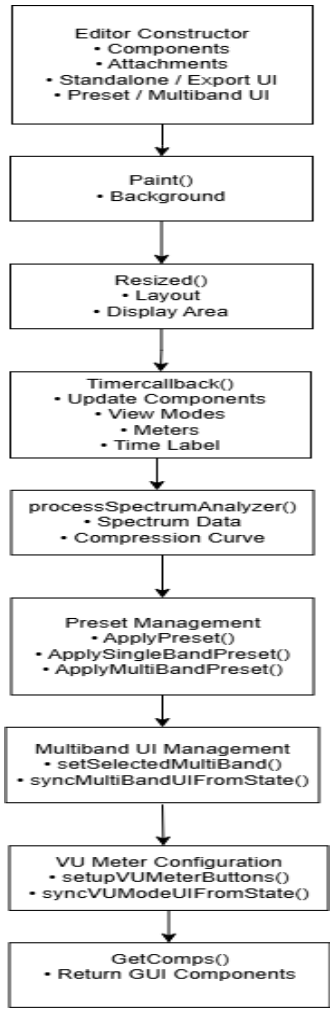
Η μέθοδος processSpectrumAnalyzer() είναι υπεύθυνη για την ενημέρωση του spectrum analyzer. Αρχικά λαμβάνει το τελευταίο διαθέσιμο audio buffer από τον processor μέσω της getLastAudioBuffer(). Αν το buffer περιέχει δείγματα, το περνά στο spectrum analyzer μέσω της pushBuffer(), ώστε να ενημερώνεται η απεικόνιση του φάσματος. Αν δεν υπάρχουν διαθέσιμα δείγματα, ο analyzer καθαρίζεται μέσω της clear(). Παράλληλα, η μέθοδος ενημερώνει την κατάσταση bypass του analyzer και διαχειρίζεται την εμφάνιση της καμπύλης συμπίεσης με βάση τις τρέχουσες τιμές threshold και ratio.

Η διαχείριση των presets γίνεται μέσω των μεθόδων applyPreset(), applySingleBandPreset() και applyMultiBandPreset(). Η applyPreset() εφαρμόζει το επιλεγμένο preset στις single-band και multi-band παραμέτρους, ενημερώνει το τρέχον preset ID στον processor και συγχρονίζει το γραφικό περιβάλλον. Η applySingleBandPreset() ρυθμίζει τις παραμέτρους input, output, threshold, ratio, attack και release. Αντίστοιχα, η applyMultiBandPreset() εφαρμόζει τιμές στις τρεις ζώνες συχνοτήτων χωρίς να μεταβάλλει τις καταστάσεις bypass, mute και solo.

Η μέθοδος setupVUMeterButtons() οργανώνει τα radio buttons του VU meter, ορίζοντας το κατάλληλο radio group, τη default επιλογή και τους click handlers που ενημερώνουν τον processor για την αλλαγή του mode. Τέλος, ο κώδικας της κλάσης ολοκληρώνεται με τη μέθοδο getComps(), η οποία επιστρέφει έναν δυναμικό πίνακα με pointers σε όλα τα GUI components, διευκολύνοντας την οργάνωση, τη διαχείριση και την ανανέωση του interface μέσω repaint.

Πίνακας 5.16: Μέθοδοι PluginEditor.cpp

Κατηγορίες	Μέθοδοι	Λειτουργίες
Αρχικοποίηση	Constructor, Destructor,	Δημιουργία, αρχικοποίηση και αποδέσμευση των στοιχείων
Γραφικά	paint()	Σχεδίαση και ενημέρωση background
Διάταξη	resized()	Τοποθέτηση και προσαρμογή των UI components
Ενημέρωση	timerCallback()	Περιοδική ενημέρωση και συγχρονισμός των οπτικών μονάδων
Επεξεργασία Οπτικοποιήσεων	processSpectrumAnalyzer()	Ενημέρωση spectrum analyzer, bypass κατάστασης και καμπύλης συμπίεσης
Presets	applyPreset(), applySingleBandPreset(), applyMultiBandPreset()	Εφαρμογή προκαθορισμένων ρυθμίσεων σε single-band και multi-band λειτουργία
Multiband UI	setSelectedMultiBand(), syncMultiBandUIFromState()	Επιλογή ζώνης, αλλαγή attachments και συγχρονισμός multi-band controls
Διαμόρφωση VU Meter	setupVUMeterButtons(), syncVUModeUIFromState()	Ρύθμιση radio group και click handlers για GR/IN/OUT επιλογή
Export UI	ExportOverlayComponent, SegmentedBar	Οπτική ένδειξη προόδου κατά την εξαγωγή αρχείου
Συλλογή Components	getComps()	Επιστροφή λίστας των UI components για μαζική διαχείριση



Σχήμα 5.25: Διάγραμμα Ροής ProcessorEditor.cpp

5.6.4 Ανάλυση PluginEditor.h

Το αρχείο PluginEditor.h καθορίζει τη δομή της κλάσης PtyxiakhCompressorAudioProcessorEditor, η οποία υλοποιεί το γραφικό περιβάλλον του συστήματος. Σε αντίθεση με το αντίστοιχο αρχείο υλοποίησης PluginEditor.cpp, στο header δηλώνονται τα public και private μέλη, τα οποία καθορίζουν τις μεθόδους και τα δεδομένα που συνθέτουν τη συνολική λειτουργία της διεπαφής.

Στο public τμήμα δηλώνονται οι βασικές μέθοδοι λειτουργίας του γραφικού περιβάλλοντος, όπως η paint(), η resized() και η timerCallback(), οι οποίες είναι υπεύθυνες για τη σχεδίαση των στοιχείων της διεπαφής, τη διαχείριση της διάταξής τους και την περιοδική ανανέωση των ενδείξεων.

Ιδιαίτερη σημασία παρουσιάζει το private τμήμα της κλάσης, στο οποίο δηλώνονται όλα τα μέλη που συγκροτούν τη δομή και τη λειτουργικότητα του γραφικού περιβάλλοντος. Σε αυτό περιλαμβάνεται αρχικά η αναφορά στον audioProcessor, η οποία επιτρέπει στο GUI να αντλεί δεδομένα από το DSP module και να συγχρονίζεται με την κατάσταση του compressor. Παράλληλα, δηλώνονται τα στοιχεία ελέγχου, όπως κουμπιά, περιστροφικοί sliders, labels και visual components, καθώς και οι αντίστοιχοι μηχανισμοί σύνδεσής τους με το AudioProcessorValueTreeState μέσω attachments, διασφαλίζοντας αμφίδρομη επικοινωνία μεταξύ διεπαφής και επεξεργασίας ήχου.

Επιπλέον, στο private τμήμα περιλαμβάνονται εσωτερικές καταστάσεις που καθορίζουν τον τρόπο λειτουργίας της διεπαφής, όπως το SelectedBand για την ενεργή ζώνη στη multi-band λειτουργία και το DisplayMode για την επιλογή μεταξύ VU meter, spectrum analyzer και waveform display. Στο ίδιο τμήμα δηλώνονται επίσης μέθοδοι για την εφαρμογή presets, τον συγχρονισμό της multi-band διεπαφής, την ενημέρωση του VU meter και την επεξεργασία των δεδομένων που προβάλλονται στον spectrum analyzer.

Το header περιλαμβάνει ακόμη στοιχεία που σχετίζονται με τη standalone λειτουργία και τη διαδικασία export. Μεταξύ αυτών περιλαμβάνονται κουμπιά φόρτωσης, αναπαραγωγής, παύσης, μετακίνησης και διαγραφής αρχείων ήχου, το time label, η ατομική μεταβλητή exportProgressAtomic, καθώς και το ExportOverlayComponent.

Τέλος, πριν από τον ορισμό της κλάσης PtyxiakhCompressorAudioProcessorEditor, το header περιλαμβάνει τις δηλώσεις των βασικών κλάσεων και δομών του γραφικού συστήματος, οι οποίες έχουν αναλυθεί σε προηγούμενη ενότητα.

Πίνακας 5.17: Κλάσεις και Δομές PluginEditor.h

Κλάση/Δομή	Λειτουργίες
CustomRotarySlider	Υλοποίηση περιστροφικού knob
LogicProStyleButton	Προσαρμοσμένο κουμπί με 3D σχεδίαση
PNGToggleButton	Υλοποίηση toggle button για εναλλαγή display mode
BypassButton	Υλοποίηση toggle button για Bypass λειτουργία
PanKnob	Περιστροφικό knob για τη ρύθμιση της στερεοφωνικής θέσης
CurveSwitch	Διακόπτης εμφάνισης καμπύλης για spectrum analyzer
RangeSlider	Slider για επιλογή εύρους τιμών
SpectrumAnalyzer	Υλοποίηση αναλυτή φάσματος
SimpleWaveformDisplay	Υλοποίηση απεικόνισης κυματομορφών
ProgrammaticVUMeter	Υλοποίηση μετρητή στάθμης με βελόνα
LedVuMeter	Υλοποίηση μετρητή στάθμης LED
MultiBandEditorComponent	Component για τη διαχείριση multi-band παραμέτρων
FrequencySliderLookAndFeel	Προσαρμοσμένο LookAndFeel για τα sliders συχνοτήτων crossover

5.7 Επίλογος

Η παρουσίαση των τεχνικών λεπτομερειών υλοποίησης του συστήματος ανέδειξε την πολυεπίπεδη αρχιτεκτονική και τη συστηματική προσέγγιση που ακολουθήθηκε για τη δημιουργία ενός λειτουργικού και σταθερού εργαλείου επεξεργασίας ήχου. Μέσα από την αναλυτική περιγραφή παρουσιάστηκε ο τρόπος με τον οποίο τα επιμέρους υποσυστήματα συνεργάζονται για να προσφέρουν ακρίβεια, απόδοση και άμεση οπτική ανατροφοδότηση στον χρήστη.

Η αρχιτεκτονική του συστήματος, όπως αποτυπώνεται στα τέσσερα βασικά αρχεία, αποτελεί ένα συνεκτικό και δομημένο σύνολο, όπου κάθε τμήμα έχει σαφή ρόλο και η επικοινωνία μεταξύ DSP, GUI και host πραγματοποιείται με ασφάλεια και αποδοτικότητα.

Συνολικά, η προσέγγιση αυτή καταδεικνύει ότι ένα καλά σχεδιασμένο audio plugin μπορεί να συνδυάζει τεχνική ακρίβεια, απόδοση σε πραγματικό χρόνο και εργονομικό GUI, παρέχοντας ένα αξιόπιστο εργαλείο για επαγγελματική χρήση.

Κεφάλαιο 6ο: Αξιολόγηση Συστήματος

6.1 Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζεται η διαδικασία αξιολόγησης του αναπτυγμένου συστήματος. Η αξιολόγηση χωρίζεται σε δύο βασικούς άξονες, την τεχνική αξιολόγηση και την αξιολόγηση από χρήστες. Στόχος της διαδικασίας είναι η διερεύνηση της ακρίβειας, της σταθερότητας και της λειτουργικής αξιοπιστίας του συστήματος σε πραγματικές συνθήκες επεξεργασίας ήχου, καθώς και η αποτίμηση της εμπειρίας χρήσης από τελικούς χρήστες. Στις επόμενες ενότητες παρουσιάζεται αναλυτικά η μεθοδολογία που ακολουθήθηκε, καθώς και τα αποτελέσματα των δύο αυτών μορφών αξιολόγησης.

6.2 Τεχνική Αξιολόγηση

Η τεχνική αξιολόγηση του αναπτυγμένου συστήματος πραγματοποιήθηκε με συστηματικές δοκιμές σε ποικίλα ηχητικά σήματα, προκειμένου να εκτιμηθεί η συμπεριφορά του σε διαφορετικά φάσματα συχνοτήτων και δυναμικές περιοχές. Τα σήματα που επιλέχθηκαν περιλάμβαναν drums, basslines, lead synths σε μορφή sine wave, καθώς και φωνητικές ηχογραφήσεις, τόσο μονοφωνικές όσο και στερεοφωνικές. Αυτή η επιλογή επέτρεψε την εκτίμηση της ακρίβειας των μετρήσεων σε σήματα με διαφορετική πολυπλοκότητα φάσματος, σύνθεση αρμονικών και δυναμικό εύρος.

Κατά τη διάρκεια των δοκιμών εφαρμόστηκαν ισοσταθμηστές και φίλτρα όπως high-pass και low-pass, με σκοπό να μεταβληθεί το φασματικό περιεχόμενο των σημάτων και να εξεταστεί η απόκριση του plugin σε διαφορετικές περιοχές συχνοτήτων. Οι δοκιμές πραγματοποιήθηκαν σε περιβάλλον ψηφιακού σταθμού επεξεργασίας ήχου και συγκεκριμένα στο FL Studio όπου το plugin τοποθετήθηκε σε bus channels, επιτρέποντας την παρακολούθηση των μετρήσεων.

6.3 Σύγκριση και Έλεγχος της Λειτουργίας

Για την αξιολόγηση της ακρίβειας του plugin χρησιμοποιήθηκαν τα καθιερωμένα επαγγελματικά plugins Voxengo SPAN και MV Meter 2. Η επιλογή αυτών των εργαλείων βασίστηκε στην ευρεία αποδοχή τους στην επαγγελματική παραγωγή και στην αξιοπιστία τους ως εργαλεία ανάλυσης φάσματος και μέτρησης επιπέδων.

Για τη σύγκριση του φασματικού περιεχομένου χρησιμοποιήθηκε το SPAN σε ρύθμιση Default Mode, το οποίο εφαρμόζει ανάλυση FFT με εργοστασιακές παραμέτρους και τυπική εξομάλυνση. Αυτή η προεπιλογή εφαρμόζει κλίση 4.5dB ανά οκτάβα στην απεικόνιση του φάσματος [55]. Επιπλέον τροποποιήθηκε η ρύθμιση Average Time στο Spectrum Mode Editor του SPAN, μειώνοντάς την στα 250 ms ώστε η χρονική απόκριση του SPAN να ταιριάζει με αυτή του compressor. Το SPAN χρησιμοποιήθηκε επίσης για τη σύγκριση των LED meters, τόσο στο επίπεδο εισόδου όσο και στο επίπεδο εξόδου, αξιολογώντας τις Peak ενδείξεις του συστήματος με αυτές του LED meter του Span.

Σχήμα 6.1: Σύγκριση με Span

Αντίστοιχα, για τη σύγκριση των επιπέδων στάθμης χρησιμοποιήθηκε το MV Meter 2 σε ρύθμιση RMS Standard Mode, το οποίο υπολογίζει τη μέση τετραγωνική τιμή του σήματος με τυπικό χρονικό παράθυρο ολοκλήρωσης. Στη σύγκριση, το MV Meter 2 ρυθμίστηκε ώστε να εμφανίζει μόνο ένα meter διότι η προεπιλογή εμφανίζει δύο meters και η βελόνα κόκκινου clipping απενεργοποιήθηκε, ώστε να μην επηρεάζει την αξιολόγηση.

Σχήμα 6.2: Σύγκριση με mvMeter2

Τα ίδια ηχητικά σήματα εφαρμόστηκαν και στα τρία εργαλεία, διασφαλίζοντας συγκρίσιμες συνθήκες. Κατά τη διαδικασία των δοκιμών, οι μετρήσεις του plugin συγκρίθηκαν ταυτόχρονα με αυτές των επαγγελματικών εργαλείων και έδειξαν ότι τα αποτελέσματα είναι παρόμοια, με ελάχιστες αποκλίσεις.

Πιο συγκεκριμένα, κατά τη σύγκριση με το mvMeter 2 η μεταβολή της βελόνας του compressor ακολουθεί αντίστοιχη δυναμική απόκριση ως προς την αύξηση και την επαναφορά της στάθμης με αυτή του συγκρίσιμου μετρητή.

Αντίστοιχα, κατά τη σύγκριση με το Voxengo SPAN, η φασματική απεικόνιση του compressor παρουσιάζει παρόμοια κατανομή ενέργειας και δυναμική συμπεριφορά. Το αναπτυγμένο σύστημα ανάλυσης καταγράφει με συνέπεια τις φασματικές κορυφές και ακολουθεί τις μεταβολές του σήματος σε σχέση με το εργαλείο αναφοράς.

Επιπλέον, πραγματοποιήθηκε ξεχωριστός έλεγχος της standalone λειτουργίας, χρησιμοποιώντας τα ίδια αρχεία ήχου με αυτά που εφαρμόστηκαν στην έκδοση του plugin μέσα στο DAW. Σκοπός του ελέγχου αυτού ήταν να διαπιστωθεί αν η standalone εφαρμογή παρουσιάζει αντίστοιχη συμπεριφορά ως προς την επεξεργασία, τις μετρήσεις στάθμης και τη φασματική απεικόνιση. Τα αποτελέσματα έδειξαν ότι η απόκριση της standalone έκδοσης ήταν παρόμοια με αυτή του plugin, γεγονός που επιβεβαιώνει τη σταθερότητα λειτουργίας του συστήματος ανεξάρτητα από τον τρόπο εκτέλεσης.

Συνολικά, οι συγκρίσεις δείχνουν ότι το σύστημα λειτουργεί σωστά και αξιόπιστα, παρέχοντας ακριβείς ενδείξεις RMS, Peak και φασματικής ανάλυσης ανεξάρτητα από το είδος των ηχητικών σημάτων. Η παρατηρούμενη ομοιότητα επιβεβαιώνει ότι η υλοποίηση των αλγορίθμων DSP του συστήματος παρουσιάζει σταθερή και ακριβή συμπεριφορά.

6.4 Αξιολόγηση Χρηστών

Πέρα από τις τεχνικές δοκιμές και τις συγκρίσεις με επαγγελματικά εργαλεία ανάλυσης, πραγματοποιήθηκε και αξιολόγηση του συστήματος από χρήστες με εμπειρία στην επεξεργασία και παραγωγή ήχου. Στόχος της διαδικασίας ήταν η συνολική αποτίμηση του συστήματος σε πραγματικές συνθήκες λειτουργίας.

Για τη συλλογή των αξιολογήσεων δημιουργήθηκε ερωτηματολόγιο μέσω της πλατφόρμας Google Forms, το οποίο διανεμήθηκε σε 8 χρήστες με εμπειρία στη χρήση DAWs και συστημάτων επεξεργασίας ήχου. Οι συμμετέχοντες αξιολόγησαν διαφορετικές πτυχές του συστήματος μέσω πενταβάθμιας κλίμακας Likert [56], με τις απαντήσεις να διαμορφώνονται ως εξής:

- 1: Καθόλου
- 2: Λίγο
- 3: Μέτρια
- 4: Αρκετά
- 5: Πολύ

Εξαίρεση αποτέλεσε η τελευταία ερώτηση (E13), η οποία αφορούσε την πρόθεση χρήσης του συστήματος σε πραγματικές συνθήκες παραγωγής και διαμορφώθηκε ως ερώτηση επιλογής με απαντήσεις «Ναι», «Ίσως» και «Όχι».

Οι ερωτήσεις του ερωτηματολογίου επικεντρώθηκαν κυρίως στην αξιολόγηση βασικών χαρακτηριστικών του συστήματος, όπως η λειτουργικότητα, η δυναμική συμπίεση, η ποιότητα του γραφικού περιβάλλοντος και η εμπειρία χρήσης. Επιπλέον, αξιολογήθηκαν επιμέρους λειτουργίες του plugin, όπως οι οπτικές ενδείξεις, η multi-band επεξεργασία, η standalone λειτουργία και η χρήση των presets. Ο Πίνακας 6.1 παρουσιάζει αναλυτικά τις ερωτήσεις του ερωτηματολογίου και τις αντίστοιχες περιγραφές τους.

Πίνακας 6.1: Πίνακας Ερωτήσεων

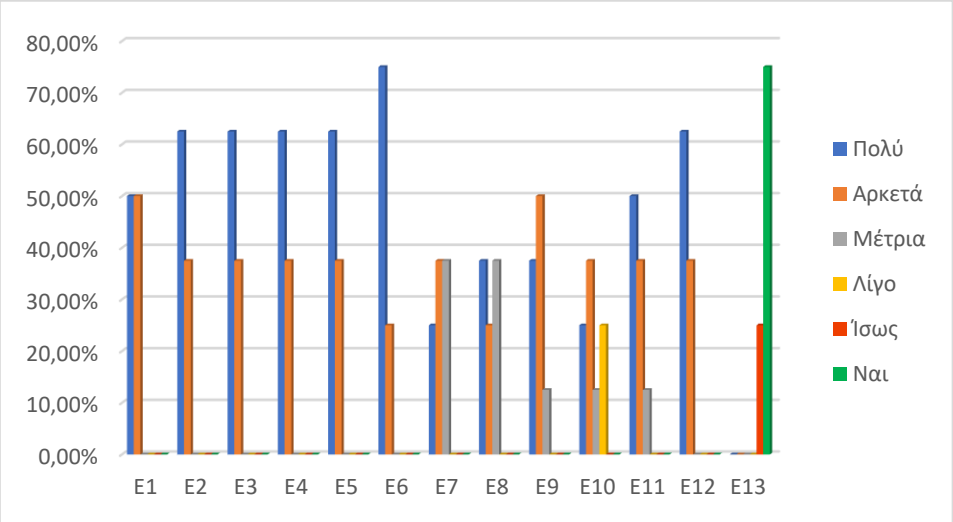
Ερώτηση	Περιγραφή
E1	Πόσο εύκολη θεωρείτε την πλοήγηση και τη χρήση του συστήματος;
E2	Πόσο ικανοποιητικός είναι ο σχεδιασμός του γραφικού περιβάλλοντος χρήστη;
E3	Πόσο εύκολα μπορείτε να ελέγξετε τη δυναμική του ήχου;
E4	Πόσο αποτελεσματική είναι η δυναμική συμπίεση;
E5	Πόσο εύχρηστη θεωρείτε τη ρύθμιση των παραμέτρων;
E6	Πόσο άμεση είναι η απόκριση του συστήματος;
E7	Πόσο χρήσιμες θεωρείτε τις οπτικές ενδείξεις του συστήματος;
E8	Πόσο χρήσιμη θεωρείτε τη χρήση των presets;
E9	Πόσο χρήσιμη θεωρείτε τη multi-band λειτουργία;
E10	Πόσο χρήσιμη θεωρείτε τη standalone λειτουργία;
E11	Πόσο σταθερή και αξιόπιστη θεωρείτε τη λειτουργία του συστήματος;
E12	Πόσο ολοκληρωμένο θεωρείτε το σύστημα ως εργαλείο επεξεργασίας ήχου;
E13	Θα χρησιμοποιούσατε το σύστημα σε πραγματικές συνθήκες παραγωγής ή επεξεργασίας ήχου;

Τα αποτελέσματα της αξιολόγησης από τους συμμετέχοντες έδειξαν συνολικά θετική ανταπόκριση ως προς τη λειτουργικότητα, την ευχρηστία και τη συνολική εμπειρία χρήσης του συστήματος. Οι περισσότερες απαντήσεις συγκεντρώθηκαν στις ανώτερες επιλογές της κλίμακας Likert, υποδεικνύοντας θετικό επίπεδο αποδοχής.

Ιδιαίτερα θετικά αποτελέσματα καταγράφηκαν στις ερωτήσεις που αφορούν τη βασική λειτουργικότητα του συστήματος (E1–E6), όπως η πλοήγηση, ο έλεγχος της δυναμικής του ήχου και η απόκριση του συστήματος, όπου επικράτησαν υψηλές αξιολογήσεις. Αντίστοιχα, παρατηρήθηκε υψηλό επίπεδο ικανοποίησης ως προς τη δυναμική συμπίεση.

Οι επιμέρους λειτουργίες, όπως οι οπτικές ενδείξεις, η multi-band επεξεργασία, η standalone λειτουργία και η χρήση presets (E7–E10), εμφάνισαν πιο διαφοροποιημένες αξιολογήσεις, οι οποίες κυμάνθηκαν κυρίως μεταξύ των απαντήσεων «Μέτρια» και «Πολύ», υποδεικνύοντας επιμέρους σημεία προς βελτίωση.

Τέλος, οι ερωτήσεις που σχετίζονται με τη συνολική αποτίμηση του συστήματος και την πρόθεση χρήσης σε πραγματικές συνθήκες (E11–E13) παρουσίασαν ιδιαίτερα θετικά αποτελέσματα, με την πλειοψηφία των συμμετεχόντων να το αξιολογεί ως αξιόπιστο και κατάλληλο για χρήση σε πραγματικά περιβάλλοντα επεξεργασίας ήχου.



Σχήμα 6.3: Διάγραμμα Απαντήσεων

6.5 Επίλογος

Στο κεφάλαιο αυτό αναλύθηκε η διαδικασία δοκιμών και αξιολόγησης του αναπτυγμένου plugin σε πραγματικές συνθήκες επεξεργασίας ήχου. Μέσω της εφαρμογής διαφορετικών ηχητικών σημάτων και φασματικών μεταβολών εξετάστηκε η συμπεριφορά του συστήματος ως προς την ακρίβεια των μετρήσεων και τη σταθερότητα της λειτουργίας του.

Η σύγκριση των αποτελεσμάτων με καθιερωμένα επαγγελματικά εργαλεία ανάλυσης έδειξε ότι οι μετρήσεις του plugin παρουσιάζουν παρόμοια συμπεριφορά με αυτές των αντίστοιχων εργαλείων. Παράλληλα, η αξιολόγηση μέσω ερωτηματολογίου ανέδειξε συνολικά θετική ανταπόκριση από τους χρήστες.

Συνολικά, τα αποτελέσματα των δοκιμών επιβεβαιώνουν ότι η υλοποίηση των αλγορίθμων είναι ορθή και σταθερή, καθιστώντας το σύστημα κατάλληλο για χρήση σε περιβάλλοντα παραγωγής και επεξεργασίας ήχο.

Κεφάλαιο 7ο: Συμπεράσματα & Μελλοντικές Βελτιώσεις

7.1 Μελλοντικές Βελτιώσεις

Παρά την πληρότητα της υπάρχουσας υλοποίησης, το πεδίο ανάπτυξης των audio plugins προσφέρει πλήθος δυνατοτήτων επέκτασης. Με βάση τα αποτελέσματα αξιολόγησης που παρουσιάστηκαν στο προηγούμενο κεφάλαιο, οι ακόλουθες προτάσεις αποσκοπούν στην περαιτέρω ενίσχυση της λειτουργικότητας και της χρηστικότητας του sound compressor.

Μια σημαντική μελλοντική προσθήκη θα μπορούσε να είναι η δυνατότητα επιλογής διαφορετικών ηχητικών χαρακτήρων, μέσω προσομοίωσης της συμπεριφοράς γνωστών αναλογικών compressors. Με την προσθήκη κατάλληλων παραμέτρων που ρυθμίζουν τα στάδια κορεσμού, όπως drive και saturation amount, ο χρήστης θα μπορούσε να επιλέγει την ποσότητα κορεσμού που θα προσέδιδε στο σήμα ώστε να επιτύχει πιο ζεστό ή επιθετικό χαρακτήρα. Μια τέτοια προσέγγιση θα παρείχε ευελιξία και ηχητική ποικιλομορφία.

Επιπλέον, μια ακόμη ενδιαφέρουσα μελλοντική βελτίωση θα μπορούσε να είναι η ενσωμάτωση sidechain λειτουργίας, η οποία θα επέτρεπε στον χρήστη να ελέγχει τη συμπίεση του σήματος βασιζόμενος σε ένα ξεχωριστό audio input. Με αυτόν τον τρόπο, θα ήταν δυνατή η δημιουργία δυναμικών και μουσικά ευαίσθητων εφέ, όπως ducking μεταξύ drums και μπάσου ή επιλεκτική συμπίεση συγκεκριμένων στοιχείων του mix, προσφέροντας μεγαλύτερο έλεγχο και λειτουργικές δυνατότητες στην παραγωγή.

Τέλος, το σύστημα θα μπορούσε να επεκταθεί μέσω της προσθήκης ρυθμίσεων για την ταχύτητα απόκρισης των οπτικών ενδείξεων. Ο χρήστης θα μπορούσε να ρυθμίζει την ταχύτητα κίνησης της βελόνας του VU meter, την ταχύτητα πτώσης των LED meters ή τον βαθμό απόκρισης του spectrum analyzer και της απεικόνισης της κυματομορφής. Με αυτόν τον τρόπο, η οπτική απεικόνιση θα μπορούσε να προσαρμόζεται στις ανάγκες του χρήστη, προσφέροντας είτε πιο άμεση και γρήγορη ένδειξη είτε πιο ομαλή και σταθερή παρακολούθηση του σήματος.

7.2 Συμπεράσματα

Συνολικά, η παρούσα διπλωματική εργασία πέτυχε τον στόχο της, αποτελώντας μια ολοκληρωμένη μελέτη και υλοποίηση ενός σύγχρονου συστήματος δυναμικής επεξεργασίας ήχου. Η εργασία γεφυρώνει επιτυχώς τη θεωρητική προσέγγιση της ψηφιακής επεξεργασίας σήματος με την πρακτική της εφαρμογή σε πραγματικά περιβάλλοντα ανάπτυξης, προσφέροντας ένα λειτουργικό, επεκτάσιμο και δημιουργικά ευέλικτο σύστημα.

Το τελικό προϊόν δεν αποτελεί απλώς μια τεχνική απόδειξη της δυνατότητας εφαρμογής θεμελιωδών αρχών DSP, αλλά και ένα χρηστικό εργαλείο με άμεση εφαρμογή σε πραγματικές συνθήκες παραγωγής ήχου. Οι προτεινόμενες μελλοντικές επεκτάσεις αναδεικνύουν τη δυνατότητα περαιτέρω εξέλιξης σε ένα ακόμη πιο ισχυρό και επαγγελματικό σύστημα, συμβάλλοντας στην κατανόηση και την πρακτική εφαρμογή της ψηφιακής επεξεργασίας στον ευρύτερο τομέα της μουσικής τεχνολογίας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Κεφάλαιο 1:

[1] K. Vezina, “Sound Amplitude,” EBSCO, 2019

Available: https://www.ebsco.com/research-starters/physics/sound-amplitude.

[2] J.-M. Réveillac, “Musical Sound Effects: Analog and Digital Sound Processing,” ISTE Press, 2018, p. 30.

[3] C. Anet, “Why is dynamic range so important?”, QSC, Available:

https://blogs.qsc.com/live-sound/why-is-dynamic-range-so-important/.

[4] S. H. Hawley, B. Colburn, and S. I. Mimilakis, “SignalTrain: Profiling Audio Compressors with

Deep Neural Networks,” 2019.

Κεφάλαιο 2:

[5] Kadis J. “Dynamic range processing and digital effects,” Stanford University, 2006–2018.

[6] Jeffs R., Holden S. & D. Bohn, “Dynamics processors — Technology & applications,” Rane Corporation, 2005, p. 14. Available: https://www.ranecommercial.com/legacy/note155.html.

[7] M. Droney, H. Massey, “Compression Applications,” TC Electronic, Sep. 2001.

[8] Vintage King, “Gates SA-39B Tube Compressor/Limiter (Vintage),” Available: https://vintageking.com/gates-sa39b-tube-compressor-limiter-vintage.

[9] A. Fox, “What is a variable-mu (tube) compressor & how does it work?”, Fox Music Production, 2024, Available: https://foxmusicproduction.com/variable-mu-tube-compressors/.

[10] Equipboard, “Fairchild 670 Compressor/Limiter,” Available:

https://equipboard.com/items/fairchild-670-compressor-limiter.

[11] Universal Audio, “Teletronix LA-2A Classic Leveling Amplifier,” Available: https://help.uaudio.com/hc/en-us/articles/206356233-Teletronix-LA-2A-Classic-Leveling-Amplifier.

[12] Nail The Mix Staff, “How the 1176 Compressor Crushes A Metal Mix,” 2025, Available: https://www.nailthemix.com/1176-compressor.

[13] SongMixMaster, “The Urei 1176 Compressor – A legend in audio engineering,” Available:

https://songmixmaster.com/the-urei-1176-compressor-a-legend-in-audio-engineering.

[14] I. Anderson, “Birth of a classic: The dbx 160 compressor,” Mixonline, 2020, Available:

https://www.mixonline.com/technology/birth-of-a-classic-the-dbx-160-compressor.

[15] Sen M. Kuo, Bob H. Lee & Wenshun Tian, “Real-time digital signal processing: Implementations, Applications,” 2nd, 2006.

[16] D. Gibson, R. Polfreman, “An Architecture For Creating Hosting Plug-Ins For Use in Digital Audio Workstations,” in Proc. Int. Computer Music Conf. (ICMC), England, 2011.

[17] D. Koszewski and B. Kostek, “Low-level audio descriptors-based analysis of music mixes from different Digital Audio Workstations – case study,” 2018 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, Poland, 2018, pp. 213-216.

[18] FabFilter, “FabFilter releases FabFilter Pro-C 2 compressor plug-in,” 2015, Available:

https://www.fabfilter.com/press/1440572940/fabfilter-releases-fabfilter-pro-c-2-compressor-plug-in.

[19] Waves Audio, “CLA-76 Compressor/Limiter,” Available:

https://www.waves.com/plugins/cla-76-compressor-limiter.

[20] Emerson College, “iZotope Neutron,” Available:

https://support.emerson.edu/hc/en-us/articles/21708926540571-iZotope-Neutron.

[21] C. Hollomey and P. Hackl-Lehner, “A Brief History of the Digital Audio Workstation,” IEEE History of Electrotechnology Conference (HISTELCON), Bonn, Germany, 2025, pp. 1-6.

Κεφάλαιο 3:

[22] S. W. Smith, “The Scientist and Engineer's Guide to Digital Signal Processing,” 2nd ed. San Diego, California, USA, California Technical Publishing, 1999.

[23] GeeksforGeeks, “Difference Between Analog and Digital signal,” Available:

https://www.geeksforgeeks.org/physics/difference-between-analog-and-digital-signal.

[24] Hollyland, “What is sample rate in audio?”, Available:

https://www.hollyland.com/blog/tips/what-is-sample-rate-in-audio.

[25] MathWorks. “Nyquist Theorem,” Available:

https://www.mathworks.com/discovery/nyquist-theorem.html.

[26] Z. Raz, “The design of an audio signal processor system based on DSP,” IEEE International Conference on Consumer Electronics Digest of Technical Papers., Rosemount, IL, USA, 1988, pp. 226-227.

[27] John G.Proakis, Dimitris G.Manolakis, “Digital Signal Processing: Principals, Algorithms and Aplications,” 3nd Edition, USA, Prentice Hall, 1996, pp. 22.

[28] Tutorialspoint, “Digital communication – Quantization,” Available:

https://www.tutorialspoint.com/digital_communication/digital_communication_quantization.htm.

[29] Monolithic Power Systems, “Fundamental Concepts: Sampling, Quantization, and Encoding,” Available: https://www.monolithicpower.com/en/learning/mpscholar/analog-to-digital-converters/introduction-to-adcs/fundamental-concepts.

[30] W. Kester, “Taking the Mystery out of the Infamous Formula, ‘SNR = 6.02N + 1.76dB,’ and

Why You Should Care,” Analog Devices, 2009.

[31] R. G. Lyons, “Understanding Digital Signal Processing,” 3rd ed., USA, Pearson Education, 2011.

[32] B. Martin and V. Juliet, “Detection of pest infestation by preprocessing sound using vector quantization,” 2010 2nd International Conference on Signal Processing Systems, Dalian, China, 2010,

pp. 219-223.

[33] M. Bosi, R. E. Goldberg, “Introduction to Digital Audio Coding and Standards,” 1st ed., New York, USA, Springer, 2003.

[34] Francisco J. Casajús-Quirós “Digital Signal Processors for real-time audio processing,” in Proc. Int. Conf. on Digital Audio Effects (DAFx), Spain, 1998.

Κεφάλαιο 4:

[35] S. Valentan and F. Wotawa, “On the Automation of Audio Plugin Testing,” IEEE 21st

International Conference on Software Quality, Reliability and Security (QRS), Hainan, China, 2021,

pp. 270-278.

[36] B. Ahmed, R. Mahajan, A. Jain, V. Attri, P. Arora and M. Kumar, “Maximizing Development

Efficiency: A Comprehensive Guide to Frameworks and Libraries, ” 2025 3rd International Conference on Disruptive Technologies (ICDT), Greater Noida, India, 2025, pp. 1325-1330.

[37] G. Maudoux and K. Mens, “Correct, Efficient, and Tailored: The Future of Build Systems,” in IEEE Software, vol. 35, no. 2, pp. 32-37, March/April 2018.

[38] V.Goudard & Remy Muller ,“Real-time audio plugin architectures,” IRCAM Centre Pompidou, 2003.

[39] A. H. Moore, R. R. Vos, P. A. Naylor and M. Brookes, “Processing Pipelines for Efficient, Physically-Accurate Simulation of Microphone Array Signals in Dynamic Sound Scenes,” ICASSP IEEE International Conference on Acoustics, Speech and Signal Processing , Toronto, ON, Canada, 2021, pp. 965-969.

[40] H. S. Ioniţă, V. Popa and B. Moroşanu, "Multiband Dynamics Compressor with Automatic Parameter Learning," 2024 15th International Conference on Communications (COMM), Bucharest, Romania, 2024, pp. 1-4.

[41] P. N. Kulkarni, P. C. Pandey and D. S. Jangamashetti, "Multi-band frequency compression for sensorineural hearing impairment," 2009 16th International Conference on Digital Signal Processing, Santorini, Greece, 2009, pp. 1-6.

[42] P. N. Kulkarni, P. C. Pandey, and D. S. Jangamashetti, “Multi-band frequency compression for improving speech perception by listeners with moderate sensorineural hearing loss,” Speech Communication, vol. 54, no. 3, pp. 341–350, 2012.

[43] Heo Hoon, Lee Mingu, Lee Seokjin and Sung Koeng-Mo, Development of Multiband Dynamic Range Compressor Regarding Noise Characteristics, Seoul National University, Seoul, Korea, Paper 8454, 2011, Available: https://aes.org/publications/elibrary-page/?id=15921.

[44] A. K. ERGÜT and M. U. SIRMA, “Parameter Based Abstraction For a Modular Software,” Turkish National Software Engineering Symposium (UYMS), Istanbul, Turkey, 2020, pp. 1-4.

[45] H. Schmid, “Audio Program Level, the VU Meter, and the Peak-Program Meter,” in IEEE

Transactions on Broadcasting, vol. BC-23, no. 1, pp. 22-26, March 1977.

Κεφάλαιο 5:

[46] Giannoulis, Dimitrios & Massberg, Michael & Reiss, Joshua, “Digital Dynamic Range

Compressor Design—A Tutorial and Analysis,” Queen Mary University of London, London, UK, 2012.

[47] Aasis Beats, “A beginner’s guide to compression in music production,” 2023, Available: https://aasisbeats.com/a-beginners-guide-to-compression-in-music-production/.

[48] M. Rayden, “What is RMS in audio world?”, Major Mixing, 2023, Available:

https://majormixing.com/what-is-rms-in-audio-world/.

[49] Learnius, “Decibel Full Scale (dBFS),” Available:

https://learnius.com/slp/4+Speech+Signal+Representations/1+Time-Domain/4+Time-Domain+Features/deciBel+Full+Scale+(dBFS).

[50] ScienceDirect, “Hamming window,” Available:

https://www.sciencedirect.com/topics/computer-science/hamming-window.

[51] Brilliant, “Discrete Fourier Transform,” Available:

https://brilliant.org/wiki/discrete-fourier-transform/.

[52] M. Brenndoerfer, “Normalization: Complete Guide to Feature Scaling with Min-Max Implementation,” 2025, Available:https://mbrenndoerfer.com/writing/normalization-feature-scaling-min-max-machine-learning-guide.

[53] G200kg Music & Software, “Knob Gallery,” Available:

https://www.g200kg.com/en/webknobman/gallery.php.

[54] JUCE, “juce::dsp::LinkwitzRileyFilter< SampleType > Class Template Reference,” JUCE Documentation. Available: https://docs.juce.com/master/classjuce_1_1dsp_1_1LinkwitzRileyFilter.html.

Κεφάλαιο 6:

[55] A. Vaneev, “Voxengo SPAN User Guide,” Voxengo, Available: https://www.voxengo.com/files/userguides/VoxengoSPAN_en.pdf/getbyname/Voxengo%20SPAN%20User%20Guide%20en.pdf.

[56] R. Likert, “A technique for the measurement of attitudes,” Archives of Psychology, no. 140, 1932.