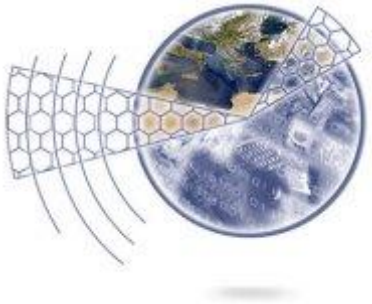
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής



Πτυχιακή Εργασία

Διαδραστική Ψηφιακή Τέχνη
Θεωρία και Πράξη

Ξυλάκης Εμμανουήλ ΑΜ:2241

Επιβλέπων καθηγητής: Τριανταφυλλίδης Γεώργιος

Επιτροπή Αξιολόγησης:

Ημερομηνία παρουσίασης:

Περιεχόμενα

Πίνακας εικόνων

1. Εισαγωγή

1.1 Σύνοψη

Η πτυχιακή έχει σαν σκοπό την έρευνα πάνω στη διαδικασία μετατροπής ηχητικών δεδομένων σε εικόνα και τη δημιουργία μιας πολυμεσικής διαδραστικής εφαρμογής πραγματικού χρόνου που θα μετατρέπει δεδομένα ήχου σε εικόνα. Η εφαρμογή αυτή θα λειτούργει όπως ένα media player(πλατφόρμα αναπαραγωγής μουσικών κομματιών) η οποία θα συμπεριλαμβάνει και το αντίστοιχο οπτικό υλικό που συναντάμε σχεδόν σε όλες τις εφαρμογές τέτοιου τύπου(όπως winamp, windows media player, i-tunes κ.α.).Πιο συγκεκριμένα μέσω μίας κάμερας με αισθητήρα βάθους θα ανιχνεύεται ό χρήστης από τη στιγμή που θα εισέρχεται στο οπτικό της πεδίο. Με την ολοκλήρωση της ανίχνευσης ο χρήστης θα μπορεί να χειρίζεται διαφορετικές λειτουργίες της εφαρμογής μέσω κάποιων καταστάσεων (modes) με τις κινήσεις των χεριών του (όπως αυξομείωση της έντασης ή αλλαγή του κομματιού).Πέρα από αυτές τις βασικές λειτουργίες, η διεπαφή της εφαρμογής θα περιέχει, επιπρόσθετες πληροφορίες για το κομμάτι που θα λαμβάνονται από τα ID3 tags των αρχείων, αναφορά στην επιλεγμένη κατάσταση(mode), στην οποία θα βρίσκεται ο χρήστης καθώς και buttons(κουμπιά), που μέσω αυτών θα εναλλάσσονται οι καταστάσεις αυτές. Η εφαρμογή ολοκληρώνεται με τη προσθήκη φίλτρων, που θα επιτρέπουν στο χρήστη να επεμβαίνει ηχητικά στο κάθε κομμάτι.

Πέρα από το πρακτικό κομμάτι η πτυχιακή έχει σαν σκοπό , να αναλύσει τις διαφορετικές τεχνικές που χρησιμοποιούνται ευρέως πάνω στο τομέα οπτικοποιήσης του ήχου(ή πιο σωστά μετατροπής ήχου σε εικόνα).Ποιοι τομείς επωφελούνται από τη τεχνική αυτή, τι πληροφορίες αντλούν και πως τις εκμεταλλεύονται. Στη συνέχεια θα αναφερθούν οι πιο πρόσφατες εφαρμογές στο χώρο των φορητών συσκευών(tablets και smartphones) και πως τις χειρίζεται ο χρήστης μέσω της οθόνης αφής.

Τέλος για την δημιουργία της εφαρμογής θα χρησιμοποιηθεί ο αισθητήρας βάθους(depth sensor) kinect της Microsoft ,για τη διαδραστικότητα του χρήστη με την εφαρμογή και η γλώσσα προγραμματισμού processing η οποία επιτρέπει σε χρήστες να προγραμματίσουν εικόνα, animation και ήχο, παρέχοντας μία πληθώρα επιλογών μέσω των βιβλιοθηκών ήχου που μας παρέχει.

Ένα βίντεο με την ολοκληρωμένη εφαρμογή βρίσκεται εδώ https://vimeo.com/84955671

Abstract

The thesis aims to research the process of converting audio data into visual data and to create a multimedia interactive application. This application will function as a media player which will include the corresponding visual material found in almost all applications of this type (like winamp, windows media player, i-tunes , etc.) . More specifically, through a depth sensor, it will detect the user from the moment he/she enters the field of view. With the completion of the user tracking, it will be able to handle different functions of the application through some situations (modes) with gestures (such as modulating the intensity or change the track). Besides these basic functions , the interface of the application will contains additional information of the track to be taken from the ID3 tags of the files , a reference to the selected mode, in which the user is located and buttons ,which will rotate these situations . The application is completed with the addition of a digital bandpass filter , which allow the user to intervene at every sound track.

Beyond the practical part the thesis aims to analyze the different techniques that are widely used on field of visualization (or more correctly converting audio to image). Which sectors benefit from this technique, what information are derived and how do we use the to our advantage. Also what are the latest applications in the field of portable devices (tablets and smart phones) and how the user operates them via the touch screen.

Finally to create the application Microsoft's kinect sensor is used, on the interactivity with the app and the programming language processing enables users to program image, animation and sound, providing a plethora of options through its audio libraries.

A demo video of the platform can be found at: https://vimeo.com/84955671

1.2 Εισαγωγή

Πληροφορική (αγγλ. information science,IS) είναι η επιστήμη που ασχολείται με την συλλογή, αποθήκευση, επεξεργασία και μετάδοση πληροφοριών. Γενικά μελετά τα φαινόμενα που συνδέονται με την πληροφορία ,ερευνά τα θεωρητικά θεμέλια των εννοιών της και του υπολογισμού, καθώς και τις τεχνολογικές εφαρμογές τους σε αυτοματοποιημένα υπολογιστικά συστήματα, από τη σκοπιά της σχεδίασης, της ανάπτυξης, της υλοποίησης, της διερεύνησης, της ανάλυσης και της προδιαγραφής τους.

Πολυμέσα στον χώρο της τεχνολογίας πληροφορίας (information field) σημαίνει πολλαπλοί μεσολαβητές μεταξύ της πηγής και του παραλήπτη της πληροφορίας ή πολλαπλά μέσα μέσω των οποίων η πληροφορία αποθηκεύεται, μεταδίδεται, παρουσιάζεται ή γίνεται αντιληπτή.Ψηφιακά πολυμέσα είναι ο τομέας που ασχολείται με την ελεγχόμενη από υπολογιστή ολοκλήρωση κειμένου, γραφικών, ακίνητης και κινούμενης εικόνας, animation, ήχου, και οποιουδήποτε άλλου μέσου ψηφιακής αναπαράστασης, αποθήκευσης, μετάδοσης και επεξεργασίας της πληροφορίας.

Ψηφιακή τέχνη είναι ένας πολυδιάστατος όρος για μια σειρά καλλιτεχνικών έργων και εφαρμογών που χρησιμοποιούν την ψηφιακή τεχνολογία ως μέσο για τη δημιουργία και τη προβολή της. Πιο συγκεκριμένα είναι η δημιουργία καλλιτεχνικών έργων που υλοποιούνται με τη βοήθεια ηλεκτρονικών μέσων όπως ο Η/Υ. Όταν λέμε "ψηφιακή " εννοούμε τους αριθμούς(ψηφία) που χρησιμοποιεί ο υπολογιστής για να μεταφράσει και να καταγράψει την πληροφορία. Αυτή η σειρά ψηφίων δεν αποτελεί προβληματισμό για τον δημιουργό, του δίνει όμως τη δυνατότητα καταγραφής και αποθήκευσης των έργων του που έχει ως αποτέλεσμα την επανεπεξεργασία τους με στόχο την βελτιστοποίηση αυτών φτάνοντας έτσι στη κατάρριψη ενός και μόνο αντιγράφου. (Miller)

Η διαδικασία της **χαρτογράφησης**(mapping) δημιουργεί γραφικές αναπαραστάσεις των πληροφοριών με τη χρήση χωρικών σχέσεων μέσα στο γραφικό για να εκπροσωπήσει σχέσεις μεταξύ δεδομένων. Η κοινή και πρωτότυπη πρακτική της χαρτογράφησης είναι η κλίμακα σχεδίασης των γεωγραφικών χαρακτηριστικών, δηλαδή η χαρτογραφία. Η σύγχρονη έννοια της οπτικοποίησης δεδομένων, περιλαμβάνει μεταφορικές επεκτάσεις του γεωγραφικού χάρτη και γραμματισμού σε άλλα είδη δεδομένων, καθώς και καινοτόμους τρόπους οπτικοποίησης δεδομένων που δεν συνδέονται σαφώς με τη γεωγραφική αρχέτυπο. Στη λαϊκή καθομιλουμένη, χαρτογράφηση μπορεί να σημαίνει απλώς την οργάνωση και συστηματοποίηση πληροφοριών.

1.3 Περίληψη πτυχιακής

Σκοπός της πτυχιακής είναι η έρευνα πάνω στην οπτικοποίηση ηχητικών σημάτων (μετατροπή ήχου σε εικόνα).Αρχικά θα εστιάσουμε στα πρώτα βήματα της ψηφιακής ανάλυσης ενός σήματος ,στα στοιχεία που εξάγουμε ,τι σημασία έχουν αυτά , που χρησιμεύουν και ποιους τομείς εξυπηρετεί η εικονική αναπαράσταση του ήχου. Στη συνέχεια θα περάσουμε σε μία ανάλυση βασικών τεχνικών και αλγορίθμων πάνω στις διαδεδομένες εφαρμογές στο τομέα αυτό. Πιο συγκεκριμένα πως ο Η/Υ αντιλαμβάνεται τις διαφορετικές διακυμάνσεις στον ήχο και πως μπορούμε να τον χρησιμοποιήσουμε σαν εργαλείο για να εκμεταλλευτούμε τις πληροφορίες και τα δεδομένα που εξάγουμε. Επίσης θα δούμε πως με τα εργαλεία αυτά που μας παρέχει ο Η/Υ μπορούμε να παράγουμε εικονικά στοιχεία που εξυπηρετούν αισθητικούς σκοπούς(ψηφιακή τέχνη)δηλαδή όχι για να εξάγουμε κάποια πληροφορία για τον ήχο μας και θα αναφέρουμε εφαρμόζονται τέτοιου είδους αναπαραστάσεις. Τελειώνοντας θα αναφέρουμε τους δύο πιο βασικούς τύπους αλγορίθμων πάνω στην εύρεση ρυθμού(beat) σε ένα μουσικό κομμάτι θα εξηγήσουμε το σκεπτικό γύρω από το καθένα και πως θα μπορέσουμε να τους εφαρμόσουμε στο πρακτικό μέρος της πτυχιακής.

1.4 Κίνητρο για την διεξαγωγή της εργασίας

Κίνητρο για την επιλογή του συγκεκριμένου θέματος ήταν το προσωπικό μου ενδιαφέρον όσον αφορά τις τεχνικές δημιουργίας οπτικών εφέ και εικονικών στοιχείων. Πέρα από αυτό ήθελα το θέμα να περιέχει σε ένα βαθμό την ανάλυση του ήχου μιας και αποτελεί μία από τις κύριες ενασχολήσεις μου. Όλα τα παραπάνω σε συνδυασμό με την ανάγκη για εξοικείωση με τις πιο πρόσφατες τεχνολογίες στο τομέα αποτέλεσαν τα κύρια κριτήρια για την υλοποίηση της συγκεκριμένης πτυχιακής.

1.5 Σκοπός και στόχοι εργασίας

Η πτυχιακή αυτή έχει σαν σκοπό να ερευνήσει τις μεθόδους και τεχνικές στο τομέα μετατροπής ήχου σε εικόνα δηλαδή τι δεδομένα εξάγουμε μέσω της ψηφιακής ανάλυσης του ήχου, τι πληροφορίες παίρνουμε από αυτά και ποιους τομείς εξυπηρετούν οι εφαρμογές που ανήκουν σ' αυτό το τομέα. . Επίσης σύμφωνα με την έρευνα και τις τεχνικές που θα μελετηθούν θα δημιουργηθεί μια εφαρμογή πάνω σε αυτές. Η τελική εφαρμογή θα αναπαράγει μουσικά κομμάτια(δηλαδή όπως ένα media player) και παραλλήλως θα εξάγει οπτικό υλικό σύμφωνα με τις πληροφορίες που θα αντλούμε από αυτά.

1.6 Δομή Εργασίας

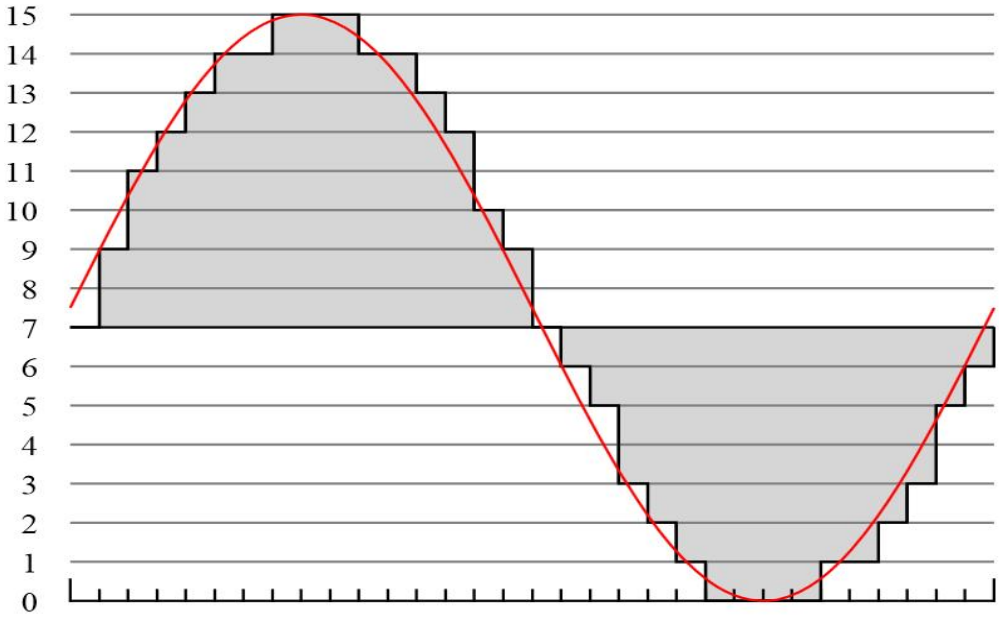
Αρχικά θα ξεκινήσουμε με τα βασικά στοιχεία που απαρτίζουν τον ψηφιακό ήχο για να περάσουμε στις βασικές εικονικές αναπαραστάσεις του και τους τομείς στους οποίους εφαρμόζονται αυτές. Στη συνέχεια θα περάσουμε σε ένα αρκετά σημαντικό κομμάτι της πτυχιακής που είναι η ανίχνευση ρυθμού σε πραγματικό χρόνο και θα την αναλύσουμε και θα ακολουθήσει μετά μία περιγραφή των εργαλείων που θα χρησιμοποιήσουμε για την υλοποίηση της εφαρμογής. Για την εφαρμογή μας θα δοκιμάσουμε πρώτα διαφορετικά περιβάλλοντα πάνω στην ψηφιακή ανάλυση του ήχου και στο πως θα μπορέσουμε μέσω αυτών να εξάγουμε εικονικά στοιχεία με βάση τον ήχο. Θα μελετηθούν ήδη υπάρχοντες τεχνικές πάνω στην ανάλυση του ήχου και η θεωρία γύρω από αυτό το τομέα και θα δοκιμαστούν διαφορετικά μουσικά είδη από προγράμματα τύπου DAW(Digital Audio Workstations) και πως θα μπορέσουμε εμείς να δημιουργήσουμε μία εφαρμογή που θα εξάγει οπτικό υλικό σε σχέση με το κάθε είδος. Θα δοκιμαστούν γλώσσες προγραμματισμού που προτείνονται για την εξαγωγή εικονικών στοιχείων και θα επιλέξουμε αυτή που είναι πιο κατάλληλη για την εφαρμογή μας. Για την ολοκλήρωση της θα συνδέσουμε την εφαρμογή μας με το kinect της microsoft σαν ένα εναλλακτικό τρόπο διαδραστικότητας με το χρήστη.

2. Παρουσίαση επιμέρους στοιχείων

2.1 Ήχος και ψηφιακός ήχος

Όπως γνωρίζουμε ο ήχος είναι η αίσθηση που προκαλείται λόγω της διέγερσης των αισθητηρίων οργάνων της ακοής από μεταβολές πίεσης του ατμοσφαιρικού αέρα. Αυτές οι μεταβολές διαδίδονται με τη μορφή ηχητικών κυμάτων και μπορούν να γίνουν αντιληπτές όταν ανήκουν σε εύρος συχνότητας 20Hz έως 20kHz. (Stearns)

Για τη μετάδοση των κυμάτων είναι απαραίτητη η ύπαρξη κάποιου υλικού μέσου (στερεό, υγρό, αέριο) γιατί ο ήχος δεν μεταδίδεται στο κενό.



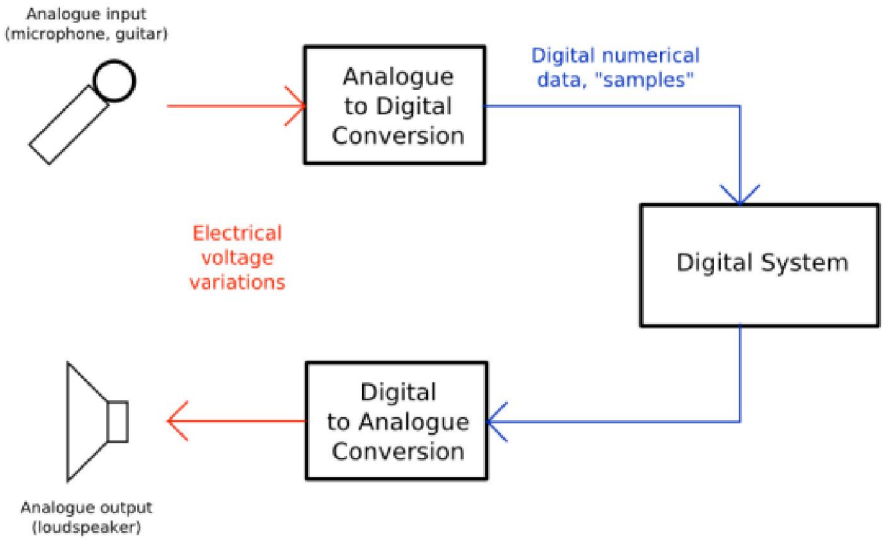
1. *ψηφιακή και αναλογική αναπαράσταση ηχητόνου*

Όταν μιλάμε για **ψηφιακό ήχο** εννοούμε την τεχνολογία που χρησιμοποιούμε για τη καταγραφή αποθήκευση και αναπαραγωγή ενός ηχητικού σήματος το οποίο βρίσκεται σε ψηφιακή και όχι αναλογική μορφή. Η μετατροπή του σήματος από αναλογικό σε ψηφιακό, γίνεται μέσω δειγματοληψίας και κβάντισης με τη βοήθεια ενός **μετατροπέα αναλογικού σήματος σε ψηφιακό (ADC Analog to Digital Converter),** ο οποίος μετασχηματίζει το αναλογικό σήμα σε μια ακολουθία από αριθμούς. Συχνά όμως το ζητούμενο σήμα εξόδου είναι επίσης αναλογικό παρόλο που η επεξεργασία έχει ψηφιακό χαρακτήρα, επομένως χρειάζεται και ένας **μετατροπέας ψηφιακού σήματος σε αναλογικό (DAC)**. Αν και αυτή η διαδικασία μπορεί να είναι πιο πολύπλοκη σε σχέση με την αναλογική επεξεργασία και έχει διακριτό πεδίο τιμών, η χρήση υπολογιστικής ισχύος στην ψηφιακή επεξεργασία σήματος έχει πολλά πλεονεκτήματα σε σχέση με την αναλογική επεξεργασία σήματος σε πολλές εφαρμογές, όπως ο εντοπισμός και η διόρθωση λαθών στις επικοινωνίες και η συμπίεση δεδομένων. [1]

Ο ήχος διέρχεται μέσω ενός αναλογικού προς ψηφιακό μετατροπέα (ADC) και με μία παλμική διαμόρφωση χρησιμοποιείται συνήθως για να κωδικοποιηθεί ως ψηφιακό σήμα. Αντίστροφη διαδικασία γίνεται πάλι για να μετατρέψουμε το ψηφιακό μας σήμα σε ακουστικό ήχο.

[1] http://en.wikipedia.org/wiki/Digital_audio

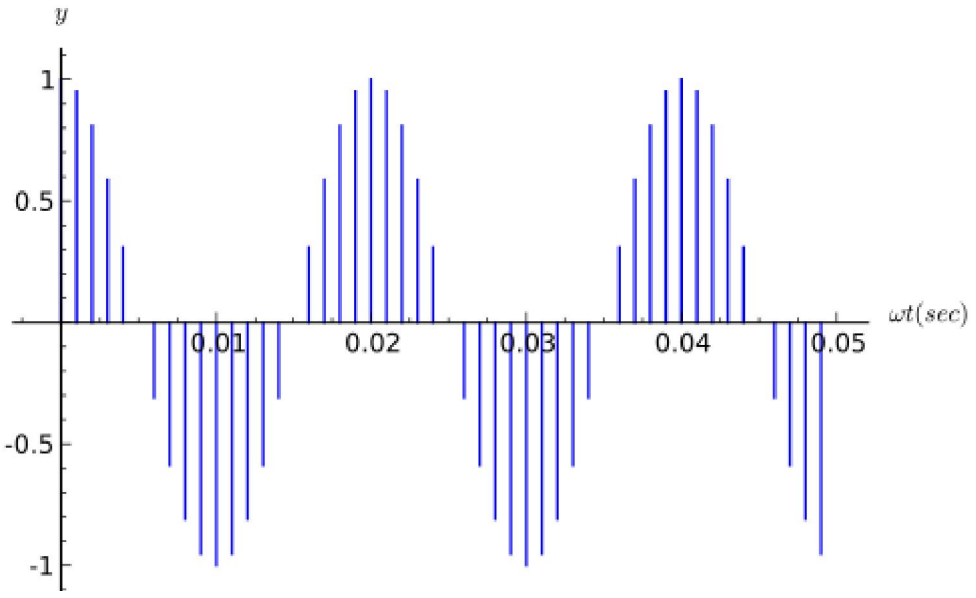
Όταν ο ήχος είναι σε ψηφιακή μορφή μπορούμε να τον αποθηκεύσουμε, επεξεργαστούμε και να τον μεταδώσουμε. Η ψηφιοποίηση του ήχου είναι αρκετά σημαντική επιτρέπει να ανακτήσουμε αρκετές πληροφορίες για το ηχητικό μας σήμα.



2. Μετατροπή αναλογικό-ψηφιακό και αντίστροφα

2.2 Ψηφιακή Επεξεργασία Σήματος

Η ψηφιακή επεξεργασία σήματος (ΨΕΣ, αγγλ.: Digital signal processing ή DSP) ασχολείται με την **αναπαράσταση σημάτων διακριτού χρόνου** ως ακολουθιών αριθμών ή συμβόλων, καθώς και με την επεξεργασία των σημάτων αυτών. Η ψηφιακή και η αναλογική επεξεργασία σήματος συναποτελούν το διεπιστημονικό γνωστικό πεδίο των εφαρμοσμένων μαθηματικών, γνωστό ως **επεξεργασία σήματος**. Κάποιες εφαρμογές της ψηφιακής επεξεργασίας σήματος είναι η επεξεργασία ήχου, η αναγνώριση φωνής, η επεξεργασία σημάτων από σόναρ, ραντάρ και συστοιχίες αισθητήρων, η εκτίμηση φάσματος, η στατιστική επεξεργασία σήματος, η ψηφιακή επεξεργασία εικόνας, η επεξεργασία σήματος στις τηλεπικοινωνίες και η επεξεργασία σεισμικών δεδομένων. Ο σκοπός της ψηφιακής επεξεργασίας σήματος συνήθως είναι **η μέτρηση, το φιλτράρισμα ή η συμπίεση συνεχόμενων αναλογικών φυσικών σημάτων**.



3. Δειγματοληψία σήματος

2.2.1 Δυαδική Πληροφορία

Όπως εξηγήσαμε και παραπάνω ο Η/Υ δεν μπορεί να "καταλάβει" την αναλογική πληροφορία ή όποια δεν είναι διακριτή αλλά συνεχόμενη. Έτσι μέσω **της μετατροπής του σήματος σε ψηφιακό** μπορεί να γίνει από αυτόν κατανοητή. Πάμε να αναφέρουμε λίγο πιο αναλυτικά κάποιες έννοιες στη ψηφιοποίηση του ήχου καθώς αποτελούν βασικά στοιχεία της ανάλυσης ενός ψηφιακού σήματος. (Stearns)

Δυαδική πληροφορία είναι τα δεδομένα μας τα οποία έχουν μετατραπεί σε μία σειρά από 0 και 1 έτσι ώστε να είναι κατανοητά από τον Η/Υ .Όταν μιλάμε για δυαδική πληροφορία θα πρέπει να καταλάβουμε ότι όσα περισσότερα bits έχουμε τόσο περισσότερες καταστάσεις μας επιτρέπονται να έχουμε. Για παράδειγμα όταν έχω 2 bits ,έχω 4 δυνατές καταστάσεις 00, 01, 10 και 11.Το πόσες καταστάσεις παίρνω σε σχέση με τα bit μου υπολογίζεται όταν υψώσω το 2 στον αριθμό τον bits που έχω στη διάθεση μου(μήκος λέξης,**word length** ή **bit depth**).[2]

$2^1 = 2$	$2^9 = 512$	$2^{17} = 131.072$
$2^2 = 4$	$2^{10} = 1.024$	$2^{18} = 262.144$
$2^3 = 8$	$2^{11} = 2.048$	$2^{19} = 524.288$
$2^4 = 16$	$2^{12} = 4.096$	$2^{20} = 1.048.576$
$2^5 = 32$	$2^{13} = 8.192$	$2^{21} = 2.097.152$
$2^6 = 64$	$2^{14} = 16.384$	$2^{22} = 4.194.304$
$2^7 = 128$	$2^{15} = 32.768$	$2^{23} = 8.388.608$
$2^8 = 256$	$2^{16} = 65.536$	$2^{24} = 16.777.216$

4. Καταστάσεις σε σχέση με τα διαθέσιμα bits

2.2.2 Βασικές έννοιες ψηφιακού ήχου

Όταν τώρα μιλάμε για ήχο σε ψηφιακή μορφή έχουμε κάποια πρότυπα όσον αφορά το μήκος τη λέξης για παράδειγμα στα CD χρησιμοποιούμε 16 bit λέξη. Το μήκος αυτό καθορίζει τη ποιότητα στο μουσικό μας κομμάτι και σε γενικές γραμμές είναι αρκετά ικανοποιητικό στα 16 bit αλλά κατά την ηχογράφηση υπάρχει μία τάση να χρησιμοποιούνται αρκετά μεγαλύτερες ρυθμίσεις όπως 24 bit. Έτσι λοιπόν για να καταλαβαίνουμε πως ακριβώς επηρεάζουν το σήμα μας αυτές του είδους οι ρυθμίσεις το μήκος της λέξης συσχετίζεται με το **πλάτος του σήματος** ή αλλιώς το εύρος της περιοχής μας από κορυφή σε κορυφή(dynamic range).

Άλλη μία σημαντική παράμετρος κατά την ψηφιοποίηση του ηχητικού μας σήματος είναι **ο ρυθμός δειγματοληψίας.** Όπως αναφέραμε παραπάνω το μέγεθος της λέξης αφορά το κατά πόσο αναλυτική θα είναι η κάθε μέτρηση αλλά το πόσο συχνά θα γίνονται αυτές οι μετρήσεις σε ένα δευτερόλεπτο ορίζονται από το ρυθμό δειγματοληψίας. Άρα καταλαβαίνουμε η παράμετρος αυτή συσχετίζεται με τη **συχνότητα** του σήματος μας. Όσον αφορά τώρα το ρυθμό δειγματοληψίας έχουμε και εδώ κάποια πρότυπα δηλαδή στα CD χρησιμοποιούμε 44100 δείγματα το δευτερόλεπτο και ο λόγος που επιλέγουμε το συγκεκριμένο μέγεθος είναι για την αναπαράσταση της λεγόμενης **συχνότητας Nyquist** η οποία είναι το μισό του ρυθμού δειγματοληψίας δηλαδή 22050 Hz. Απ ότι

[2] The Art of Digital Audio 3rd Edition, John Watkinson

καταλαβαίνουμε ο συγκεκριμένος ρυθμός είναι υπεραρκετός άμα λάβουμε υπόψη μας ότι το ανθρώπινο αυτί λαμβάνει μέχρι το πολύ 20000 Hz.

Παρόλα' αυτά και στο συγκεκριμένο μέγεθος επιλέγουμε τιμές μεγαλύτερες από 44100 δείγματα/δευτερόλεπτο τις 48000 δείγματα το δευτερόλεπτο .

Από τα παραπάνω καταλαβαίνουμε ότι για να μπορέσουμε να κάνουμε τέτοιες μετρήσεις μέσω υπολογιστή σε τέτοιο ρυθμό(44100 δείγματα) μεγέθους 16 bit κάθε δευτερόλεπτο απαιτεί τέλειο συγχρονισμό και ακρίβεια.

Μία σημαντική έννοια που παίζει ενεργό ρόλο στη ψηφιακή ανάλυση του ήχου είναι το μέγεθος της μνήμης ή **buffer size** που διαθέτουμε και επηρεάζεται καθαρά από το ποσοστό της μνήμης που διαθέτουμε στην εφαρμογή μας. Όταν έχουμε να υπολογίσουμε τόσο πολλά δεδομένα σε πραγματικό χρόνο θα πρέπει να υπάρχει μία παράμετρος που θα ελέγχει αυτή τη ροή των δεδομένων και αυτή είναι η δουλειά του buffer στον ήχο. Είναι κάτι σαν μία προσωρινή μνήμη, πίνακας που κρατάει δεδομένα ήχου στη περίπτωση μας έτοιμα να μεταφερθούν στη κάρτα ήχου μας και από κει στα ηχεία. Το μέγεθος του buffer είναι κρίσιμο για την εφαρμογή που θα υλοποιήσουμε καθώς άμα δεν ορίσουμε από πριν σωστό μέγεθος μπορεί να επιβαρύνει αρκετά τη μνήμη του υπολογιστή και να μην λειτουργεί ομαλά. (Stearns)

2.3 Φίλτρα ήχου

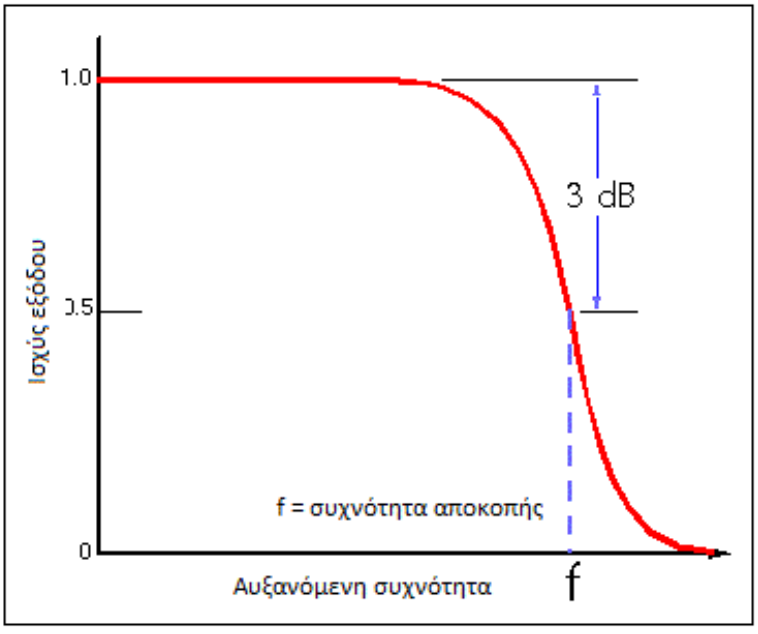
Τα φίλτρα είναι ηλεκτρικά δικτυώματα που αφήνουν να περνούν απαραμόρφωτα ηλεκτρικά σήματα μέσα σε συγκεκριμένες ζώνες συχνοτήτων και ταυτόχρονα μηδενίζουν κάθε άλλο ηλεκτρικό σήμα με συχνότητα έξω από αυτές τις ζώνες. (ΦΙΛΤΡΑ)

Τα φίλτρα διακρίνονται σε **παθητικά** και σε **ενεργά**. Τα παθητικά περιλαμβάνουν μόνο ηλεκτρικά στοιχεία R, L και C. Τα ενεργά περιλαμβάνουν και ενεργητικά στοιχεία (π.χ. τελεστικούς ενισχυτές) ενισχυτές. Τα φίλτρα χρησιμοποιούνται τις ασύρματες και ενσύρματες επικοινωνίες, στην ηλεκτρακουστική κ.α.

Υπάρχουν τέσσερεις βασικές κατηγορίες φίλτρων:

1. Χαμηλοπερατά φίλτρα(low-pass filters):

Τα χαμηλοπερατά φίλτρα αφήνουν να περάσουν απαραμόρφωτα ηλεκτρικά σήματα μέχρι μια ορισμένη συχνότητα αποκοπής f, ενώ μηδενίζουν κάθε ηλεκτρικό σήμα με συχνότητα μεγαλύτερη

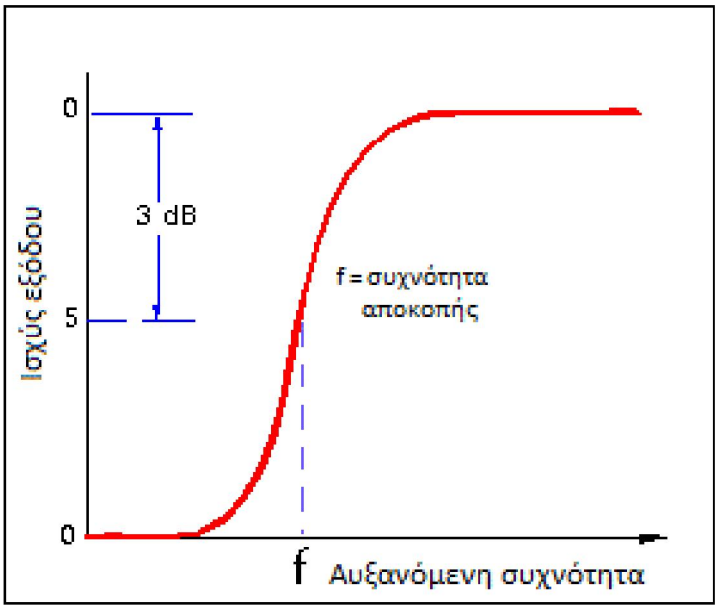


5. Χαμηλοπερατό φίλτρο, αναπαράσταση εξόδου προς συχνότητα

από την f.

2. Υψηλοπερατά φίλτρα(high-pass filters):

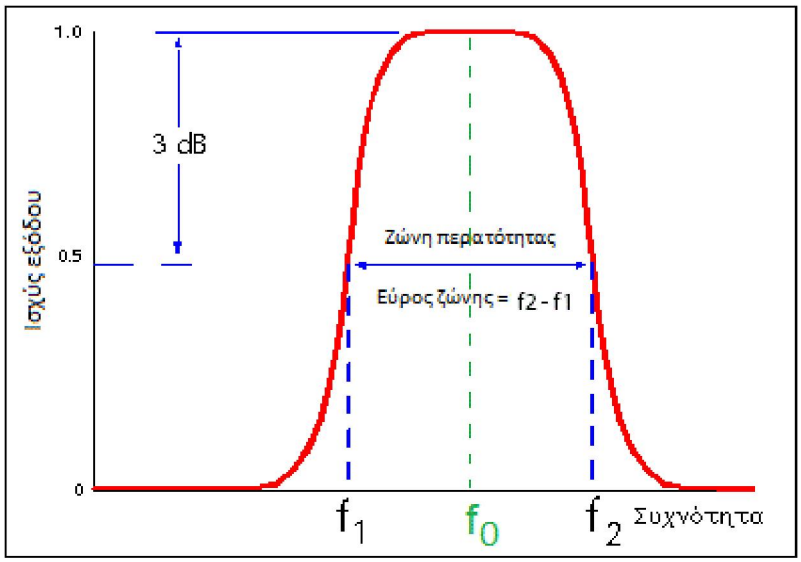
Τα υψηλοπερατά φίλτρα αφήνουν να περάσουν απαραμόρφωτα τα ηλεκτρικά σήματα με συχνότητα μεγαλύτερη από μια ορισμένη συχνότητα αποκοπής f, ενώ μηδενίζουν κάθε ηλεκτρικό σήμα με συχνότητα μικρότερη από την f.



6. Υψηλοπερατό φίλτρο, αναπαράσταση εξόδου προς συχνότητα

3. Ζωνοπερατά φίλτρα(Band-pass filters)

Αφήνουν να περάσουν απαραμόρφωτα τα ηλεκτρικά σήματα με συχνότητα μεταξύ δύο συχνοτήτων f1 και f2 ενώ μηδενίζουν τα σήματα με συχνότητα έξω από αυτή τη ζώνη. Οι συχνότητες f1 και f2 ονομάζονται **συχνότητες αποκοπής** και η f0 ονομάζεται κεντρική συχνότητα του φίλτρου. Το εύρος ζώνης του φίλτρου είναι η τιμή μεταξύ f2 και f1 και μεταφράζεται σε Hz. Το φάσμα των συχνοτήτων μεταξύ f1και f2 ονομάζεται **διέλευσης ζώνης** του φίλτρου.

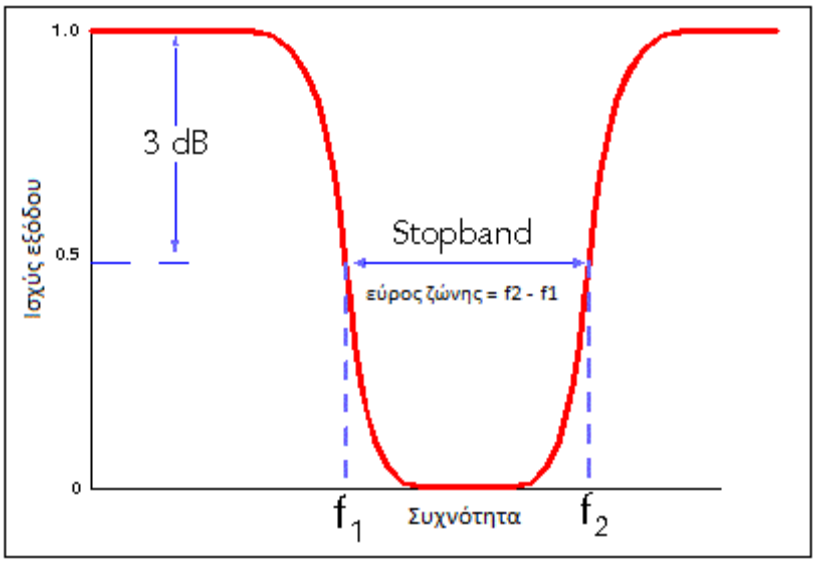


7. Ζωνοπερατό φίλτρο

Τα ζωνοπερατά φίλτρα χρησιμοποιούνται κυρίως στον τομέα των ασύρματων πομπών και δεκτών. Η κύρια λειτουργία ενός τέτοιου φίλτρου σε ένα πομπό είναι να περιορίσει το εύρος ζώνης του σήματος εξόδου στο ελάχιστο απαραίτητο για να μεταφέρει δεδομένα με την επιθυμητή ταχύτητα και στην επιθυμητή μορφή. Σε ένα δέκτη, ένα ζωνοπερατό φίλτρο επιτρέπει σήματα μέσα σε ένα επιλεγμένο φάσμα των συχνοτήτων που πρέπει να ακούσει ή να αποκωδικοποιηθεί, ενώ παρεμποδίζει τα σήματα σε συχνότητες από τα ανεπιθύμητα πάρει μέσα. Ένα ζωνοπερατό φίλτρο βελτιστοποιεί επίσης το λόγο σήματος-προς-θόρυβο (ευαισθησία) του δέκτη.

Στη μετάδοση και λήψη αιτήσεων, καλά σχεδιασμένα ζωνοπερατά φίλτρα, έχοντας το βέλτιστο εύρος ζώνης για την ταχύτητα της επικοινωνίας που χρησιμοποιείται, μεγιστοποιεί τον αριθμό των σημάτων τα οποία μπορούν να μεταφερθούν σε ένα σύστημα, ελαχιστοποιώντας ταυτόχρονα την παρεμβολή και τον ανταγωνισμό μεταξύ των σημάτων. (Rouse)[3]

4. Ζωνοφρακτικά φίλτρα (Band-reject filters)



8. Ζωνοφρακτικό φίλτρο

Τα ζωνοφρακτικά έχουν ακριβώς την αντίθετη λειτουργία με τα ζωνοπερατά. Μηδενίζουν τα ηλεκτρικά σήματα με συχνότητα μεταξύ δύο συχνοτήτων f1 και f2 ενώ αφήνουν να περάσουν απαραμόρφωτα όλα τα σήματα με συχνότητα έξω από αυτή τη ζώνη.

Ψηφιακά φίλτρα

Δεδομένου ότι οι πληροφορίες κωδικοποιούνται με διαφορετικό τρόπο σε αναλογικά και ψηφιακά συστήματα, ο τρόπος με τον οποίο επεξεργάζεται το σήμα είναι, συνεπώς, διαφορετικός. Όλες οι λειτουργίες που μπορούν να εκτελεσθούν σε ένα αναλογικό σήμα όπως **ενίσχυση ή φιλτράρισμα**, μπορούν επίσης να αναπαραχθούν στον ψηφιακό τομέα. Κάθε ψηφιακό κύκλωμα είναι επίσης ένα ανάλογο κύκλωμα, από το ότι η συμπεριφορά οποιουδήποτε ψηφιακού κυκλώματος μπορεί να εξηγηθεί χρησιμοποιώντας τους κανόνες των αναλογικών κυκλωμάτων.

[3] Band pass filters, Rouse Margaret

Η Ψηφιακή επεξεργασία σήματος επιτρέπει την φθηνή κατασκευή μιας ευρείας ποικιλίας **ψηφιακών φίλτρων**. Τα αναλογικά κυκλώματα είναι πιο δύσκολο να σχεδιάσουν, απαιτώντας περισσότερες δεξιότητες, από τα αντίστοιχα ψηφιακά συστήματα. Τις είναι τις από τις κύριους λόγους για τις οποίους τα ψηφιακά συστήματα είναι πιο διαδεδομένα σε σχέση με τις αναλογικές συσκευές. Ένα ανάλογο κύκλωμα πρέπει να είναι σχεδιασμένα με το χέρι, και η διαδικασία είναι πολύ λιγότερο αυτοματοποιημένη από ό, τι στα ψηφιακά συστήματα. (wikipedia)

2.4 Ρυθμός και ανίχνευση ρυθμού

Tempo (ρυθμός)

Στη μουσική ορολογία, το tempo (από τα Ιταλικά tempo = χρόνος, πληθ. tempi) αποτελεί την ένδειξη η οποία δηλώνει τον ρυθμικό χαρακτήρα με τον οποίο πρέπει να εκτελεστεί ένα μουσικό έργο. Η σωστή απόδοση του tempo είναι υψίστης σημασίας για να αποκτήσει το έργο τον χαρακτήρα και το ύφος που επιθυμεί ο συνθέτης. (wikipedia)

Το tempo, όσον αφορά την ταχύτητα, υπολογίζεται μαθηματικά σε χτύπους ανά λεπτό(beats per minute). Σαν συμβολισμός αναγράφεται με πόσους χτύπους ανά λεπτό ισούται η βασική ρυθμική αξία στην οποία είναι γραμμένο το μουσικό κείμενο. Αυτή η ένδειξη ονομάζεται μετρονομική. Για παράδειγμα, αν το κείμενο έχει μετρική ένδειξη 4/4, όπου ρυθμική μονάδα είναι το τέταρτο, η μετρονομική ένδειξη θα δηλώνει πόσα τέταρτα χωρούν μέσα σε ένα λεπτό (π.χ. ♩=120).

Το tempo είναι μία κρίσιμη έννοια τόσο στη σύγχρονη όσο και στη κλασική μουσική. Η εύρεση ,ανίχνευση ρυθμού στη μουσική ανήκει στην κατηγορία ανάλυσης ψηφιακών σημάτων και χρησιμοποιείται ευρέως στο χώρο τις ανάκτησης τις μουσικής πληροφορίας. Το tempo σε ένα μουσικό κομμάτι μπορεί να δώσει αρκετές πληροφορίες το είδος της μουσικής και το ύφος του. Παρακάτω αναφέρονται ενδεικτικά μουσικά είδη και τα κατά μέσο όρο Bpms τους.

Genre	Bpm
Reggae	80
Dub	70-130
Hip hop	65-110
Classic rock	130-170
Pop	130-150
DrumNbass	160-180
Techno	130-160
Dubstep	140

Classical	Bpm/tempo
Largo	40-60
Larghetto	60-66
Adagio	66-76
Andante	76-108
Moderato	108-120
Allegro	120-168
Presto	168-200

9. Bpm για διαφορετικά μουσικά είδη

Beat detection(ανίχνευση παλμού ή ρυθμού ή tempo)

Η προσομοίωση φυσικών φαινομένων που υπακούν σε γνωστές μαθηματικές εξισώσεις είναι πάντα εφικτή. Αλλά τι γίνεται με πιο αφηρημένες έννοιες, τις τα συναισθήματα, τα οποία δεν ακολουθούν νόμους; Τα απλούστερα πράγματα που μπορούμε να αισθανθούμε είναι συχνά τα δυσκολότερα πράγματα για να συλλάβει σε ένα πρόγραμμα ο Η/Υ. Το ίδιο ακριβώς ισχύει και για τον

ρυθμό στη μουσική. Η αίσθηση του ρυθμού για τον άνθρωπο είναι κάτι το φυσιολογικό πως μπορείς να διδάξεις σε μία μηχανή αυτή την αίσθηση; (Patin)

Παρακάτω θα αναλύσουμε τις δύο πιο βασικές μεθόδους που χρησιμοποιούνται από εφαρμογές και προγράμματα πάνω στην ανίχνευση του ρυθμού. Η πρώτη μέθοδος χρησιμοποιεί την ενέργεια που εκπέμπει το μουσικό κομμάτι κατά τη διάρκεια τις αναπαραγωγής του και η δεύτερη χρησιμοποιεί το φάσμα συχνοτήτων του.

2.5 Αλγόριθμοι εύρεσης ρυθμού

2.5.1.Ανάλυση μέσω ενέργειας

Το ανθρώπινο σύστημα ακρόασης καθορίζει τον ρυθμό τις μουσικής από μία ψευδό – περιοδική διαδοχή παλμών(beats). Το σήμα το οποίο συλλαμβάνεται από το αυτί περιέχει μια ορισμένη ενέργεια, αυτή η ενέργεια μετατρέπεται σε ένα ηλεκτρικό σήμα το οποίο ερμηνεύει ο εγκέφαλος. Προφανώς, όσο περισσότερη ενέργεια μεταφέρει ο ήχος , τόσο πιο δυνατά ο ήχος θα φαίνεται. Αλλά τις ήχος θα ακούγεται σαν ένα beat(παλμό) μόνο αν η ενέργειά του είναι πολύ ανώτερη σε σχέση με τις γειτονικούς ήχους. Συνεπώς, αν το αυτί παρακολουθεί ένα μονότονο ήχο με τις κορυφές ενέργειας θα ανιχνεύσει παλμούς, τις, αν παρακολουθεί ένα συνεχή δυνατό ήχο δεν θα αντιληφθεί κάποια beats. Έτσι, μπορούμε να πούμε ότι τα beats είναι μεγάλες διακυμάνσεις τις ηχητικής ενέργειας. (Patin)

Σε αυτό το μοντέλο για να μπορέσουμε να ανιχνεύσουμε διακυμάνσεις στην ενέργεια θα πρέπει να συγκρίνουμε τη **μέση διακύμανση τις ηχητικής ενέργειας** με τη **στιγμιαία διακύμανση**. Τις πούμε ότι δουλεύουμε σε λειτουργία στερεοφωνικού ήχου με δύο λίστες τιμών: (an) και (bn).Η (an) περιέχει τη λίστα για τιμές του πλάτους που λαμβάνονται κάθε Τε δευτερόλεπτα για το αριστερό κανάλι και η (bn) για το δεξί κανάλι. Η άμεση ενέργεια θα είναι στην πραγματικότητα η ενέργεια που περιέχεται σε 1024 δείγματα (1024 τιμές των α [n] και b [n]), 1024 δείγματα αντιπροσωπεύουν περίπου 1/50 του δευτερολέπτου, το οποίο είναι λίγο πολύ «στιγμιαία». Η άμεση ενέργεια αυτή πρέπει να συγκριθεί με το γειτονική μέση ενέργεια, για παράδειγμα, αν ένα τραγούδι έχει έντονο τελείωμα, η ενέργεια που περιέχεται σε αυτό δεν θα πρέπει να επηρεάσει το ρυθμό ανίχνευσης στην αρχή του κομματιού. Έτσι θα υπολογίσει ο μέσος όρος τις ενέργειας για 44032 δείγματα τα οποία είναι περίπου 1 δευτερόλεπτο. Αυτό το 1 δευτερόλεπτο είναι τις συμβιβασμός μεταξύ του να έχουμε “μακρινές” κορυφές ενεργειών και “κοντινές” κορυφές για να πετύχουμε μία πολύτιμη σύγκριση.

Έτσι μπορούμε να πούμε ότι ο αλγόριθμος για κάθε 1024 δείγματα λειτουργεί ως εξής:

- Υπολογίζουμε την **άμεση ενέργεια** του ήχου στα πρώτα 1024 δείγματα και για τα δύο κανάλια a(n) και b(n)

$$e = e_{stereo} = e_{right} + e_{left} = \sum_{k=i_0}^{i_0+1024} a[k]^2 + b[k]^2$$

- Υπολογίζουμε την **μέση ενέργεια του ήχου** για 44032 δείγματα σε κομμάτια των 1024 (44032/1024=43)

$$< E >= \frac{1}{43} \times \sum_{i=0}^{43} (E[i])^2$$

- Υπολογίζουμε τη **διακύμανση V** των πηγών ενέργειας στην <E> με τον ακόλουθο τύπο:

$$V = \frac{1}{43} \times \sum_{i=0}^{43} (E[i] - < E >)^2$$

Και στη συνέχεια μέσω τις διακύμανσης V παίρνουμε μία τιμή για τη σταθερά C(όσο μεγαλύτερη διακύμανση έχουμε τόσο μικρότερη σταθερά θα πάρουμε. Η σταθερά C αλλάζει αναλόγως το είδος μουσικής. Χρησιμοποιώντας μια γραμμική σταδιακή μείωση των «C» με το «V» και μια γραμμική παλινδρόμηση Π.χ. V= 200, C → 1.0 , V= 25, C → 1.45)

$$C = (-0.0025714 \times V) + 1.5142857$$

- Μετακινούμαστε στην επόμενη θέση του πίνακα E .Δημιουργούμε χώρο για τις καινούργιες του τιμές με το να αποβάλουμε τις προηγούμενες.
- Τοποθετούμε τις επόμενες τιμές ενέργειας από τον e στη θέση E[0].
- Κάνουμε σύγκριση ‘e’ σταθεράς με C*<lessthanE> και άμα είναι μεγαλύτερο βρήκαμε beat.

Ο παραπάνω αλγόριθμος λειτουργεί ικανοποιητικά για είδη μουσικής τύπου techno, rap όπου ο ρυθμός είναι ευδιάκριτος και ο θόρυβος περιορίζεται σε χαμηλά επίπεδα. Τι γίνεται τις όταν έχουμε ένα κομμάτι πιο πολύπλοκο με ακανόνιστο ρυθμό; Ο αλγόριθμος εντοπίζει κορυφές και μπορεί αρκετά εύκολα να παραλείψει το beat αν είναι επισκιασμένο από τις ήχους. Πιο συγκεκριμένα έχουμε μία κιθάρα και μία φυσαρμόνικα και ταυτοχρόνως παράγουν ήχο τις έντασης αλλά παρατηρούμε ότι η νότα από τη κιθάρα και τη φυσαρμόνικα δίνουν μία αίσθηση ρυθμού που αυτό οφείλεται καθαρά στη τονικότητα του κάθε οργάνου. Ο αλγόριθμος που αναλύσαμε δεν θα μπορούσε να αντιληφθεί το ρυθμό αυτό και εκλαμβάνει τον ήχο σαν ένα διαρκή θόρυβο χωρίς αυξομειώσεις στην ένταση. Ο παρακάτω αλγόριθμος δεν θα έχει αυτό το πρόβλημα γιατί δεν θα δέχεται σαν όρισμα την ένταση του κομματιού αλλά τη τονικότητα του(συχνότητα του σήματος).

2.5.2. Ανάλυση σε ζώνες συχνοτήτων

Όπως εξηγήσαμε παραπάνω ο προηγούμενος αλγόριθμος είναι ιδανικός σε μουσικά είδη που είναι ευδιάκριτος ο ρυθμός και ο θόρυβος είναι περιορισμένος σε χαμηλά επίπεδα. Το πρόβλημα όμως με αυτόν είναι ότι θα μπορούσε να παρουσιάσει αρκετά προβλήματα σε πιο θορυβώδη είδη, όπως τραγούδια σε ροκ ή ποπ μουσική.Τώρα ο δεύτερος αλγόριθμος που θα αναλύσουμε ξεπερνάει αυτά τα προβλήματα με το να του δώσουμε την ικανότητα να καθορίσει σε ποια ζώνη συχνότητας έχουμε ένα beat και αν είναι αρκετά ισχυρό ώστε να το λάβει υπόψη. Βασικά θα προσπαθήσει να εντοπίσει μεγάλες διακυμάνσεις ηχητικής ενέργειας, σε συγκεκριμένες ζώνες συχνοτήτων, όπως ακριβώς και στην τελευταία ανάλυσή μας μόνο που αυτή τη φορά θα είμαστε σε θέση να διαχωρίσουμε τα beats σε σχέση με το "χρώμα" τους (ζώνη συχνότητας).Έτσι μπορούμε να δώσουμε έμφαση σε beats χαμηλότερης ή υψηλότερης συχνότητας. Πάμε τώρα να δούμε πως δουλεύει ο αλγόριθμος αυτός. (Patin)

Τα σήματα πηγής εξακολουθούν να προέρχονται (an) και (bn) ή μπορεί να ληφθεί από ένα αρχείο ή απευθείας από μια συνεχή ροή μικροφώνου ή line in. Κάθε φορά που έχουμε συσσωρεύσει 1024 νέα δείγματα, θα τα περνάμε στο πεδίο της συχνότητας με μετασχηματισμό Fourier (FFT). Θα έχουμε έτσι ένα φάσμα συχνοτήτων με 1024 δείγματα. Στη συνέχεια, διαιρούμε το φάσμα σε υποζώνες πχ. 32. Όσο περισσότερες οι υποζώνες, τόσο πιο ευαίσθητος θα είναι ο αλγόριθμος, αλλά τόσο πιο δύσκολο θα γίνει να προσαρμοστεί σε περισσότερα διαφορετικά είδη τραγουδιών. Στη συνέχεια υπολογίζουμε την ηχητική ενέργεια που περιέχεται σε καθεμία από τις υποζώνες και τη συγκρίνουμε με την πρόσφατη μέση ενέργεια που αντιστοιχεί σε αυτή την υποζώνη. Αν μία ή περισσότερες υποζώνες έχουν μια ενέργεια ανώτερη από τον μέσο όρο τους τότε έχουμε εντοπιστεί beat.

Για 1024 δείγματα ο αλγόριθμος λειτουργεί ως εξής:

- Εφαρμόζουμε το μετασχηματισμό Fourier στα καινούργια 1024 δείγματά μας που παίρνουμε από τους a(n) και b(n).Ο μετασχηματισμός εισάγει ένα σύνθετο αριθμητικό σήμα. Θα πούμε (an) είναι το πραγματικό μέρος του σήματος και (bn) το φανταστικό μέρος. Έτσι ο μετασχηματισμός θα γίνει στις 1024 μιγαδικές τιμές των:

$$(a_n) + i \times (b_n)$$

- Από το μετασχηματισμό Fourier παίρνουμε 1024 μιγαδικούς αριθμούς. Υπολογίζουμε το τετράγωνο της μονάδας τους και τους αποθηκεύουμε σε ένα νέο buffer 1024 θέσεων. Ο buffer αυτός περιέχει τα πλάτη των συχνοτήτων μας.
- Χωρίζουμε το buffer σε 32 υποζώνες(subbands), υπολογίζουμε την ενέργεια για τη κάθε μία υποζώνη και την αποθηκεύουμε στον Es[i](πίνακας 32 θέσεων για κάθε 'i' υποζώνη).

$$Es[i] = \frac{32}{1024} \times \sum_{k=i\times 32}^{(i+1)\times 32} B[k]$$

- Τώρα, για κάθε υποζώνη «i» αντιστοιχεί ένας buffer για τις ενέργειες που ονομάζεται (Ei). Αυτός περιέχει τους τελευταίους 43 υπολογισμούς ενέργειας για την 'i' υποζώνη. Εμείς τον υπολογισμό της μέσης ενέργεια <E> για την 'i' υποζώνη απλά με τη χρήση:

$$< Ei >= \frac{1}{43} \times \sum_{k=0}^{42} Ei[k]$$

- Μετακινούμαστε στον <Ei> προς τα δεξιά. Έχουμε κάνει το χώρο για τη νέα ενεργειακή αξία της «i» υποζώνης και αποβάλουμε τη παλαιότερη.
- Τοποθετούμε τις καινούργιες ενεργειακές τιμές από την επόμενη υποζώνη Es[i] στον Ei[0].
- Για κάθε υποζώνη 'i' αν Es[i] > (C*<E>) έχουμε beat.

Η σταθερά C δεν έχει καμία σχέση με τη σταθερά C του πρώτου αλγόριθμου γιατί εδώ έχουμε να κάνουμε με ενέργεια σε υποζώνες συχνοτήτων. Καταλαβαίνουμε ότι ο δεύτερος αλγόριθμος είναι πιο ακριβής από τον πρώτο γιατί δεν αφήνει σημαντικά στοιχεία στο κομμάτι μας να περνάνε απαρατήρητα και να χάνονται μέσα στο θόρυβο. Ο αλγόριθμος μπορεί να βελτιωθεί με το να αυξήσουμε τις υποζώνες από 32 σε 64. Αυτό θα απαιτεί βέβαια περισσότερους υπολογισμούς άρα περισσότερη μνήμη αλλά θα είναι πιο ακριβής στην ανίχνευση των beats. (Patin)[4]

[4] Beat Detection Algorithms by Frederik Patin

3. State of the art

3.1 Απεικόνιση του ήχου (music visualization)

Η διαδικασία απεικόνισης του ήχου ανήκει στο τομέα της επεξεργασίας και ανάλυσης του ήχου και χρησιμοποιείται για την εξαγωγή εικονικών δεδομένων. Οι τεχνικές απεικόνισης σημάτων εκτείνονται από τις πιο απλές(π.χ. Προσομοίωση εικόνας ενός παλμογράφου) μέχρι πιο πολύπλοκες οι οποίες συχνά περιλαμβάνουν ένα πλήθος από σύνθετες επιδράσεις. Οι εφαρμογές πάνω στην οπτικοποίηση δίνουν μία πιο ολοκληρωμένη εικόνα για το σήμα μας και λειτουργούν ως επιπρόσθετο κομμάτι της ψηφιακής ανάλυσης. Τα τελευταία χρόνια η οπτικοποίηση του ήχου έχει πάρει άλλες διαστάσεις και χρησιμοποιείται στη δημιουργία οπτικού υλικού σε συνδυασμό με τη μουσική σαν παρακλάδι της ψηφιακής τέχνης. (wikipedia)

10. (Atari Video Music)

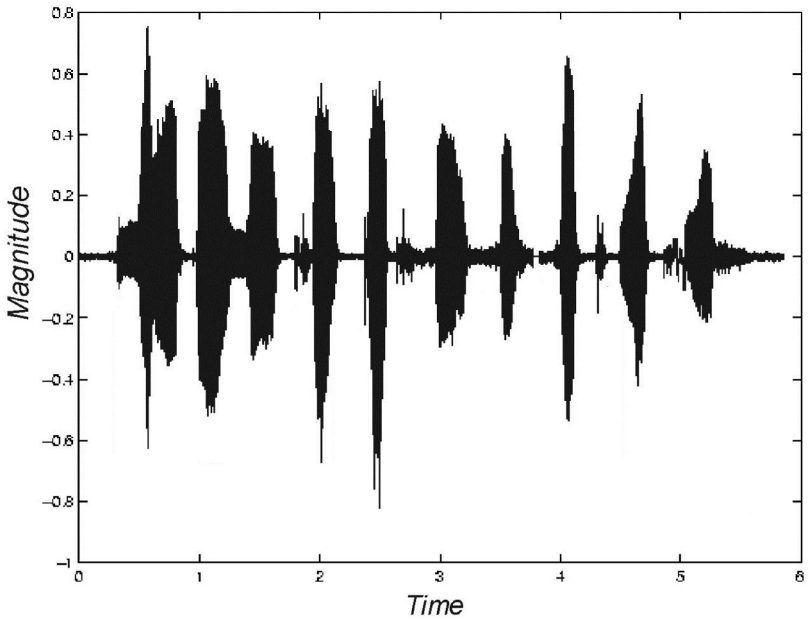
Music information retrieval(MIR) ή αλλιώς ανάκτηση μουσικής πληροφορίας είναι μικρός επιστημονικός τομέας που συνδυάζει μουσικολογία, ψυχολογία και ψηφιακή ανάλυση του ήχου. Οι εφαρμογές πάνω σε αυτό τον τομέα χωρίζονται σε τρεις κατηγορίες .Χαμηλού, μεσαίου και υψηλού επιπέδου εφαρμογές αναλόγως με τη πληροφορία που ανακτούμε μέσω αυτών .Οι εφαρμογές που ανήκουν στο μεσαίο και υψηλό επίπεδο μας παρέχουν πληροφορίες όπως τονικότητα, είδος μουσικής και σημασιολογία(χαρά, θλίψη) ενός μουσικού κομματιού .Εμείς όμως θα ασχοληθούμε με τα χαρακτηριστικά του χαμηλότερου επιπέδου που είναι η ανάλυση του ήχου και τις πληροφορίες που αντλούμε από αυτή τη διαδικασία. (Pramerdorfer)[5]

[5] An Introduction to Processing and Music Visualization, PramerDorfer

3.2 Ανεπεξέργαστη αναπαράσταση ηχητικού σήματος

3.2.1 Απεικόνιση κυματομορφής

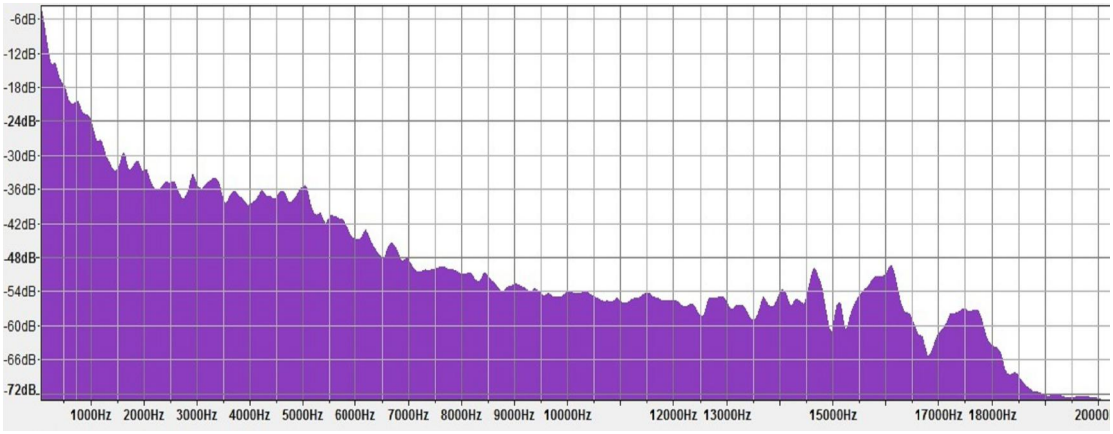
Η ανάλυση ανεπεξέργαστων δεδομένων πλαισίου αναφέρεται στην ανάλυση και απεικόνιση του ήχου χωρίς περαιτέρω διαδικασία .Στο τομέα ανάλυσης του ήχου υπάρχουν ήδη αρκετές αναπαραστάσεις με τις πιο διαδεδομένες να είναι τρεις. Αναπαράσταση του ήχου στο πεδίο του χρόνου ,αναπαράσταση στο πεδίο των συχνοτήτων και στο φάσμα συχνοτήτων. Η πιο γνωστή, απλή και πιο διαδεδομένη από όλες είναι η αναπαράσταση ενός ψηφιακού σήματος ως κυματομορφή (**waveform**).Η εφαρμογή αναπαριστά το ψηφιακό μας σήμα έχοντας στο κάθετο άξονα το πλάτος (Amplitude, Magnitude) του σήματος μας και στον οριζόντιο το χρόνο του. Το πλάτος του σήματος συνηθίζεται να συσχετίζεται με την ένταση του. Όσο μεγαλύτερο πλάτος έχουμε τόσο μεγαλύτερή θα είναι και η ένταση μας και όσο μικρότερο πλάτος τόσο μικρότερη η ένταση. Με τη συγκεκριμένη αναπαράσταση έχουμε ένα μειονέκτημα και αυτό είναι ότι δε μπορεί να μας δώσει με ακρίβεια τη συχνότητα(ή συχνότητες) του σήμα μας. Έτσι για αυτό το σκοπό έχουμε την **αναπαράσταση στο πεδίο των συχνοτήτων.** (Stearns)



11. Αναπαράσταση κυματομορφής στο πεδίο του χρόνου

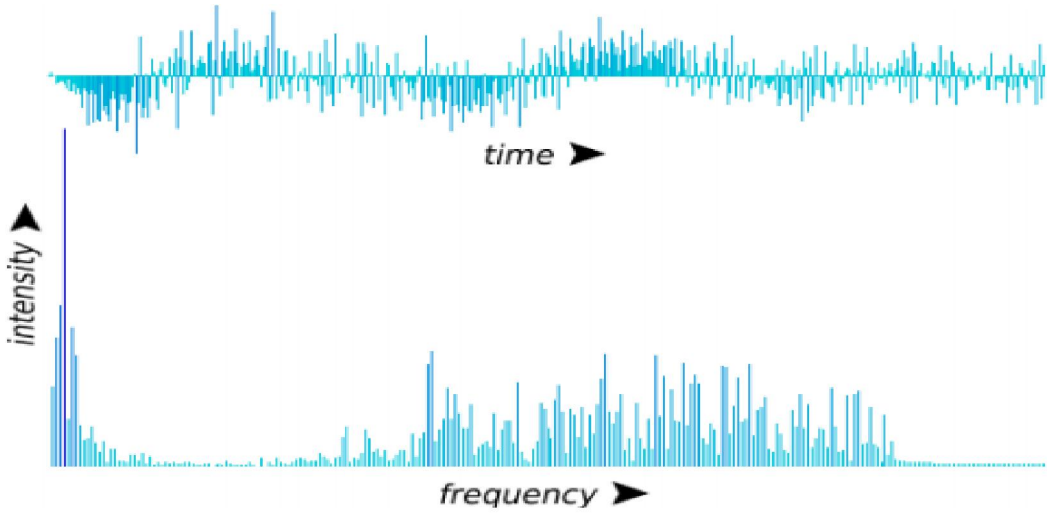
3.2.2 Απεικόνιση στο πεδίο των συχνοτήτων(frequency domain)

Η αναπαράσταση στο πεδίο των συχνοτήτων σε σύγκριση με της απλής κυματομορφής έχει την διαφορά ότι δεν βρίσκεται στο πεδίο του χρόνου αλλά στο πεδίο της συχνότητας. Στον οριζόντιο άξονα βρίσκονται οι συχνότητες από τις οποίες απαρτίζεται το σήμα και όχι ο χρόνος και στο κάθετο άξονα έχουμε το πλάτος ή την ένταση της κάθε συχνότητας(Amplitude,Magnitude). (Pramerdorfer)



12. Αναπαράσταση σήματος στο πεδίο των συχνοτήτων

Πολλά χαρακτηριστικά ήχου μπορεί να υπολογιστούν από το μετασχηματισμό του σήματος στο πεδίο των συχνοτήτων. Κατά τη διαδικασία μετασχηματισμού, το σήμα εισόδου αποσυντίθεται σε συνιστώσες συχνότητες του, που ονομάζεται το **φάσμα των συχνοτήτων** του. Ακριβέστερα, το σήμα εισόδου αποσυντίθεται σε ένα άθροισμα αριθμών που αντιπροσωπεύουν το πλάτος και τη φάση των διαφόρων ημιτονοειδών συνιστωσών του σήματος εισόδου. Σε πολλές εφαρμογές, όπως την επεξεργασία του ήχου, οι πληροφορίες για τη φάση του σήματος συνήθως δεν είναι σημαντική και ως εκ τούτου απορρίπτεται.



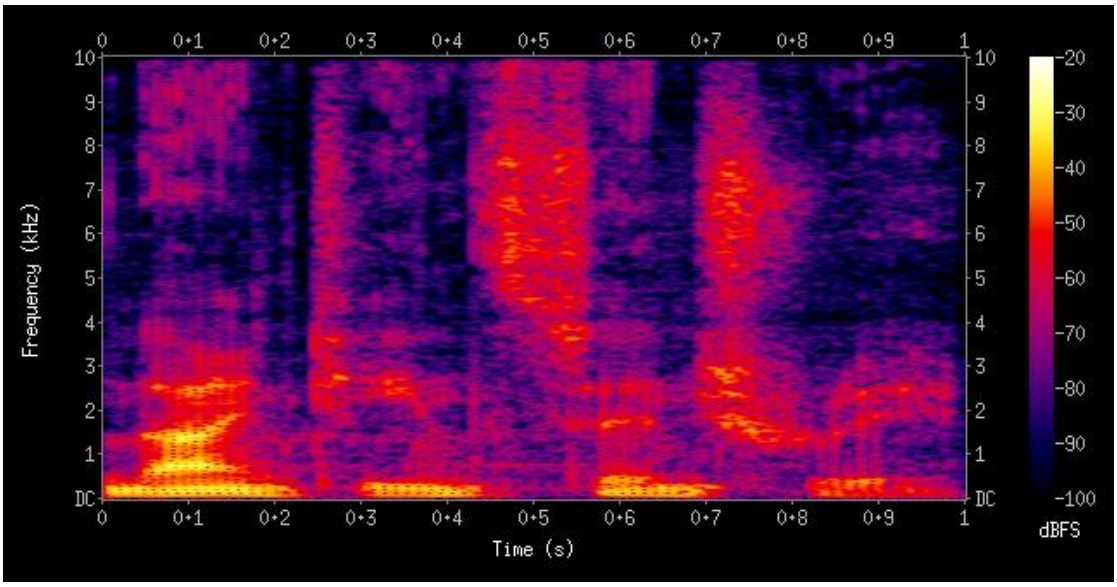
13. Πεδίο του χρόνου και συχνότητας

Η αναπαράσταση πλαισίου του σήματος της εικόνας 13 με 1024 δείγματα στο πεδίο του χρόνου (πάνω) και στο πεδίο συχνοτήτων(κάτω).

Επειδή το φάσμα αποκαλύπτει τις συχνότητες από τις οποίες αποτελείται ένα ηχητικό σήμα, πολλά χρήσιμα χαρακτηριστικά μπορούν να παραχθούν από αυτό. Ο μετασχηματισμός διεξάγεται συνήθως με την εφαρμογή ενός διακριτού μετασχηματισμού Fourier σε κάθε καρέ του σήματος. Η προσέγγιση αυτή ονομάζεται **Short-Time Fourier Transform (STFT).**Ενώ με τη συγκεκριμένη μέθοδο έχουμε μία ακριβή εικόνα για τις συχνότητες δεν μπορούμε να ξέρουμε το πως το σήμα μας εξελίσσεται, αλλάζει σε σχέση με το χρόνο. Είναι σαν να βλέπουμε μία **στιγμιαία εικόνα** του ήχου, της συχνότητας και του πλάτους. Πέρα από το φάσμα συχνοτήτων έχουμε και το φάσμα Mel μια απεικόνιση που είναι πιο κοντά στην ανθρώπινη αντίληψη για τον ήχο και πιο απλή για το λόγο το ότι τα δεδομένα μας είναι συσσωρευμένα μετατρέποντας ένα πιο φιλικό αποτέλεσμα προς το χρήστη.

3.2.3 Απεικόνιση μέσω φασματογράφου

Η πιο ολοκληρωμένη τεχνική απεικόνισης του ήχου είναι του φασματογράφου (**spectrogram ή spectrograph**).Στο αναπαράσταση αυτή στο κάθετο άξονα έχουμε τις συχνότητες του σήματος και στον οριζόντιο το χρόνο .Πέρα από αυτό παρατηρούμε διάφορα χρωματικά επίπεδα που συμβολίζουν την ένταση ή πλάτος του σήματος για τη δεδομένη χρονική στιγμή και συχνότητα. Όπως παρατηρούμε στις πιο μεγάλες εντάσεις το χρώμα είναι πιο ανοιχτό ενώ για τις χαμηλές είναι πιο σκούρο. Το φασματογράφημα χρησιμοποιείται στην αναγνώριση φωνητικών λέξεων την ανάλυση ήχων που παράγουν τα ζώα και σε αρκετές άλλες εφαρμογές. (Stearns)[6]



14 .Αναπαράσταση από φασματογράφο

Παραπάνω αναφέραμε τις πιο βασικές τεχνικές ανάλυσης του ήχου σε εικόνα. Πάνω σε αυτές βασίζονται το μεγαλύτερο μέρος των εφαρμογών οπτικοποίησης του ήχου.

Χρησιμοποιώντας λοιπόν τις παραπάνω τεχνικές αντλούμε πληροφορίες για το μουσικό μας κομμάτι και με τη χρήση κατάλληλων αλγορίθμων εκμεταλλευόμαστε αυτές τις πληροφορίες για να δημιουργήσουμε οπτικό υλικό. Όμως όλοι οι παραπάνω μέθοδοι ανάλυσης του ηχητικού μας σήματος μας παρουσιάζουν το ηχητικό μας σήμα ακατέργαστο. Παρακάτω αναφέρουμε πως αλγόριθμοι όπως οι αλγόριθμοι εύρεσης beat σε ένα μουσικό κομμάτι χρησιμοποιούν αυτές τις ακατέργαστες πληροφορίες για να εξάγουν οπτικό υλικό.

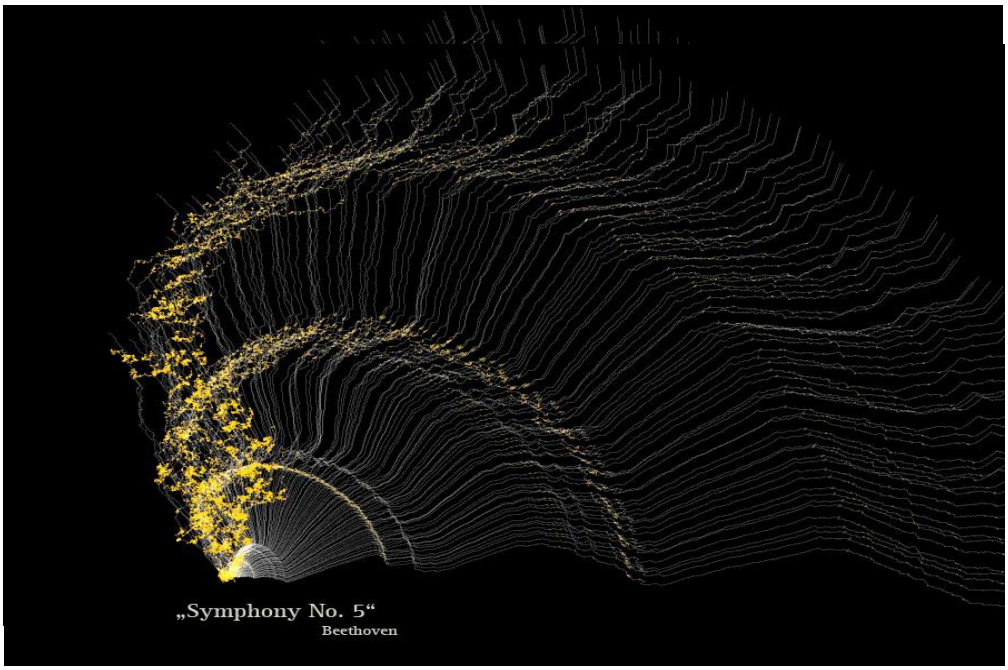
3.3 Επεξεργασμένη αναπαράσταση(generative visualization)

Τα εικονικά στοιχεία που παίρναμε στη προηγούμενη ενότητα ήταν βασισμένα σε ακατέργαστα δεδομένα και μας εξυπηρετούν σαν επιπρόσθετη πληροφορία για το ηχητικό μας σήμα. Στόχος μας είναι με τις απεικονίσεις αυτές να μας παρουσιάζουν τα δεδομένα μας όσο πιο καθαρά και με σαφήνεια γίνεται και να μην δημιουργήσουμε εσφαλμένα αποτελέσματα για το σήμα μας. Σε αυτή την ενότητα θα παρουσιάσουμε τεχνικές που δεν έχουν σαν σκοπό την εξαγωγή πληροφοριών από τα δεδομένα μας αλλά η δημιουργία αφηρημένου και αισθητικά ελκυστικού εικονικού υλικού που

[6] Music Production, Coursera Course, Loudon Sterns

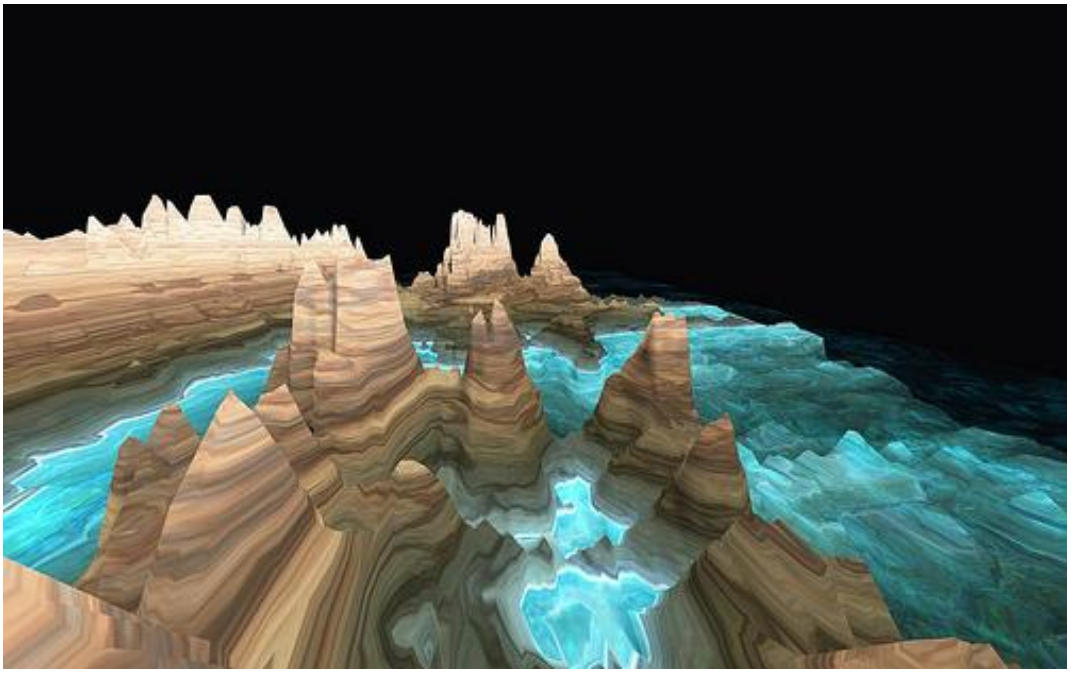
ανταποκρίνεται με ορισμένα στοιχεία του μουσικού μας κομματιού όπως συχνότητες ή το πλάτος. (Pramerdorfer)

Μια προσέγγιση για τις παραγωγικές απεικονίσεις είναι να ξεκινήσουμε με τη κατανομή συχνότητας των δεδομένων ήχου και τέλος να τροποποιήσουμε και να χρησιμοποιήσουμε αυτά τα δεδομένα με κάποιον αφηρημένο τρόπο. Ένα πολύ καλό παράδειγμα είναι το Narratives 2.0 όπου τα διαφορετικά κανάλια του κομματιού αντιπροσωπεύονται από ευθείες γραμμές που ξεκινάν από το κέντρο και ξεδιπλώνονται μέσα στο χώρο αναλόγως την ένταση της συχνότητας του κάθε κομματιού. Στα σημεία που παρατηρούμε αποχρώσεις του πορτοκαλί έχουμε τις πιο υψηλές σε ένταση συχνότητες.



15.Narratives application

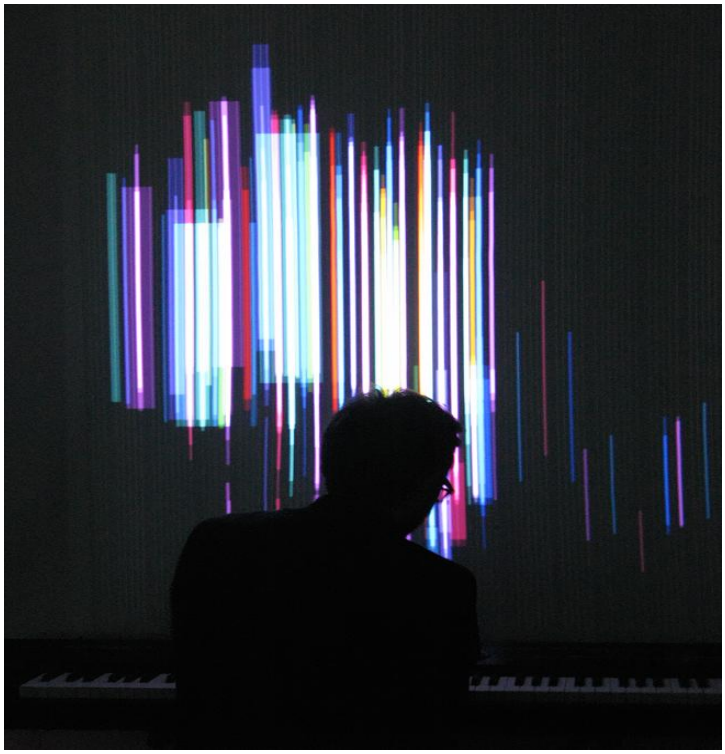
Μια παρόμοια εφαρμογή είναι το Audio-driven Landscape(δημιουργία τοπίου με βάση τον ήχο) του Robert Hogin(2010).Η εφαρμογή αυτή χρησιμοποιώντας μετασχηματισμό Fourier(FFT) δημιουργεί ένα είδος τρισδιάστατης απεικόνισης πεδίου συχνοτήτων και με το πλάτος να χρωματίζει ανάλογα την ένταση του. (flight404)



16.Audio driven Landscape application

Ο σκοπός είναι η δημιουργία ενός τοπίου με χαμηλές εντάσεις να παράγουν αποχρώσεις του γαλάζιου και τις πιο υψηλές αποχρώσεις του καφέ. Αρχικά ο ήχος περνάει από μετασχηματισμό Fourier ,στη συνέχεια κόβει τις πολύ υψηλές και χαμηλές συχνότητες για να επικεντρωθεί στις μεσαίες ζώνες συχνοτήτων. Το αποτέλεσμα μετά γίνεται πιο ομαλό με τη χρήση φίλτρων υπολογίζοντας έτσι το κατά μέσω όρο των συχνοτήτων του και τέλος κάνοντας χρήση κατάλληλων textures του δίνει μία πιο φυσική αίσθηση.

Εξηγώντας το πως καταφέραμε να πάρουμε ένα τέτοιο αποτέλεσμα καταλαβαίνουμε από τη αλλοίωση περνάνε τα δεδομένα μας. Στην επόμενη εφαρμογή ο Jonas Friedemann Heuer δημιούργησε το **Clavilux 2000,** ένα αρμόνιο το οποίο θα παράγει οπτικά εφέ σύμφωνα με το ύφος και την ένταση του κομματιού. Για κάθε νότα που παίζεται στο αρμόνιο ένα νέο οπτικό στοιχείο εμφανίζεται σε μορφή λωρίδας, η οποία ακολουθεί σε διαστάσεις, θέση και το χρώμα της, ο τρόπος με τον συγκεκριμένο κλειδί παίχτηκε: Το μήκος και η κάθετη θέση δείχνουν την ταχύτητα, το πλάτος της λωρίδας αντανακλά τη διάρκεια της κάθε νότας. Αξίζει να αναφερθεί ότι η επιλογή των χρωμάτων δεν είναι τυχαία.



17.Clavilux 2000

Τέλος άλλη μία παρόμοια τεχνική εφαρμόστηκε στο βιντεοκλίπ των **Radiohead** στο κομμάτι **Bodysnatchers** όπου τα φωνητικά και κάθε όργανο επηρέαζε το βίντεο με διαφορετικό τρόπο. Στο βιντεοκλίπ φαίνονται τα κλαδιά ενός δέντρου να ανοίγουν ,παραλλήλως να δημιουργούνται κάποια μοτίβα γύρω από τα κλαδιά και χρώματα σε αλληλεπίδραση με το κομμάτι.

18.Bodysnatchers video instance

Σύμφωνα με τον δημιουργό **Glenn Marshall** η διαδραστικότητα του βίντεο λειτουργεί ως εξής:

1. Το μπάσο δημιουργεί την κόκκινη σκίαση και πάλλεται ρυθμικά σύμφωνα με αυτό
2. Η πρώτη κιθάρα επηρεάζει την ένταση τις φωτεινότητας
3. Τα πρίμα επηρεάζουν το μέγεθος των sprites
4. Τα φωνητικά επηρεάζουν και αυτά επιπρόσθετα στο μέγεθος και την κατεύθυνση των sprites τις και τα εφέ των αστερισμών που εξελίσσονται στο φόντο του κλιπ.

Τέλος όλη η ένταση του κομματιού επηρεάζει τη λήψη όπου στα δυνατά σημεία του εστιάζει και στα χαμηλά απομακρύνεται.

3.4 Εφαρμογές στο χώρο

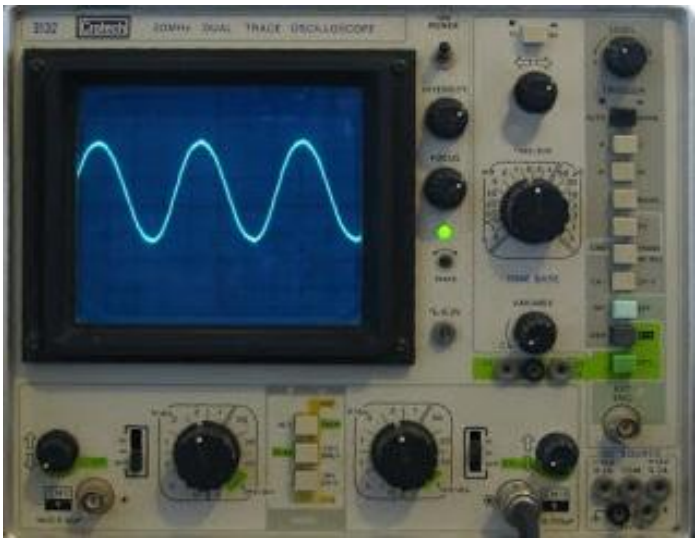
Οι εφαρμογές στο χώρο της οπτικοποίησης του ήχου χωρίζονται σε δύο βασικές κατηγορίες ανάλογα με τη σκοπιμότητα τους.

Για **ερευνητικούς** σκοπούς στο χώρος ανάλυσης ψηφιακών σημάτων χρησιμοποιούμε τις τεχνικές οπτικοποίησης για να εξάγουμε επιπρόσθετες πληροφορίες για τον ήχο μας. Στο τομέα αυτό τα αποτελέσματα που παίρνουμε μέσω της ψηφιακής ανάλυσης πρέπει να είναι αξιόπιστα άρα θα πρέπει και οι εφαρμογές μας να εστιάζουν σε αυτό το κομμάτι.

Για **αισθητικούς και καλλιτεχνικούς** σκοπούς στο χώρο της ψηφιακής τέχνης ή video art όπου εξάγουμε, παράγουμε οπτικό υλικό από τον ήχο. Τέτοιο οπτικό υλικό χρησιμοποιείται σε

συνδυασμό με τη μουσική σε ζωντανές συναυλίες, καλλιτεχνικές παραστάσεις για τη δημιουργία ενός περισσότερο ατμοσφαιρικού περιβάλλοντος .

Παλμογράφος(Oscilloscope) είναι ένας τύπος ηλεκτρονικού μέσου δοκιμής που επιτρέπει την παρατήρηση των συνεχώς μεταβαλλόμενων τάσεων ενός σήματος, και παρουσιάζεται συνήθως ως μια δισδιάστατη γραφική παράσταση χρησιμοποιώντας τον κάθετη Y-άξονα, συναρτήσει του χρόνου (οριζόντια ή X-άξονας).Παλμογράφοι χρησιμοποιούνται συνήθως για την ακριβή παρατήρηση της κυματομορφής ενός ηλεκτρικού σήματος. Επιτρέπει την μέτρηση κορυφής σε κορυφή(peak to peak) τάσης μιας κυματομορφής, τη συχνότητα των περιοδικών σημάτων, το χρόνο μεταξύ των παλμών, το χρόνος που απαιτείται για ένα σήμα να ανέλθει σε πλήρες πλάτος (χρόνος ανόδου) και το σχετικό χρονισμό αρκετών σχετικών σημάτων. (wikipedia)



19.Παλμογράφος

Εφαρμογές φασματογράφου

Η εικονική απεικόνιση του φάσματος μέσω **φασματογράφου** χρησιμοποιείται σε αρκετούς τομείς και αποτελεί ένα κρίσιμο κομμάτι της ψηφιακής ανάλυσης συχνοτήτων και όχι μόνο. Παρακάτω αναφέρουμε τις πιο βασικές εφαρμογές του φασματογράφου. (wikipedia)

- Αρχικά αναλογικά φασματογραφήματα είχαν εφαρμοστεί σε ένα ευρύ φάσμα τομέων όπως στη μελέτη των ήχων που αναπαρήγαγαν τα πτηνά και μετέπειτα με βελτίωση της τεχνολογίας και του εξοπλισμού σε όλα τα ζώα. Συγκεκριμένα τα διακριτά χαρακτηριστικά της FM διαμόρφωσης είναι πιο κατανοητά μέσω της απεικόνισης του φασματογραφήματος.
- Είναι χρήσιμες για την παροχή βοήθειας στην αντιμετώπιση ελαττωμάτων του λόγου και ορθοφωνίας για το τμήμα του πληθυσμού που αντιμετωπίζει κώφωση.
- Οι μελέτες της φωνητικής και σύνθεσης ομιλίας διευκολύνεται με τη χρήση των φασματογραφημάτων.

- Μερικά σύγχρονα μουσικά είδη δημιουργούνται χρησιμοποιώντας φασματογραφήματα ως ενδιάμεσο μέσο. Αλλάζει την ένταση των διαφορετικών συχνοτήτων, ή ακόμη δημιουργεί νέες, και στη συνέχεια εφαρμόζοντας ανάστροφο μετασχηματισμό.
- Χρησιμοποιείται από τις γεωλογικές υπηρεσίες στο τομέα της σεισμολογίας.

DAW (Digital Audio Workstations) software

Ένα Digital Audio Workstation (DAW) είναι ένα ηλεκτρονικό σύστημα που προορίζετε αποκλειστικά ή κατά κύριο λόγο για την καταγραφή, επεξεργασία και αναπαραγωγή ψηφιακού ήχου. Τα σύγχρονα DAWs είναι λογισμικά που τρέχουν σε υπολογιστές σε συνοδεία με interfaces ήχου. Πιο συγκεκριμένα τα DAW λογισμικά είναι προσομοιωτές εργαλείων που χρησιμοποιούνται στο τομέα παραγωγής και επεξεργασίας του ήχου. (Stearns)

Ένα σύστημα επεξεργασίας του ήχου, βασισμένο σε υπολογιστή περιέχει τα εξής: έναν H/Y, μία εξωτερική κάρτα ήχου(αλλιώς μετατροπέας ήχου ή audio interface) ,ένα λογισμικό για ψηφιακή επεξεργασία ήχου και τουλάχιστον μία συσκευή εισόδου για προσθήκη ή τροποποίηση ηχητικών δεδομένων. Αυτή η συσκευή θα μπορούσε να είναι είτε το ποντική του υπολογιστή μέχρι κάτι πιο περίπλοκο όπως ένα midi controller. Ο H/Y λειτουργεί σαν υποδοχέας για τη κάρτα ήχου, το πρόγραμμα επεξεργασίας και παρέχει την επεξεργαστική ισχύ για όλο το σύστημα. Η κάρτα ήχου εξυπηρετεί σαν AD/DA μετατροπέας(αναλογικό σε ψηφιακό και αντίστροφα) και βοηθάει σε επιπρόσθετα κομμάτια της επεξεργασίας. Το λογισμικό ελέγχει όλο το hardware και παρέχει μια διεπαφή χρήστη για να καταστεί δυνατή η καταγραφή, επεξεργασία και αναπαραγωγή. Τα περισσότερα DAWs έχουν εκτεταμένη MIDI καταγραφή, επεξεργασία και δυνατότητες αναπαραγωγής.

Τα βασικά στοιχεία που περιέχει ένα ολοκληρωμένο DAW λειτουργικό είναι: κονσόλα μίξης, επιφάνεια ελέγχου, μετατροπέα ήχου. Τα εργαλεία αυτά διαφοροποιούνται ανάλογα με τη κατηγορία στην οποία ανήκει το κάθε DAW. Έτσι κάθε λογισμικό έχει διαφορετικές δυνατότητες ανάλογα με πιο στάδιο της ψηφιακής επεξεργασίας εξυπηρετεί .Τα βασικά στάδια στην ηχογράφηση και παραγωγή του ήχου είναι τα εξής: ηχογράφηση,editing(επεξεργασία),mixing(μίξη) και mastering.

Η εικονική αναπαράσταση των ηχητικών δεδομένων σε λογισμικά τύπου DAW είναι το κύριο εργαλείο στην επεξεργασία του ήχου. Συνήθως επιτρέπουν στο χρήστη να χειρίζεται τα διαφορετικά χαρακτηριστικά του ήχου οπτικά(μέσω της εικονικής αναπαράστασης). Κάθε πρόγραμμα έχει διαφορετική προσέγγιση όσον αφορά την επεξεργασία του ήχου μέσω της εικονικής αναπαράστασης αλλά γενικότερα έχουμε κάποιες έννοιες που είναι κοινές για όλα όπως threshold, envelope, transform, gain κ.α.

Τα πιο γνωστά DAW λογισμικά είναι τα : Cubase, Propellerhead Reason, FL Studio, Logic Pro και Pro tools.

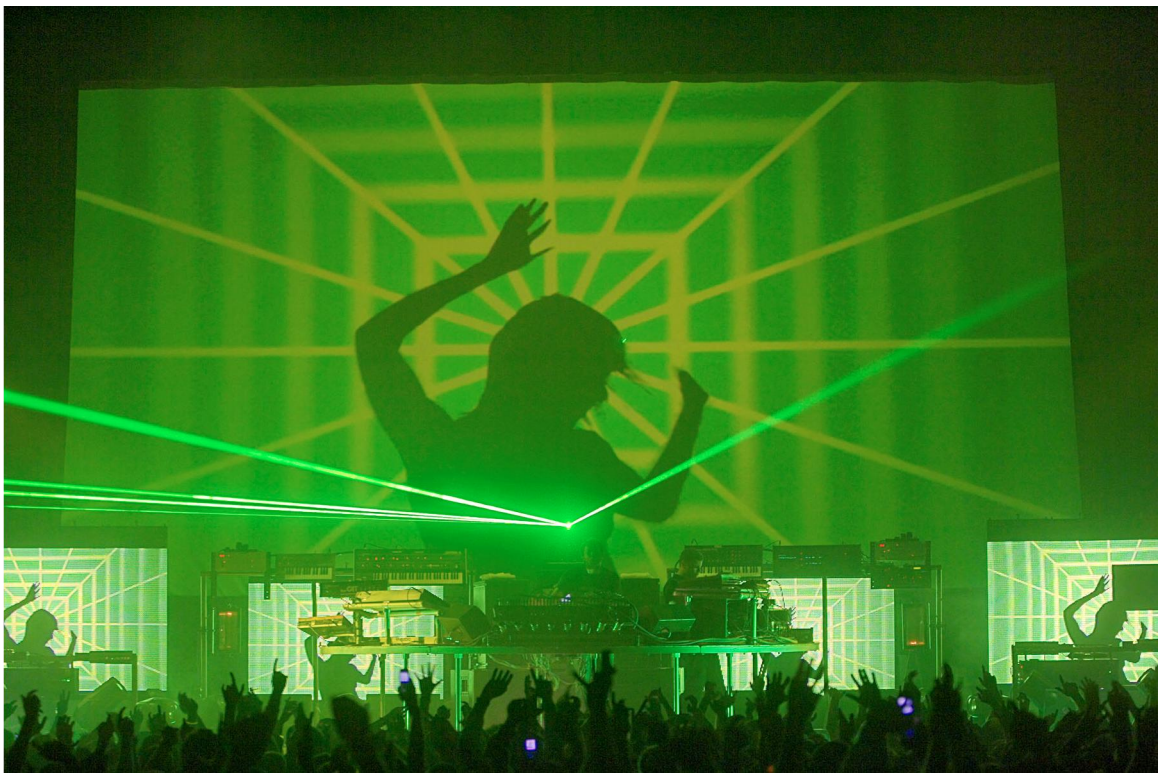
20 .Αναλυτής φάσματος σε plugin για το πρόγραμμα Blue Cat Audio

Media players

Με τον όρο media player χαρακτηρίζουμε τα προγράμματα (winamp, windows media player, itunes κ.α.) που χρησιμοποιούμε για να αναπαράγουμε δεδομένα ήχου και εικόνας(πολυμέσα).Κάποια από αυτά εστιάζουν περισσότερο στην αναπαραγωγή ηχητικών δεδομένων ενώ άλλα στην αναπαραγωγή βίντεο. Αρκετά από αυτά υποστηρίζουν την απεικόνιση ήχου και χρησιμοποιούν διαφορετικούς αλγόριθμους που βασίζονται στις τεχνικές όπως: beat detection algorithms,sound energy detection ,σε πραγματικό χρόνο. Πέρα από τα visuals που συνοδεύουν το κάθε πρόγραμμα, τα τελευταία χρόνια αρκετές εταιρίες ασχολούνται με τη δημιουργία καινούργιων, βασισμένα πάνω σε νεότερες τεχνικές και εφαρμόζονται στα players με τη μορφή plugin.

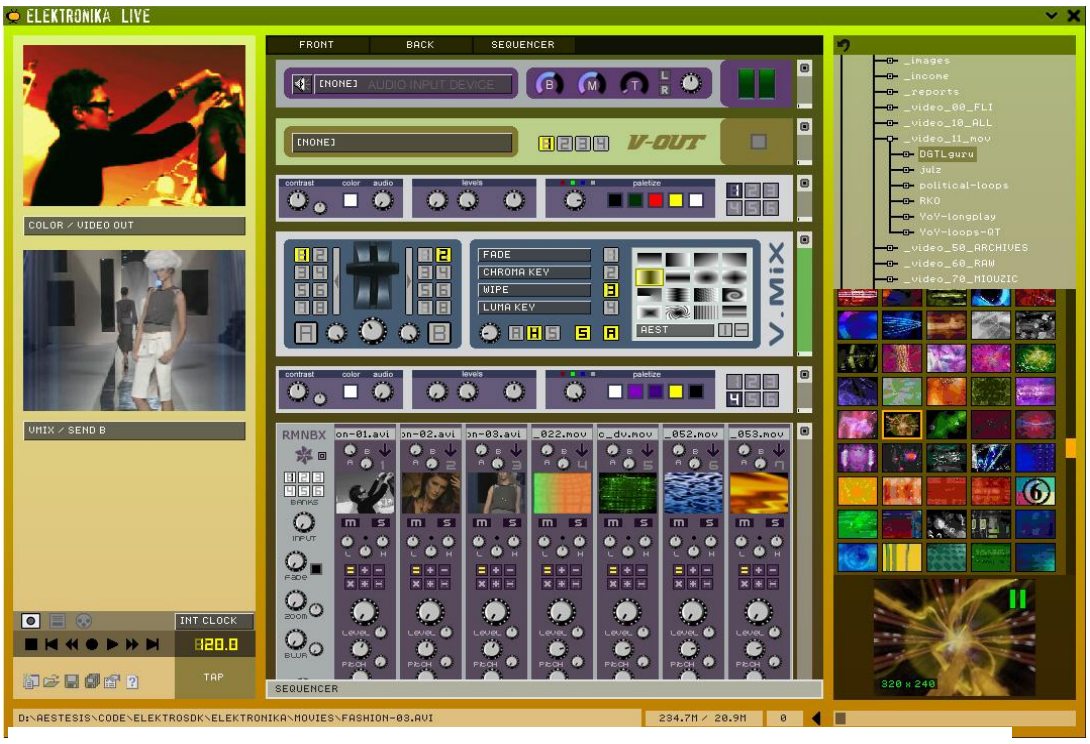
Vjing

Vjing είναι μία μορφή καλλιτεχνικής απόδοσης που συσχετίζεται με τη μουσική και την απεικόνιση του ήχου. Χαρακτηριστικά της VJing είναι η δημιουργία ή η χειραγώγηση της εικόνας σε πραγματικό χρόνο μέσω της τεχνολογικής διαμεσολάβησης και για ένα ακροατήριο, σε συγχρονισμό με τη μουσική. Προέρχεται από τις λέξεις video jockey (αντίστοιχο του disc jockey,Dj). VJing γίνεται συχνά σε εκδηλώσεις όπως συναυλίες, φεστιβάλ μουσικής και μερικές φορές σε συνδυασμό με άλλες παραστατικές τέχνες. Αυτό οδηγεί σε μια ζωντανή παράσταση που μπορεί να περιλαμβάνει μουσική, ηθοποιούς και χορευτές.



21. Live Performance

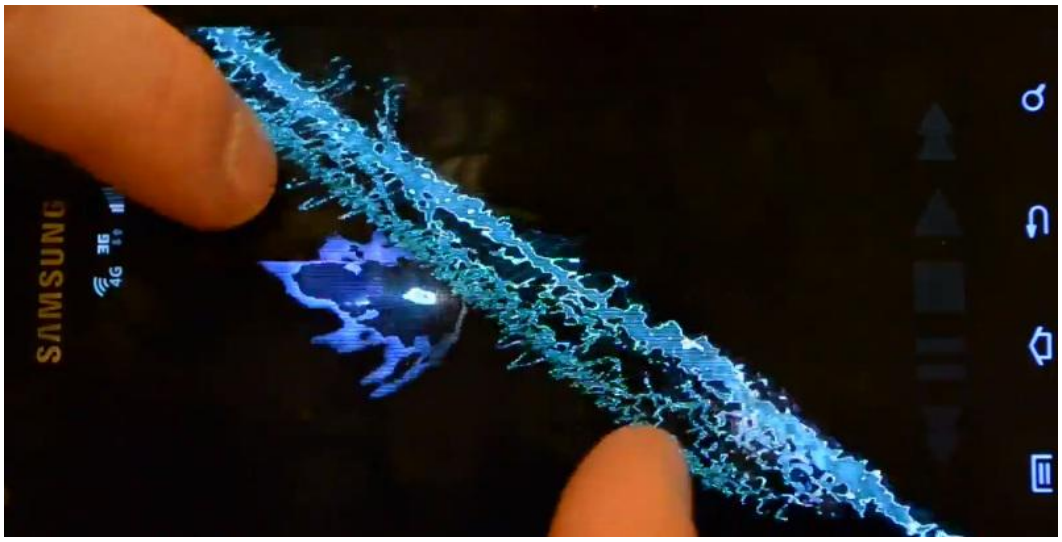
Το Vjing αναπτύχθηκε αρχικά χρησιμοποιώντας υλικό τις βιντεοκάμερες, συσκευές βίντεο για τη μετάδοση αυτοσχέδιων παραστάσεων από ζωντανή σύλληψη σε συνδυασμό με προηχογραφημένα και μιξαρισμένα στοιχεία από μεταδόσεις τις τηλεόρασης. Η διαδικασία αυτή του "μιξαρίσματος" οπτικού υλικού σε πραγματικό χρόνο συνεχίζεται και εξελίσσεται με τις καθώς νέο υλικά και λογισμικά(είτε στο εμπόριο είτε αυτοσχέδια) αναπτύσσονται από Vjs για αυτό τον σκοπό και μόνο. Αρκετά λογισμικά παράγονται από μικρές εταιρίες αφιερωμένα σε αυτό το τομέα τις VJAMM,RESOLUME,AESTESIS ELECTRONIKA(freeware) κ.α



22. Vj software interface

Smartphones and tablet applications

Με την εξέλιξη της επεξεργαστικής δύναμης των φορητών συσκευών όπως tablets και smartphones ξεκίνησαν να δημιουργούνται και στο χώρο αυτό αρκετές εφαρμογές τύπου visualizer σε λειτουργικά Android όπως και iOS. Αρκετές από αυτές υποστηρίζουν την χειραγώγηση των visuals μέσω touch , δίνοντας έτσι μία πιο ενδιαφέρουσα προσέγγιση στη δημιουργία εφέ μέσω του ήχου. Όπως κατηγοριοποιήσαμε στο παραπάνω κεφάλαιο τις εφαρμογές έτσι και εδώ έχουν αναπτυχθεί εφαρμογές που είτε εξυπηρετούν σαν εργαλεία πάνω στη ψηφιακή αναπαράσταση του ήχου, είτε σαν αφηρημένες καλλιτεχνικές δημιουργίες μέσω αυτού. Σε Android και iOS λειτουργικά μέσω των app stores τους έχουν οι χρήστες τη δυνατότητα να κατεβάσουν τέτοιες εφαρμογές και αρκετές διατίθενται με μηδενικό κόστος όπως : Audio Glow Music Visualizer, Music Visualizer και Visualisator 5000.



23. Η εφαρμογή ProjectM Music Visualizer

Η εφαρμογή ProjectM Music Visualizer υποστηρίζει multi touch επιτρέποντας στο χρήστη να δημιουργεί τα δικά του visuals(πάνω).

Όσον αφορά τώρα τις εφαρμογές – εργαλεία έχουν κυκλοφορήσει μία πληθώρα εφαρμογών που απευθύνονται σε επαγγελματίες ηχολήπτες, ερευνητές ή ακόμα και απλούς χρήστες. Τέτοιες εφαρμογές μπορούν να είναι είτε προσομοιωτές υλικών όπως ο παλμογράφος ή ισοσταθμιστής (equalizer) είτε αναλυτές πραγματικού χρόνου(real time analyzers –RTA) για μετασχηματισμό Fourrier ή αναπαράσταση κυματομορφής.

24. Εφαρμογή XA1, αναλυτής φάσματος πραγματικού χρόνου

4.Πειραματικό μέρος εφαρμογής

4.1 Ανάλυση εφαρμογής

Η αρχική ιδέα για τη πλατφόρμα είναι η δημιουργία τις διαδραστικού media player τύπου winamp που θα λειτουργεί πλήρως με τα gestures(χειρονομίες) του kinect. Επιπρόσθετα θα δημιουργεί οπτικό υλικό (visuals) που θα αλληλεπιδρά με το μουσικό κομμάτι χρησιμοποιώντας τεχνικές πάνω τις αλγόριθμους εύρεσης ρυθμού(beat detection algorithms) και mapping. Παρακάτω φαίνονται τα βήματα για την ολοκλήρωση τις εφαρμογής

- Ανάλυση, αξιολόγηση και επιλογή υλικού για την ανάπτυξη τις εφαρμογής
- Δημιουργία media player(κατανόηση εννοιών ψηφιακού ήχου τις sample rate,buffer size)
- Δημιουργία οπτικού υλικού
- Αξιολόγηση επίδοσης εφαρμογής πάνω στη αλληλεπίδραση ήχου-οπτικού υλικού σε πραγματικό χρόνο
- Εισαγωγή kinect (κατανόηση, ανάλυση depth image και point cloud)
- Εισαγωγή skeleton track(κατανόηση, εφαρμογή)
- Δημιουργία των modes(mapping hand coordinates)
- Δημιουργία inputs
- Τελικό στάδιο εφαρμογής χειρισμός των inputs μέσω gestures

4.2 Επιλογή υλικού ανάπτυξης εφαρμογής

Αρχικά η επιλογή τις processing έγινε διότι προσφέρει το ιδανικό περιβάλλον για την δημιουργία εφαρμογών πάνω στον οπτικό σχεδιασμό στην παραγωγή, σχεδιασμό γραφικών, προσφέρει ευκολία στη χρήση και είναι ευρέως διαδεδομένη ανάμεσα σε προγραμματιστές και σχεδιαστές για τη ανάπτυξη διαδραστικών πολυμεσικών εφαρμογών.

Πέρα από τα παραπάνω τις παρέχει αρκετές βιβλιοθήκες που τις επιτρέπουν να χειριστούμε τον ήχο αρκετά εύκολα(παρακάτω αναφέρονται οι τρεις πιο βασικές).

Τέλος η βιβλιοθήκη minim επιλέχτηκε για την υλοποίηση τις εφαρμογής και το kinect της Microsoft που επιτρέπει τη διαδραστικότητα με το χρήστη.

Το kinect προστέθηκε στην εφαρμογή σαν μία εναλλακτική μέθοδο για να έρχεται ο χρήστης σε επαφή με την εφαρμογή καθώς με την εισαγωγή του τις προσφέρει μία διαφορετική προσέγγιση όσον αφορά τα inputs.

4.3 Ανάλυση υλικού ανάπτυξης εφαρμογής

4.3.1 Παρουσίαση Microsoft's Kinect

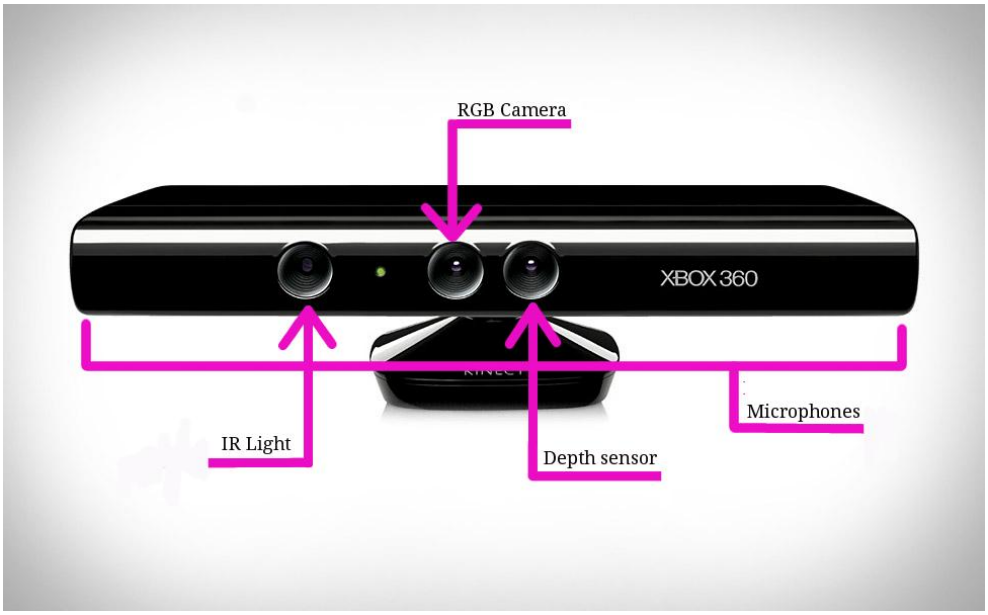
Το kinect είναι μία συσκευή ανίχνευσης κίνησης και χρησιμοποιείται από τη παιχνιδομηχανή Xbox 360 της Microsoft και από ηλεκτρονικούς υπολογιστές.

Κυκλοφόρησε στην αγορά γύρω στα τέλη του 2010 και προοριζόταν για να συνδέεται με το xbox 360 και να προσφέρει μία εναλλακτική εμπειρία στον χρήστη όσον αφορά τη διαδραστικότητα με τη παιχνιδομηχανή. Ήδη από τις πρώτες μέρες κυκλοφορίας του τράβηξε την προσοχή αρκετών hacker που γρήγορα ανέπτυξαν έναν ανοιχτό κώδικα για το χειρισμό του kinect στα linux. Αυτή η αρχική προσπάθεια οργανώθηκε από την ομάδα OpenKinect. Ο βασικός κώδικας γράφτηκε από Hector Martin, και κυκλοφόρησε ως «libfreenect''. Στη συνέχεια η ομάδα που δημιούργησε το υλικό του kinect (Primesense), κυκλοφορεί το OpenNI/NITE(toolkit για το kinect).Εν τέλει η Microsoft κυκλοφορεί το δικό τις API για το kinect μαζί με προσαρμοσμένο κώδικα ανίχνευσης σκελετού γύρω στα μέσα του 2011.

25. Depth sensors

Τα χαρακτηριστικά του kinect

- 1 αισθητήρα βάθους (depth sensor) από 0.8 – 3.5 μέτρα
- 1 RGB camera
- 4 μικρόφωνα
- λειτουργεί με εξωτερική τροφοδοσία



26. Microsoft's kinect characteristics

Πως λειτουργεί το kinect;

Το kinect μέσω του IR (infrared) προβολέα,προβάλει σημεία(points) στο χώρο.Ο αισθητήρας βάθους(Depth sensor) χρησιμοποιεί τη θέση του κάθε σημείου για να διατυπώσει μία εικόνα βάθους. Η θέση των σημείων που προβάλλονται αποθηκεύονται μέσα σε πίνακα που είναι ουσιαστικά τα δεδομένα τις. Οι έξοδοι που τις παράγει το kinect και με τις οποίες εμείς θα συνδέσουμε τη πλατφόρμα τις είναι τρεις:

1. εικόνα από την RGB κάμερα
2. εικόνα βάθους από τον αισθητήρα
3. ήχο από τα μικρόφωνα

Οι πληροφορίες που λαμβάνουμε από το kinect μπορούν να αξιοποιηθούν μέσω κάποιων βιβλιοθηκών τις Libfreenect, Microsoft kinect sdk και OpenNI/NITE (για την εφαρμογή χρησιμοποιήθηκε το OpenNI/NITE).Κάθε βιβλιοθήκη αναλόγως την εφαρμογή έχει τα δικά τις χαρακτηριστικά και υποστηρίζει διαφορετικές γλώσσες.

DepthImage(Εικόνα βάθους)

Το kinect είναι μία κάμερα "βάθους".Οι συνηθισμένες κάμερες συλλέγουν το φως που προβάλλεται πάνω στα αντικείμενα και δημιουργούν εικόνα. Το kinect από την άλλη καταγράφει το βάθος των αντικειμένων που βρίσκονται στην εμβέλειά του. Με τη χρήση υπέρυθρων και τις κάμερας δημιουργείται μία "εικόνα" που περιέχει πληροφορίες για την απόσταση των σημείων.

27. Depth image

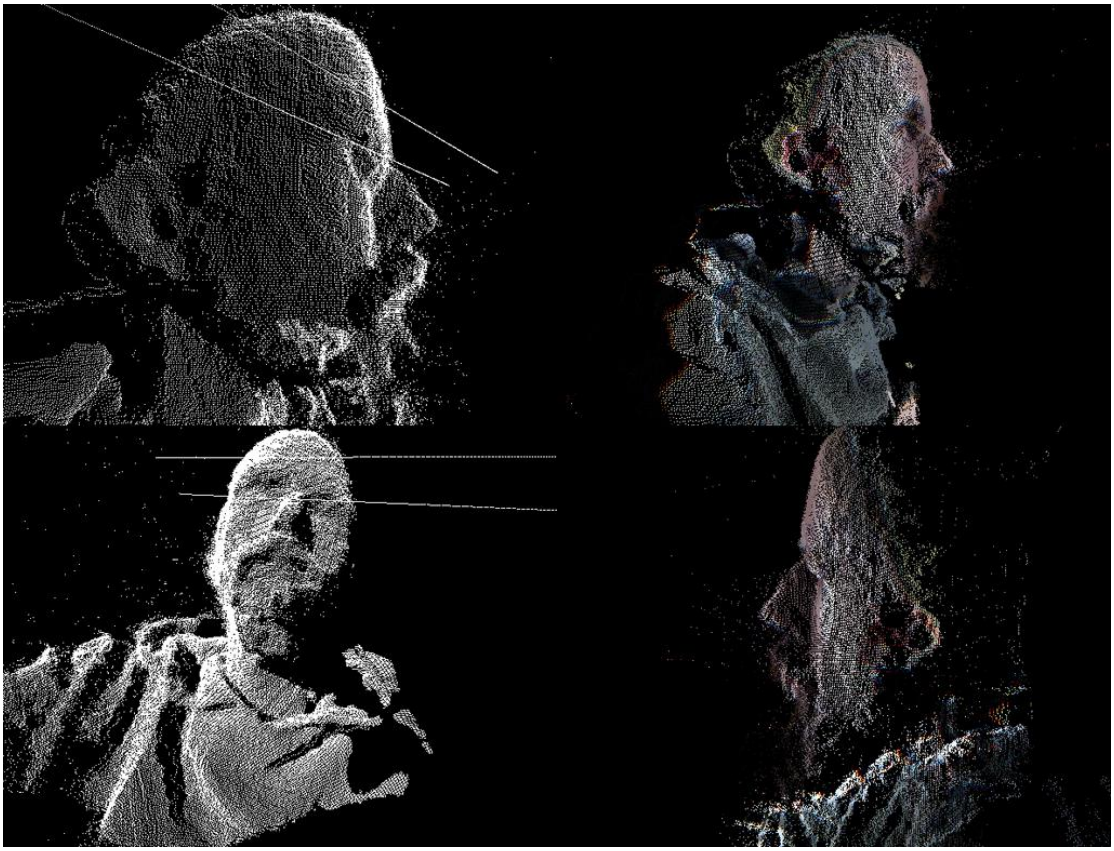
Point cloud

Παραπάνω παρατηρούμε πως χρησιμοποιούμε το kinect για να εξάγουμε δεδομένα σύμφωνα με την απόσταση του κάθε αντικειμένου για μία δισδιάστατη εικόνα. Άρα και οι εφαρμογές τις λειτουργούν μέσα σε δισδιάστατο χώρο.

Point cloud είναι μία τεχνική που χρησιμοποιείται ευρέως για να αναπαραστήσουμε το χώρο και τα αντικείμενα τις σε 3 διαστάσεις x, y και z. Έτσι αντί να έχουμε μία επίπεδη εικόνα τις είχαμε με τα δεδομένα που παίρνουμε από την εικόνα βάθους τώρα παίρνουμε δεδομένα από σύνολο σημείων μέσα στο χώρο που το κάθε ένα σημείο από αυτά έχει τις δικές του x,y,z συντεταγμένες. Μόλις καταφέρουμε να έχουμε σημεία σε τρισδιάστατο χώρο η αναπαράσταση αυτών ουσιαστικά παράγει τρισδιάστατα μοντέλα.

Με αυτό τις μπορούμε να παρατηρούμε την εικόνα από διαφορετικές οπτικές γωνίες και συνθήκες φωτισμού και να εφαρμόζουμε χρώματα και υφή πάνω στην επιφάνεια κάθε αντικειμένου με εκπληκτικά αποτελέσματα. (Borenstein)[7]

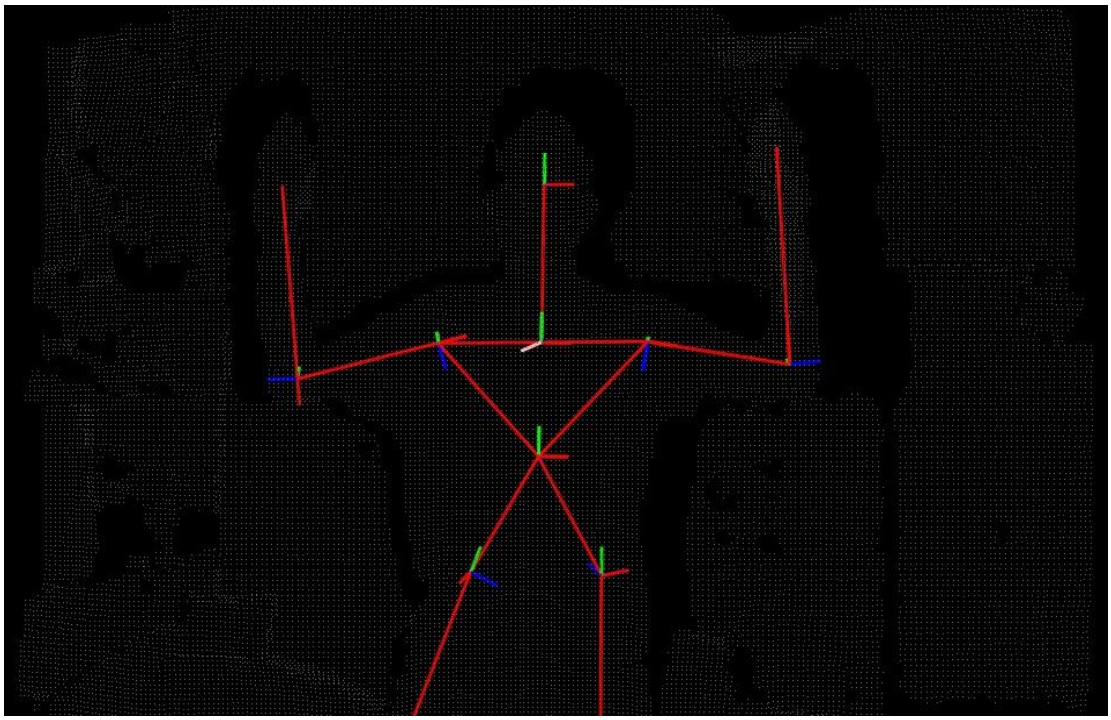
[7] Making Things See, Greg Borenstein



28. Διαφορετικές γωνίες από εικόνα point cloud

Skeleton Tracking

Μια από τις πιο χρήσιμες λειτουργίες του kinect και μάλλον πιο διαδεδομένη στην δημιουργία διαδραστικών εφαρμογών είναι η ανίχνευση του χρήστη(skeleton track).



29.Skeleton tracking process

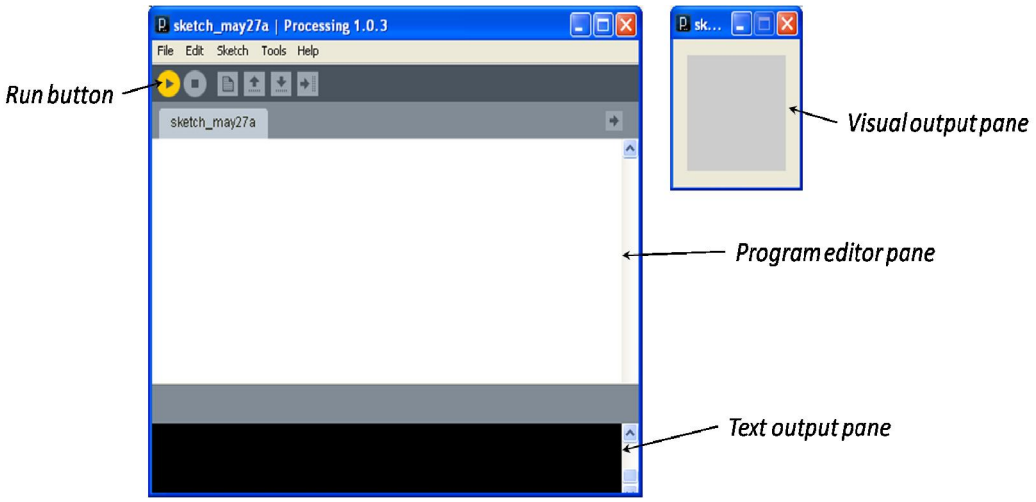
Η OpenNI τις παρέχει αυτή τη δυνατότητα αρκετά εύκολα χωρίς να περνάμε από τη διαδικασία ανίχνευσης του κάθε pixel μέσα στο χώρο. Η ανίχνευση σκελετού(skeleton track) εντοπίζει τον χρήστη και μετέπειτα περνά από μία διαδικασία αντιστοίχησης του κάθε μέλους του χρήστη(κεφάλι, χέρια, πόδια, γόνατα κ.α.).Καταλαβαίνουμε λοιπόν ότι αντί εφαρμογή τις να περνάει από επαναλήψεις για να ανιχνεύει το κάθε pixel μέσα στο χώρο, παίρνει απευθείας δεδομένα για την ακριβή θέση του κάθε μέλους του σώματός. Περισσότερα πάνω στην ανίχνευση χρήστη αναφέρονται στα παρακάτω κεφάλαια.

4.3.2 Παρουσίαση περιβάλλοντος και γλώσσας Processing

Processing είναι μία open source γλώσσα προγραμματισμού και ένα περιβάλλον που απευθύνεται σε καλλιτέχνες, σχεδιαστές και προγραμματιστές και χρησιμοποιείται για την δημιουργία πολυμεσικών εφαρμογών και animation στο χώρο τις ψηφιακής τέχνης και των νέων μέσων τέχνης (new media art).Αρχικά είχε σαν σκοπό να διδάξει τις θεμελιώδεις έννοιες του προγραμματισμού σε ένα οπτικό περιεχόμενο. Η processing τις και το περιβάλλον τις έχει αναπτυχθεί πάνω σε java. Παρόλο που η γλώσσα αναπτύχθηκε στη Java, το συντακτικό τις είναι απλουστευμένο και το προγραμματιστικό τις μοντέλο βασίζεται στα γραφικά. Πέρα από τη γλώσσα έχουμε το κύριο API τις processing το οποίο απαρτίζεται από αρκετές βιβλιοθήκες που το κάνουν πιο ολοκληρωμένο.[8]

Το περιβάλλον τις processing

Σε σχέση με την java είναι μία γλώσσα η οποία δεν απαιτεί γνώσεις πάνω σε αντικειμενοστραφής έννοιες τις κλάσεις, υποκλάσεις, αντικείμενα αλλά επιτρέπει σε χρήστες πρόσβαση σε αυτές. Η processing παρέχει αρκετές βιβλιοθήκες(κάποιες από αυτές συνοδεύουν το πρόγραμμα) που διευκολύνουν αρκετά τις χρήστες. Αυτές οι βιβλιοθήκες είναι προγραμματισμένες σε java αλλά ολοκληρωμένες σε processing. Το PDE(processing development environment) είναι το κύριο περιβάλλον τις processing και απεικονίζεται στο παρακάτω στιγμιότυπο. (Fry & Reas)



30. Processing interface

[8] Getting Started with Processing, Casey Reas & Ben Fry

Αρχικά η εισαγωγή οποιασδήποτε βιβλιοθήκης γίνεται με την import. Ότι βιβλιοθήκες έχουμε σκοπό να χρησιμοποιήσουμε τις εισάγουμε στην αρχή του προγράμματος.

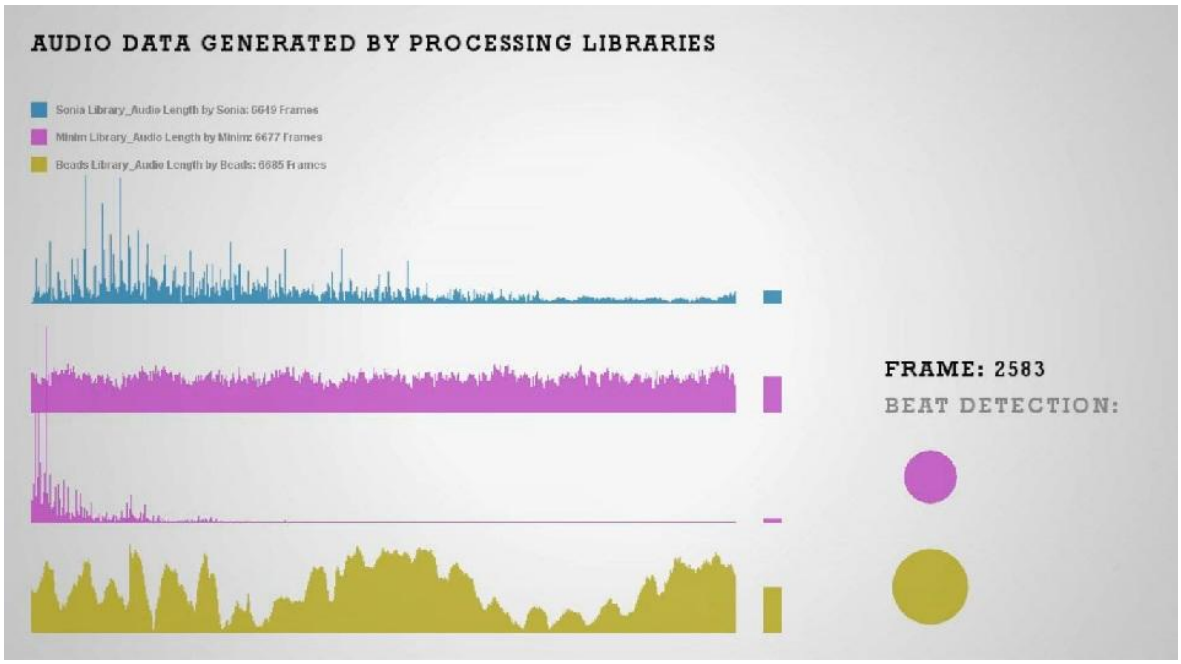
Ένα κύριο στοιχείο της processing στις συναρτήσεις της που κατά μία έννοια είναι ευδιάκριτη η λειτουργία τους με τον τρόπο που έχουν ονομαστεί(πχ background(); Αλλάζει το φόντο στο περιβάλλον τις, ellipse(); Δημιουργεί μία έλλειψη , fill(); Και noFill(); Χρωματίζει το εσωτερικό τις αντικείμενου κ.α.). Οι πιο σημαντικές λειτουργίες της processing που αποτελούν το βασικό σκελετό του προγράμματος της είναι η setup() και η draw().

Στη **setup()** δηλώνουμε συνήθως μεταβλητές(αν και οι μεταβλητές που χρησιμοποιούμε σε όλο το πρόγραμμα που θέλουμε να είναι "ορατές" για τις κλάσεις δηλώνονται έξω από τη setup),ορίζουμε το μέγεθος του παραθύρου της με την size() και γενικότερα ρυθμίζουμε αντικείμενα τα οποία κατά την εκτέλεση τις εφαρμογής της παραμένουν σταθερά. Η **draw()** κατά το τρέξιμο της εφαρμογής από προεπιλογή τρέχει με 60 frames το δευτερόλεπτο(άμα θέλουμε να χαμηλότερο framerate αυτό ορίζεται στη setup()).

Στη draw() λοιπόν βρίσκονται οι κυρίες εντολές του προγράμματος ,δηλαδή τι θέλουμε να δημιουργήσουμε πάνω σε αυτά που ορίσαμε μέσα στη setup(). Μεταβλητές που κατά την εκτέλεση της εφαρμογής δεν μένουν σταθερές και λειτουργίες που παίρνουν είσοδο από το χρήστη και γενικότερα αντικείμενα που παραμετροποιούνται βρίσκονται μέσα στη draw(). Κάθε λοιπόν εφαρμογή θα πρέπει να περιέχει αυτές τις συναρτήσεις. Πέρα από αυτές έχουμε τις συναρτήσεις και αντικείμενα που παρέχονται από τις βιβλιοθήκες που χρησιμοποιούμε.

Processing sound libraries

- minim audio library
- SONIA
- beads



31. Processing Sound libraries

Για τη επεξεργασία και το χειρισμό δεδομένων ήχου έχουν αναπτυχθεί για τη processing αρκετές βιβλιοθήκες με διαφορετικές λειτουργίες η καθεμία. Οι πιο διαδεδομένες είναι η minim, beads και SONIA.

Στο παραπάνω instance συγκρίνουμε αναπαράσταση κυματομορφής ,φάσμα συχνοτήτων και αντικείμενα για ανίχνευση beat για καθεμιά. Στη τελική εφαρμογή χρησιμοποιήθηκε η minim βιβλιοθήκη σαν πιο διαδεδομένη και μας επιτρέπει αρκετά εύκολα να πραγματοποιήσουμε μετασχηματισμό Fourier και μας παρέχει ήδη αρκετά εργαλεία σε σχέση με τις υπόλοιπες βιβλιοθήκες.

Minim sound library

Η minim είναι βιβλιοθήκη ήχου που συνοδεύει το IDE της processing. Αρχικά παρείχε την δυνατότητα σε χρήστες να ενσωματώνουν ηχητικά κομμάτια στα σκίτσα τους και μετέπειτα υλοπιεί εργαλεία για τη ψηφιακή ανάλυση-επεξεργασία του ήχου.[9] Η minim χρησιμοποιεί το JavaSound API παρέχοντας έτσι σε συνδυασμό με το περιβάλλον της processing σε χρήστες που δεν είναι αρκετά εξοικειωμένοι με προγραμματισμό εργαλεία για τον χειρισμό του ήχου. Πέρα από αυτό είναι αρκετά ευέλικτη για πιο έμπυρους χρήστες. (Fede)

Βασικές λειτουργίες της είναι:

- αναπαραγωγή μουσικής
- ηχογράφηση μέσω line in-mic input
- δημιουργία οπτικών αναπαραστάσεων ψηφιακών σημάτων(κυματομορφή, φάσμα συχνοτήτων)
- χρήση φίλτρων για τη ψηφιακή επεξεργασία κομματιών
- Beat detection

4.4 Σύγκριση Αποδοτικότητας Αλγορίθμων εύρεσης ρυθμού

Στο τρίτο κεφάλαιο κάναμε μία έρευνα πάνω τις δύο βασικές κατηγορίες αλγορίθμων εύρεσης ρυθμού και αναλύσαμε το σκεπτικό που κρύβεται πίσω από τις λειτουργίες του καθενός. Σε αυτό το κεφάλαιο θα δοκιμάσουμε την αποδοτικότητα τους σε διαφορετικά είδη μουσικής για να μπορέσουμε να καταλήξουμε στον ιδανικότερο για την εφαρμογή μας. Τα είδη που επιλέξαμε να δουλέψουμε ανήκουν στη σύγχρονη μουσική πράγμα που σημαίνει ότι δεν εφαρμόστηκαν οι αλγόριθμοι σε είδη όπως κλασική, jazz, blues και κλασσικής ροκ. Έτσι οι δύο βασικές κατηγορίες πάνω στους αλγόριθμους εύρεσης ρυθμού είναι:

- Αλγόριθμος εύρεσης ρυθμού μέσω τις ενέργειας του ήχου
- Αλγόριθμος εύρεσης ρυθμού μέσω ζώνες συχνοτήτων

Μέσω της βιβλιοθήκης minim οι αλγόριθμοι αυτοί έχουν υλοποιηθεί στη processing και παρουσιάζονται αναλυτικά μαζί με το κώδικα τις κάθε τις στον εξής ιστότοπο: http://code.compartmental.net/tools/minim/manual-beatdetect/

[9] http://code.compartmental.net/

Πριν ξεκινήσουμε την ανάλυση αυτή θα πούμε κάποια λόγια για το πώς η minim τις υλοποιεί και θα εξηγήσουμε ένα κομμάτι του κώδικα για να μπορέσει η παρακάτω διαδικασία να γίνει πιο κατανοητή.

Αρχικά στη setup() έχουμε τη δημιουργία του αντικειμένου beatDetect() με διαφορετικό constructor για το Energy mode και διαφορετικό για το Frequency mode.Στο energy mode δημιουργείται το αντικείμενο beat χωρίς ορίσματα ενώ στο frequency δημιουργείται με δύο ορίσματα, μέγεθος buffer κομματιού και ρυθμός δειγματοληψίας. Τέλος στη setup() καθορίζουμε και τη συχνότητα με την οποία θα ελέγχουμε για beat με τη μέθοδο setSensitivity() όπου δέχεται ένα ακέραιο σε ms(δηλ. για 1000ms θα αναζητάει για beat κάθε ένα δευτερόλεπτο). Όταν το κομμάτι έχει υψηλά Bpm συνήθως ορίζουμε την setSensitivity στα 300ms.

.

```
// Create a BeatDetect object that is in SOUND_ENERGY mode.
BeatDetect()
// Create a BeatDetect object that is in FREQ_ENERGY mode
// and expects a sample buffer with the requested attributes.
BeatDetect(int timeSize, float sampleRate)
// Analyze the samples in ab.
void detect(AudioBuffer ab)
// Constant used to request frequency energy tracking mode.
static int FREQ_ENERGY
// Constant used to request sound energy tracking mode.
static int SOUND_ENERY
// Set the detection mode to use
void detectMode(int algo)
// Sets the sensitivity of the algorithm (in milliseconds)
void setSensitivity(int s)
```

32. Η setup() και για τα δύο modes

Στη συνέχεια για το energy mode, μέσα στη draw() γίνεται η ανίχνευση μέσω της beat.detect() που σαν όρισμα παίρνει είτε song.mix(και για τα δύο κανάλια) είτε song.left ή song.right. Έτσι όταν πιάσει beat η μέθοδος beat.isOnset() επιστρέφει true. Τώρα στο frequency mode οι μέθοδοι που χρησιμοποιούμε είναι οι isKick(),isSnare(),isHat() και isRange() όπου και αυτές επιστρέφουν Boolean σε περίπτωση που πιάσουν beat. Παρακάτω θα εξηγήσουμε αυτές τις μεθόδους πιο αναλυτικά. Αυτά είναι τα κύρια αντικείμενα και οι μέθοδοι που υλοποιεί η minim για το κάθε ένα από τους αλγόριθμου. Πάμε τώρα να τους δούμε πιο αναλυτικά.

4.4.1 Αποδοτικότητα Αλγόριθμου μέσω τις ενέργειας του ήχου(energy mode)

Ο συγκεκριμένος αλγόριθμος βασίζεται πάνω στις διακυμάνσεις της ενέργεια(ή αλλιώς έντασης) του ήχου και λειτουργεί συγκρίνοντας τη **μέση διακύμανση της ηχητικής ενέργειας** με τη

στιγμιαία διακύμανση. Έχοντας σαν δεδομένα τα παραπάνω περάσαμε στη δοκιμή του σε είδη μουσικής με κύριες κατηγορίες την pop-rock και την ηλεκτρονική.

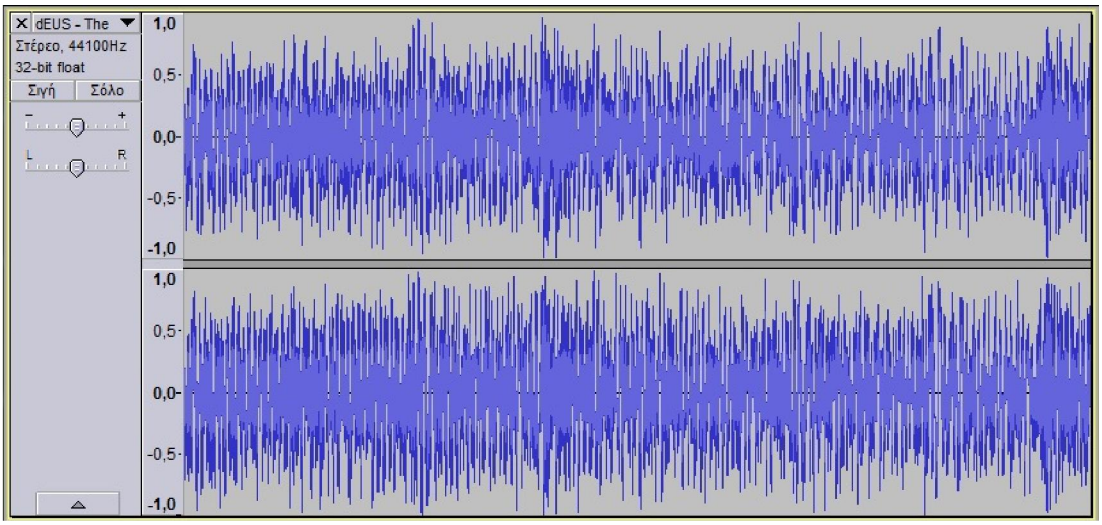
Pop-Rock

Ο αλγόριθμος αυτός δεν συστήνεται για είδη τα οποία ανήκουν στη κατηγορία pop-rock και γενικότερα κομμάτια τα οποία περιλαμβάνουν κιθάρες, ντραμς, μπάσο, πλήκτρα και φωνητικά. Σε αυτά τα είδη μουσικής πρωταγωνιστικό ρόλο έχουν συνήθως οι κιθάρες, φωνή και τα πλήκτρα αφήνοντας έτσι το ρυθμικό κομμάτι(μπάσο-ντραμς) να κυμαίνεται σε χαμηλότερες εντάσεις, δυσκολεύοντας έτσι τη λειτουργία του αλγορίθμου. Πέρα από τα παραπάνω σε σημεία όπου στο κομμάτι είναι αρκετά ευδιάκριτα τα ρυθμικά στοιχεία, ο αλγόριθμος γίνεται πιο αποδοτικός παρόλα αυτά η αποτελεσματικότητα του κυμαίνεται σε ποσοστά κάτω από 20%.

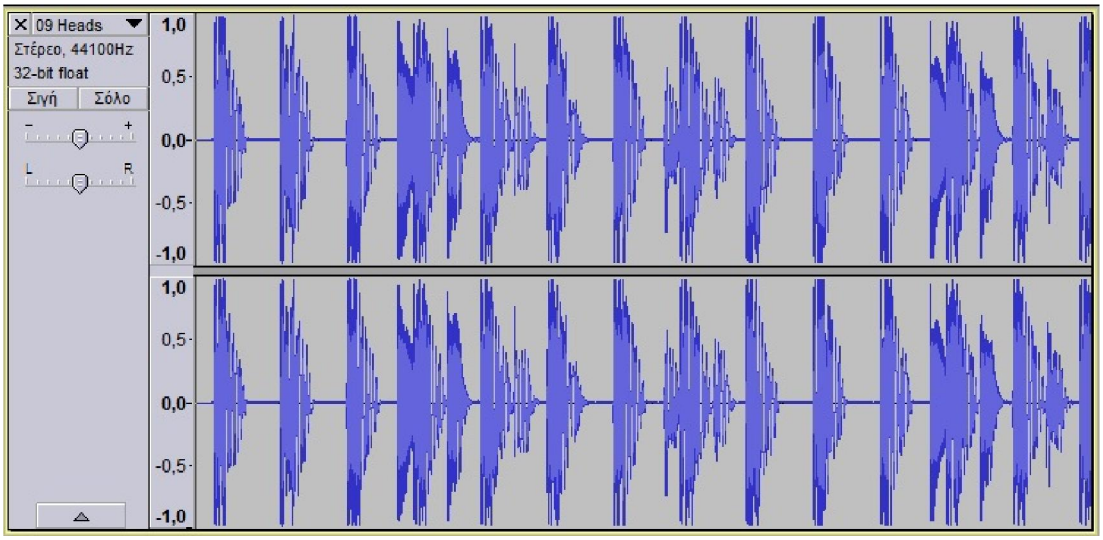
Ηλεκτρονική

Στο τομέα τώρα της ηλεκτρονικής κάναμε μία πιο λεπτομερή ανάλυση διότι διαπιστώθηκε ένα πιο ευρύ φάσμα στα ποσοστά της αποδοτικότητας του αλγόριθμου. Στη rock τα ποσοστά αποδοτικότητας ήταν μεταξύ 20% και 35% ενώ στην ηλεκτρονική κυμαίνονταν από 35% μέχρι 90% και αυτό οφείλεται στη ποικιλία που επικρατεί στα διαφορετικά είδη της ηλεκτρονικής μουσικής. Έτσι δοκιμάσαμε να τον αλγόριθμο στα εξής είδη: Techno, House, Drum and Bass, BreakBeat, Electro, Dub και Trance. Διαπιστώθηκε λοιπόν ότι ο αλγόριθμος λειτουργεί αποτελεσματικά, άσχετα από το κάθε κομμάτι αυτό κάθε αυτό, στην Techno, Trance και House με ποσοστά αποδοτικότητας συνήθως πάνω από 80%. Τα αποτελέσματα που πήραμε από αυτά τα είδη είναι δικαιολογημένα άμα σκεφτούμε ότι σε αυτά τα είδη, το beat αποτελεί το κύριο στοιχείο, πράγμα που το κάνει αρκετά ευδιάκριτο σε σχέση με τα υπόλοιπα στοιχεία όπως ,τις μπασογραμμές και τη μελωδία. Στα είδη που έχουμε έντονα μελωδικά στοιχεία και φωνή, όπως Electro,Electro pop, synth pop,ο αλγόριθμος έχει αποδοτικότητα γύρω στο 35% με 50 %. Οι εντάσεις του ρυθμικού μέρους μπορεί να πιο έντονες σε κάποια σημεία, αναλόγως το κομμάτι και εκεί ο αλγόριθμος γίνεται εμφανέστατα πιο αποδοτικός, αλλά όπως είπαμε όπου μπλέκουν διαφορετικά στοιχεία μαζί, δυσκολεύουν αρκετά την ανίχνευση του beat.

Για να κατανοήσουμε καλύτερα τα παραπάνω, ακολουθούν δύο στιγμιότυπα που λάβαμε μέσω του προγράμματος audacity(software για ηχογράφηση και επεξεργασία μουσικής). Στα στιγμιότυπα αυτά έχουμε δύο απεικονίσεις κυματομορφής(δηλαδή έντασης προς χρόνο) για δύο κομμάτια διαφορετικού είδους, για τα οποία ο αλγόριθμός είχε διαφορετικά αποτελέσματα.



33. Στιγμιότυπο κομματιού ροκ



34. Στιγμιότυπο κομματιού ηλεκτρονικής μουσικής

Οι δύο εικόνες παραπάνω απεικονίζουν στιγμιότυπα ενός ροκ και ενός ηλεκτρονικού κομματιού. Συγκρίνοντας τώρα τις διαφορετικές κυματομορφές, παρατηρούμε ότι και με το μάτι είναι αρκετά ευδιάκριτος ο ρυθμός στο ηλεκτρονικό, ενώ στο ροκ που περιέχει πιο πολλά στοιχεία δεν μπορεί να μας δώσει μία σαφή εικόνα για τον ρυθμό του κομματιού. Όπως είπαμε και παραπάνω ο αλγόριθμος λειτουργεί με τις διαφορετικές διακυμάνσεις στην ένταση του ήχου, έτσι γνωρίζοντας τον τρόπο λειτουργίας του αλγόριθμου, κατανοούμε σε τι μουσικά κομμάτια ο αλγόριθμος αυτός είναι αποδοτικός.

4.4.2 Αποδοτικότητα Αλγορίθμου μέσω ζώνες συχνοτήτων(frequency mode)

Έχοντας τώρα δοκιμάσει τον αλγόριθμο ενέργειας του ήχου, καταλαβαίνουμε σε τι μουσικά κομμάτια μπορούμε να τον εφαρμόσουμε με επιτυχία. Η δεύτερη τώρα κατηγορία λειτουργεί με διαφορετικό σκεπτικό, αφού σαν είσοδο δέχεται τις διαφορετικές διακυμάνσεις στην ένταση αλλά σε

συγκεκριμένες ζώνες συχνοτήτων. Όπως και στο παραπάνω αλγόριθμο, έτσι και αυτός δοκιμάστηκε στα ίδια είδη μουσικής.

Η ιδιαιτερότητα του αλγόριθμου αυτού είναι ότι μας επιτρέπει να επιλέξουμε από μόνοι μας τις ζώνες συχνοτήτων που μας ενδιαφέρουν και ανιχνεύει σε αυτές για beat. Ο αλγόριθμος αρχικά προσφέρει τρεις προεπιλεγμένες ζώνες, για ψιλές, μεσαίες και χαμηλές συχνότητες μέσω των isKick(), isSnare και isHat() συναρτήσεων αλλά όπως είπαμε η minim έχει και την isRange(int low,int high,int threshold) για να επιλέξουμε εμείς τις ζώνες που μας ενδιαφέρουν. Η isRange() δέχεται 3 ακέραιους κατά το κάλεσμα της οι πρώτοι 2 είναι από πια μέχρι πια ζώνη να πάρει(low μέχρι high) και ο τρίτος είναι το κατώφλι. Έτσι έστω για παράδειγμα ότι καλώ την isRange(5,20,7), που αυτό σημαίνει ότι από τις ζώνες 5 μέχρι 20 άμα τουλάχιστον 7 μου γυρίσουν τιμή true τότε η isRange επιστρέφει true.

Έτσι αφού καταλάβαμε πώς να δουλεύουμε με την isRange() περνάμε τώρα να δοκιμάσουμε τις προεπιλεγμένες συναρτήσεις που αναφέραμε.

Pop-rock

Ο αλγόριθμος σε γενικές γραμμές δουλεύει αρκετά καλά με διαφορετικά είδη της ροκ πιάνοντας τα μεσαία, ψιλά και τα χαμηλά επίσης αρκετά ικανοποιητικά. Σε κομμάτια που δε περιλαμβάνουν ντραμς κ ο ρυθμός μπορεί να πατάει είτε στο μπάσο είτε στη φωνή, ο αλγόριθμος αδυνατεί να ανιχνεύσει ρυθμό ακριβή ρυθμό. Τα προβλήματα που παρουσίαζε ο προηγούμενος αλγόριθμος στο pop-rock είδος εδώ δεν τα έχουμε επειδή επικεντρωνόμαστε στις ζώνες συχνοτήτων του κομματιού όπου αυτές μας δίνουν το beat και όχι η ένταση. Το πιο σημαντικό πρόβλημα που είχαμε με το προηγούμενο αλγόριθμο ήταν η αδυναμία του να ξεχωρίσει το ρυθμό, όταν είχαμε παράλληλα φωνητικά, κιθάρες ή άλλα στοιχεία. Με αυτή τη τεχνική ξεπεράσαμε αυτά τα προβλήματα κατά πολύ και βελτιώσαμε την αποδοτικότητα της εφαρμογής.

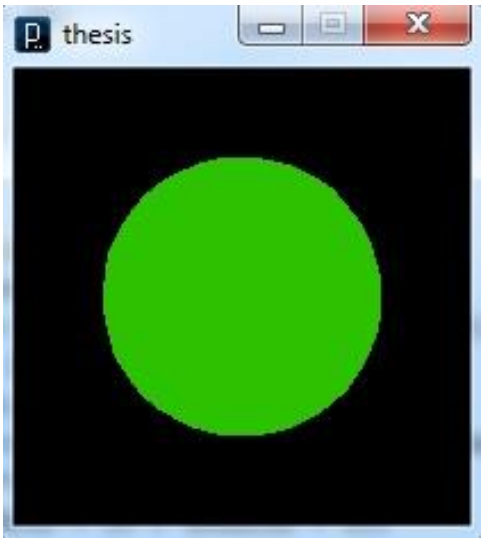
Η αποδοτικότητα που λαμβάναμε στις διαφορετικές κατηγορίες της ροκ και για τις τρεις ζώνες ήταν κατά μέσο όρο κοντά στο 70%.

Ηλεκτρονική

Στην ηλεκτρονική τώρα ο αλγόριθμος δοκιμάστηκε σε αρκετά είδη μουσική από αργό μέχρι γρήγορο tempo και τα αποτελέσματα που λάβαμε ήταν πλήρως ικανοποιητικά. Σε μουσικά είδη όπως breakbeat και drum n bass(γρήγορο tempo), είχαμε σωστή ανταπόκριση ιδιαίτερα με την isSnare() και την isKick(). Σε πιο αργό tempo όπως dub, trip hop είχαμε ακόμα καλύτερα ανταπόκριση και από τις τρεις ζώνες και στα είδη όπως electro,techno και house.

35. Στιγμιότυπο εφαρμογής minim στο frequency mode

Παρακάτω φαίνεται ένα από τα προεπιλεγμένα interface της minim για τον αλγόριθμο. Έχουμε από τα αριστερά προς τα δεξιά isKick(), isSnare(), isHat(), να αλλάζουν μεγέθη κάθε φορά που επιστρέφουν true(δηλ, πιάνουν beat.



36. Εφαρμογή minim στο energy mode

4.4.3 Συμπεράσματα

Σε αυτό το κεφάλαιο δοκιμάσαμε και τους δύο πιο βασικούς αλγόριθμους πάνω στην ανίχνευση ρυθμού. Οι υλοποιήσεις και τα παραδείγματα που παρουσιαστήκαν έγιναν στο περιβάλλον της processing μέσω της βιβλιοθήκης ήχου minim. Ο πρώτος αλγόριθμος που δοκιμάσαμε αφορούσε την **ενέργεια του ήχου** και πως αντιλαμβάνεται ο υπολογιστής το ρυθμό μέσω των διακυμάνσεων στην ένταση. Η μέθοδος αυτή πέρα την ευκολία στην υλοποίηση της, αποδείχθηκε ιδανική για μουσικά είδη με ευδιάκριτο ρυθμό, όπως στην ηλεκτρονική. Σε μουσικά είδη που περιλαμβάνουν πιο πολύπλοκα στοιχεία και ο ρυθμός κυμαίνεται σε χαμηλές εντάσεις, αδυνατούσε να πιάσει τα beat.

Ο δεύτερος αφορούσε την ενέργεια του ήχου αλλά σε συγκεκριμένες ζώνες συχνοτήτων. Περνούσε δηλαδή πρώτα από τη διαδικασία μετασχηματισμού σε συχνότητες και μετά, ανίχνευε την ένταση σε αυτές. Η μέθοδος αυτή μπορεί να ήταν πιο δύσκολη στην υλοποίηση αλλά τα αποτελέσματα που πήραμε και στην rock όπως και στην ηλεκτρονική ήταν πλήρως ικανοποιητικά με τις προεπιλεγμένες μεθόδους. Πέρα από τις προεπιλεγμένες μεθόδους έχουμε τη δυνατότητα να δημιουργήσουμε δικές μας, επιλέγοντας εμείς σε ποιες ζώνες θέλουμε να ανιχνεύουμε για beats, φέρνοντας έτσι την εφαρμογή ακριβώς στα μέτρα μας.

Για τη δημιουργία κάποιων στιγμιότυπων που αναπαριστούσαν τη κυματομορφή χρησιμοποιήσαμε το freeware πρόγραμμα επεξεργασίας ήχου **Audacity.**

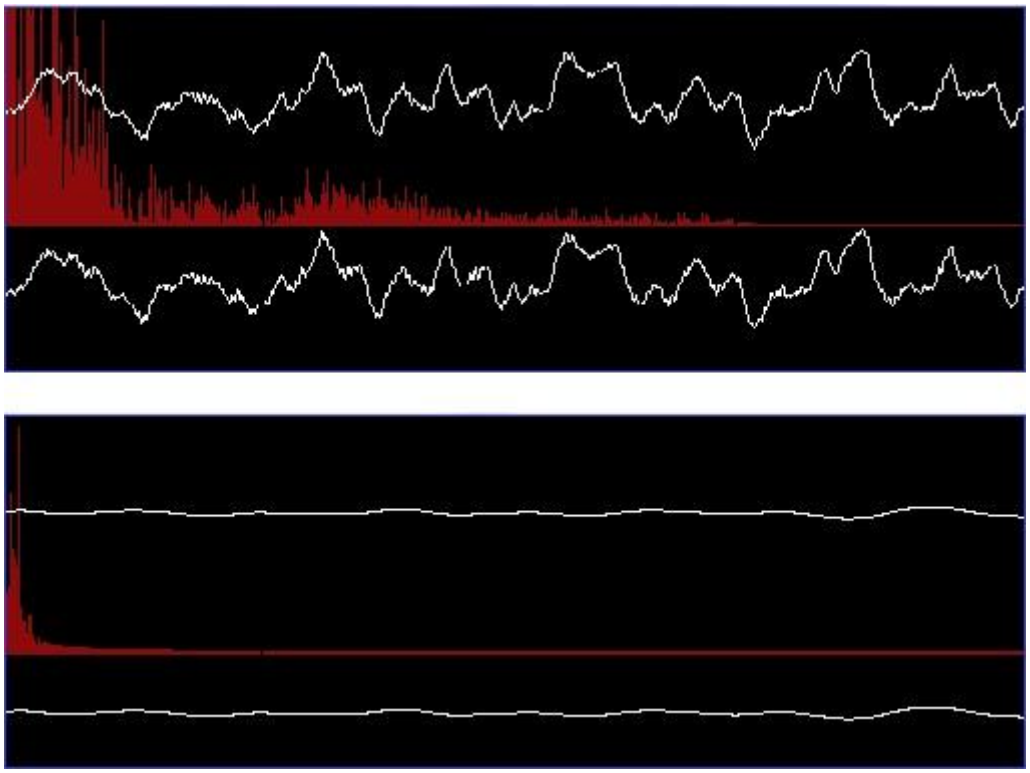
4.5 Ψηφιακά φίλτρα στη πράξη

Στα παραπάνω κεφάλαια εξηγήσαμε τη χρησιμότητα των ψηφιακών φίλτρων και αναφέραμε ενδεικτικά τις τέσσερις βασικές κατηγορίες τους(χαμηλοπερατά, υψιπερατά, ζωνοπερατά και ζωνοφρακτικά). Παρακάτω θα τα δούμε στη πράξη μέσω της processing, αφού η minim μας παρέχει έτοιμα εργαλεία και αντικείμενα για την υλοποίηση τους. Θα παρατηρήσουμε κυρίως τις διαφορετικές μεταβολές στην κυματομορφή καθώς εμείς θα αλλάζουμε τη συχνότητα αποκοπής σε κάθε ένα από αυτά. Επίσης θα παρατηρήσουμε το τρόπο και τις μεθόδους που χρησιμοποιεί η βιβλιοθήκη minim για την υλοποίηση των φίλτρων αυτών.

4.5.1 Χαμηλοπερατά / Υψιπερατά φίλτρα

Στο κεφάλαιο με τα ψηφιακά φίλτρα αναφέραμε ότι τα **χαμηλοπερατά** φίλτρα αφήνουν να περνάν απαραμώρφοτα τα σήματα με συχνότητες **χαμηλότερες** από τη συχνότητα αποκοπής ενώ τα **υψιπερατά** φίλτρα επιτρέπουν τη διέλευση σημάτων με συχνότητες **μεγαλύτερες** από τη συχνότητα αποκοπής.

Στη παρακάτω εικόνα βλέπουμε το χαμηλοπερατό φίλτρο στη πράξη.

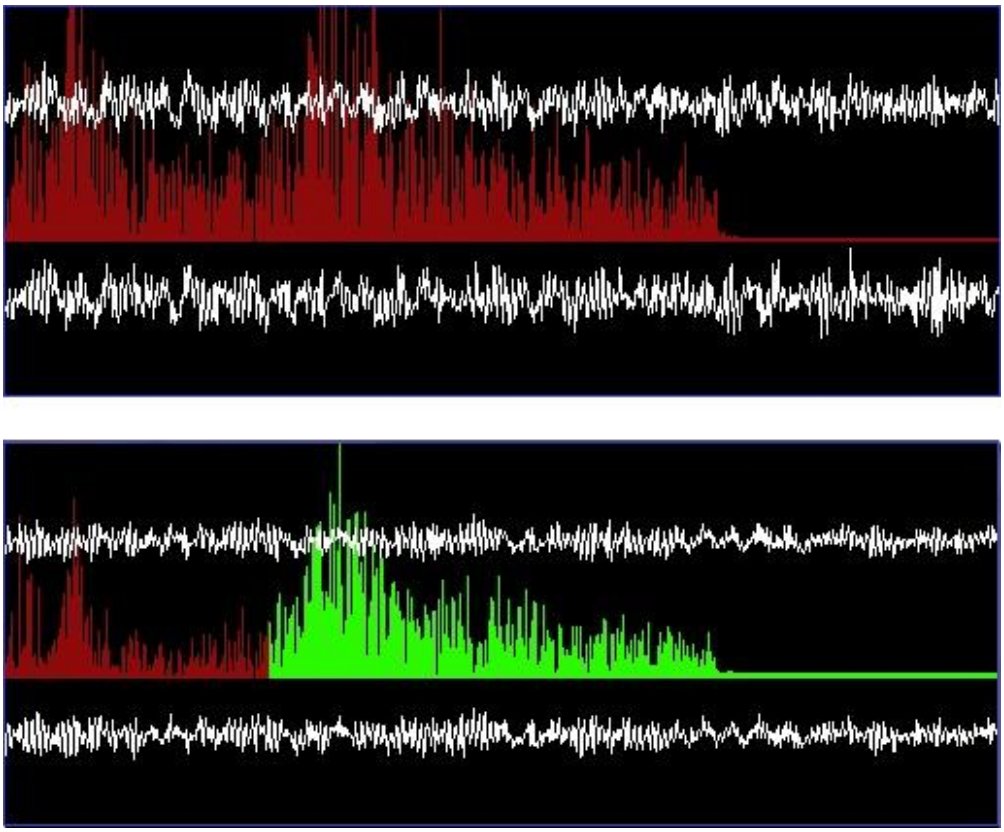


37. Απεικόνιση κυματομορφής και Fourier με τη χρήση χαμηλοπερατού φίλτρου

Στη παραπάνω στιγμιότυπο παρατηρούμε το αποτέλεσμα που λαμβάνουμε με την εφαρμογή του φίλτρου στο κομμάτι μας. Στη πάνω εικόνα **η συχνότητα αποκοπής fc** κυμαίνεται από **800 - 1000 Hz**, που ως συνέπεια επιτρέπει στις υψηλές συχνότητες να περνούν χωρίς να επηρεάζεται το κομμάτι. Στη κάτω εικόνα η συχνότητα αποκοπής έχει τιμές από **20 – 200 Hz** αποκόπτοντας έτσι τις υψηλές συχνότητες και αφήνοντας τις αρκετά **χαμηλές αναλλοίωτες.** Καταλαβαίνουμε λοιπόν ότι άμα η συχνότητα αποκοπής fc έχει αρκετά υψηλές τιμές, το σήμα μας παραμένει σχεδόν αναλλοίωτο.

Τα παραπάνω επιβεβαιώνονται από τις απεικονίσεις στην *εικόνα 37*, όπου παρατηρούμε στο πεδίο των συχνοτήτων(με κόκκινο), της πάνω εικόνας, όλες τις συχνότητες, ενώ στη κάτω μόνο τις χαμηλές. Πέρα από αυτό παρατηρούμε και τις αλλαγές στο πλάτος(ένταση) της κυματομορφής(με άσπρο) από τη μία κατάσταση στην άλλη, όπου κάτω τείνει να γίνει ευθεία.

Στο παρακάτω στιγμιότυπο παρατηρούμε τη χρήση τις υψιπερατού φίλτρου.



38. Απεικόνιση κυματομορφής και Fourier με τη χρήση υψιπερατού φίλτρου

Στη περίπτωση τώρα του υψιπερατού φίλτρου έχουμε ακριβώς την αντίθετη λειτουργία. Η **συχνότητα αποκοπής fc** λειτουργεί για να κόβει συχνότητες χαμηλότερες από αυτή. Έτσι στην *εικόνα 38* έχουμε τη πρώτη περίπτωση όπου η συχνότητα αποκοπής παίζει ανάμεσα στα **500Hz,** αφήνοντας έτσι ένα μεγάλος μέρος των συχνοτήτων να περάσει, ενώ στη δεύτερη περίπτωση(κάτω) η συχνότητα αποκοπής είναι περίπου **3500Hz** κόβοντας έτσι τις χαμηλές και αφήνοντας απαράλλακτες τις υψηλές. Για να κατανοήσουμε ακριβώς τη λειτουργία της fc, στο πεδίο των συχνοτήτων, οι

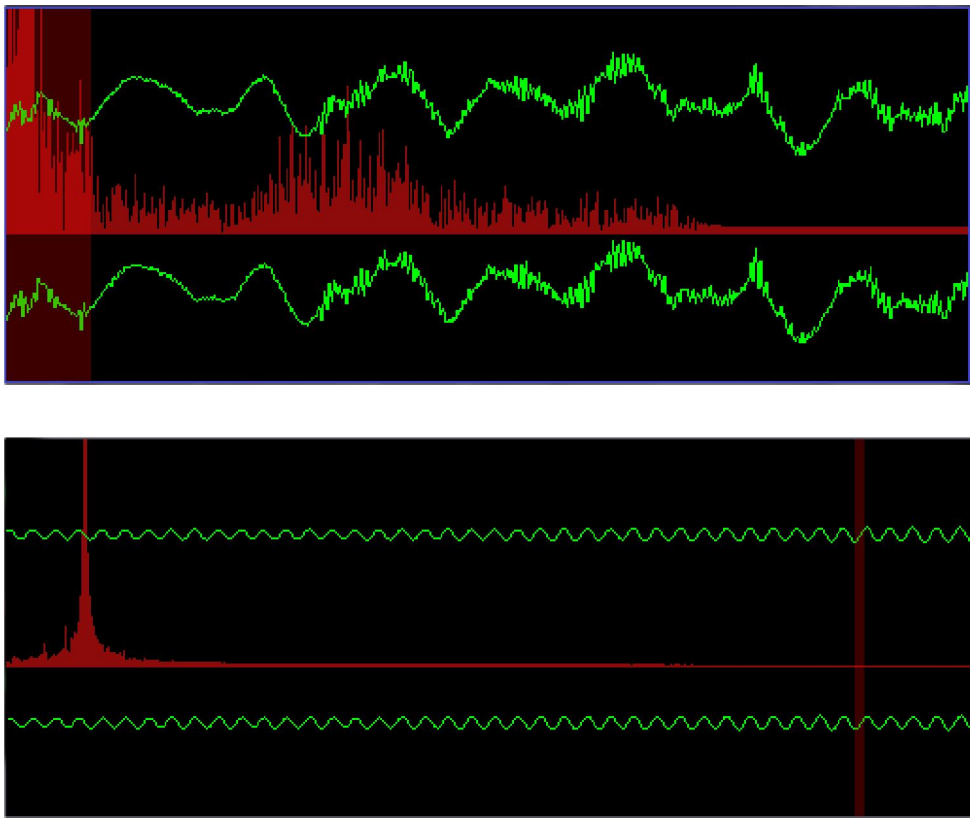
συχνότητες που μένουν απαράλλακτες είναι **με πράσινο χρώμα** ενώ αυτές που έχουν μειωμένη ένταση από τη χρήση του φίλτρου είναι **με κόκκινο.** Τέλος παρατηρούμε τη μειωμένη ένταση του πλάτους που είναι αρκετά εμφανής στη κυματομορφή(με άσπρο).

4.5.2 Ζωνοπερατά / Ζωνοφρακτικά φίλτρα

Όπως και με τα προηγούμενα φίλτρα θα ακολουθήσουμε και εδώ την ίδια μέθοδο υλοποίησης μέσω της minim και θα παρατηρήσουμε τις διαφορετικές αλλαγές στο οπτικό υλικό. Παραπάνω εξηγήσαμε θεωρητικά τη λειτουργία των Ζωνοπερατών/Ζωνοφρακτικών φίλτρων, όπου επιλέγουν ένα **εύρος από συχνότητες**, μεταξύ των συχνοτήτων αποκοπής, και είτε τις κόβουν είτε επιτρέπουν τη διέλευση τους.

Στη περίπτωση του ζωνοπερατού φίλτρου, όπως καταλαβαίνουμε και από την ονομασία του, επιτρέπει στις συχνότητες που βρίσκονται ανάμεσα στις συχνότητες αποκοπής, να περνούν ανεπηρέαστες(ζώνη διέλευσης), με ένα εύρος ζώνης μεγέθους f2 – f1.

Στο παρακάτω στιγμιότυπο έχουμε εικονική αναπαράσταση του ζωνοπερατού φίλτρου



39. Απεικόνιση κυματομορφής και Fourier με τη χρήση ζωνοπερατού φίλτρου

Παραπάνω παρατηρούμε τη χρήση του φίλτρου σε δυο διαφορετικές καταστάσεις. Στη πάνω κατάσταση έχουμε το φίλτρο στο μέγιστο εύρος ζώνης και με κεντρική συχνότητα f0 αρκετά χαμηλή, έτσι λαμβάνουμε ένα σήμα **σχετικά ανεπηρέαστο** που περιέχει σχεδόν όλες τις συχνότητες (τουλάχιστον από αυτές που μας γίνονται αντιληπτές). Στη δεύτερη τώρα κατάσταση επιλέξαμε τη

κεντρική συχνότητα να βρίσκεται αρκετά υψηλά σε σχέση με τη προηγούμενη, με αποτέλεσμα να κόψει αρκετές από τις χαμηλές συχνότητες. Πέρα από αυτό φέραμε το εύρος ζώνης σε ένα χαμηλό επίπεδο επίσης για να παρατηρήσουμε κατά πόσο μεταβάλλεται η ζώνη διέλευσης και τι αποτέλεσμα θα έχουμε οπτικά και ακουστικά.

Στη πρώτη κατάσταση είχαμε κεντρική συχνότητα f0 = 190 Hz και εύρος ζώνης κοντά στα 500 Hz, ενώ στη δεύτερη κεντρική συχνότητα 1600 Hz και εύρος ζώνης 50 – 80 Hz. Το πάχος του ορθογωνίου που διαγράφεται στην εικόνα όπως και η τοποθεσία του στο x άξονα επηρεάζεται από το εύρος ζώνης και την f0 αντίστοιχα.

Παρατηρήσαμε τις επιδράσεις ενός ζωνοπερατού φίλτρου στο κομμάτι μας. Το ζωνοφρακτικό φίλτρο τώρα λειτουργεί ακριβώς με την αντίστροφη διαδικασία. Αφήνει δηλαδή τις έξω συχνότητες ανεπηρέαστες και "φράζοντας " αυτές που βρίσκονται μέσα στο εύρος ζώνης του φίλτρου.

Για να κλείσουμε με τη κατηγορία αυτών των φίλτρων, πάμε να δούμε πως υλοποιούνται ακριβώς μέσω της minim. Λόγο το ότι όλα τα φίλτρα υλοποιούνται ακριβώς με την ίδια μέθοδο μέσω της minim θα δούμε πως γίνεται στη περίπτωση του υψιπερατού φίλτρου.

- Αρχικά ξεκινάμε με τα imports και τη δημιουργία τις αντικειμένου τύπου

```
import ddf.minim.effects.*;

HighPassSP hpf;
```

  HighPassSP (ή LowPass αν θέλαμε χαμηλοπερατό φίλτρο)

- Στη συνέχεια (μέσα στη setup()) υλοποιούμε το φίλτρο που δημιουργήσαμε ορίζοντας μία αρχική τιμή για τη συχνότητα αποκοπής fc (στη συγκεκριμένη περίπτωση 100Hz) και περνώντας το ρυθμό δειγματοληψίας σε αυτό. Αφού υλοποιήσουμε το φίλτρο κάνουμε χρήση τις μεθόδου addEffect στο κομμάτι τις.

```
hpf = new HighPassSP(100, groove.sampleRate());
groove.addEffect(hpf);
```

- Τελευταίο βήμα είναι η αντιστοίχηση τις συχνότητας αποκοπής με το δεξί τις χέρι(στη συγκεκριμένη περίπτωση η αντιστοίχηση έγινε με τη τοποθεσία του κέρσορα στο x άξονα). Με τη μέθοδο setFreq(cutoff), περνάμε τις διαφορετικές τιμές στη συχνότητα αποκοπής fc.(με όνομα μεταβλητής cutoff).

```
void mouseMoved()
{
  // map the mouse position to the range [1000, 14000],
  //an arbitrary range of cutoff frequencies
  float cutoff = map(mouseX, 0, width, 1000, 14000);
  hpf.setFreq(cutoff);
  println(cutoff);
}
```

4.6 Σύνοψη Πειραματικού μέρους

Στο κεφάλαιο αυτό ξεκινήσαμε με την ανάλυση της τελικής εφαρμογής και την παρουσίαση των επιμέρους εργαλείων που θα χρησιμοποιήσουμε για την υλοποίηση της. Συγκρίναμε τις διαφορετικές τεχνικές πάνω στην **αυτόματη εύρεση ρυθμού** και διαπιστώσαμε ποιες από αυτές θα επιλέξουμε σύμφωνα με τα ειδή μουσικής που έχουμε στη διάθεσή μας. Τέλος περάσαμε στη παρουσίαση και υλοποίηση των βασικών ψηφιακών φίλτρων και καταλάβαμε τη χρησιμότητα κάθε κατηγορίας. Ο λόγος για τον οποίο επιλέξαμε να εισάγουμε ένα ψηφιακό φίλτρο στο τελικό στάδιο της εφαρμογής εξυπηρετεί δύο σκοπούς. Πρώτον επιτρέπουμε στο χρήστη της εφαρμογής μία λειτουργία παραπάνω, μετατρέποντας την εφαρμογή σε κάτι παραπάνω από ένα κοινό media player, και δεύτερον τον βάζουμε σε θέση να κατανοήσει τη σημασία της συχνότητας σε ένα μουσικό κομμάτι και το τι αντίκτυπο έχει εικονικά και ακουστικά « παίζοντας» με αυτήν.

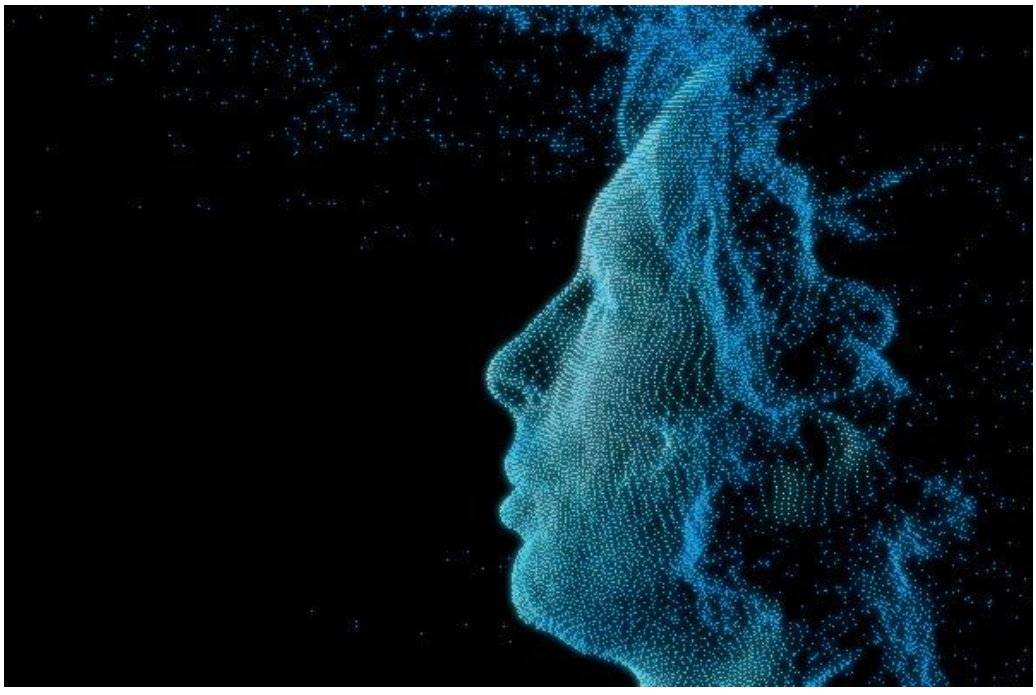
5. Δημιουργία εφαρμογής

5.1 Design chapter

Στο κεφάλαιο αυτό παρουσιάζεται η αρχική ιδέα για την εφαρμογή και πως διαμορφώθηκε μέχρι την ολοκλήρωσή της. Θα αναφέρουμε τα πρώτα προβλήματα που παρουσιάστηκαν κατά τη σύνδεση του kinect με την πλατφόρμα , πως αντιμετωπίστηκαν καθώς και τη σταδιακή αλλαγή του interface μετά από δοκιμές(από άλλους χρήστες), για να μπορέσουμε να πετύχουμε το πιο φιλικό περιβάλλον για το χρήστη.

Αρχικά η ιδέα ήταν η δημιουργία μίας εφαρμογής που θα δέχεται δεδομένα ήχου και θα εξάγει ένα είδος «αφηρημένου» οπτικού υλικού σε τρισδιάστατο χώρο. Η εφαρμογή θα δεχόταν τα δεδομένα αυτά μέσω του input του μικροφώνου του laptop γιατί μέχρι τότε δεν υπήρχε η ιδέα της εισαγωγής ενός player στην εφαρμογή. Έτσι ξεκίνησε μία έρευνα πάνω σε ήδη υπάρχουσες εφαρμογές στο χώρο των visuals και στις γλώσσες – περιβάλλοντα που χρησιμοποιούνται για την ανάπτυξη τέτοιων εφαρμογών. Οι πιο διαδεδομένες γλώσσες πάνω σε αυτές τις εφαρμογές ήταν C#, Processing και JAVA όπου χρησιμοποιούσαν για real time rendering σε 3D και 2D, τη βιβλιοθήκη OpenGL για τη διαδραστικότητα με τη GPU του συστήματος. Έτσι από τις παραπάνω επιλέχθηκε η processing γιατί όπως αναφέραμε παρέχει ένα φιλικό περιβάλλον για τον χρήστη, αρκετές βιβλιοθήκες για το χειρισμό του ήχου και συστήνεται γενικότερα για τη δημιουργία πολυμεσικών εφαρμογών, χωρίς την ιδιαίτερη χρήση αντικειμενοστραφή εννοιών.

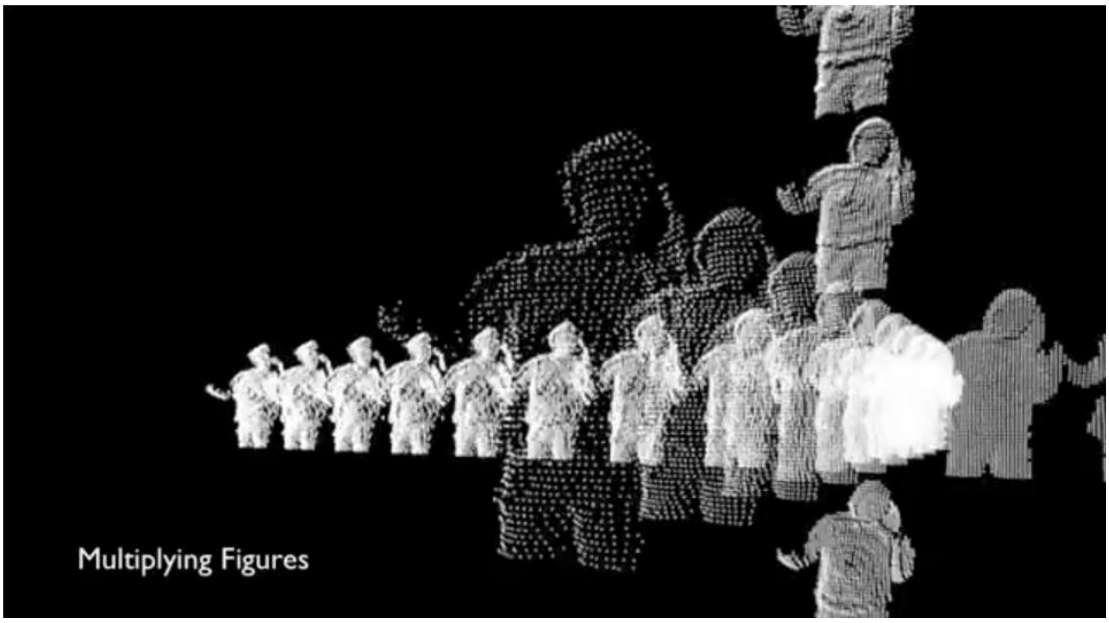
Στη συνέχεια ξεκινήσαμε να δουλεύουμε με τις διαφορετικές βιβλιοθήκες ήχου της processing καταλήγοντας έτσι στη minim. Η minim μας επιτρέπει αρκετά εύκολα να δημιουργήσουμε οπτικό υλικό από τον ήχο με μία διαδικασία που ονομάζεται **mapping** δηλαδή την εφαρμογή(αντιστοίχηση) παραμέτρων του ήχου(ένταση, συχνότητα) σε σχήματα, ευθείες ή σε οποιοδήποτε άλλο εικονικό στοιχείο. Σε αυτό το σημείο να αναφέρουμε ότι παράλληλα δουλεύαμε με το kinect και τα point clouds και δημιουργήθηκε η ιδέα για κάτι παρόμοιο με το videoclip του κομματιού House of cards των Radiohead. (Aaron Koblin)



40. Instance από το videoclip "House of Cards"

Το concept πίσω από το "House of cards" ,σύμφωνα με τον σχεδιαστή **Aaron Koblin** ,ήταν η δημιουργία βίντεο χωρίς τη χρήση κάμερας παρά μόνο αισθητήρων και scanner.

Όπως είχαμε εξηγήσει και παραπάνω χρησιμοποιώντας τη τεχνική point cloud μέσω του kinect, προβάλουμε σημεία στο χώρο μας, όπου κάθε σημείο έχει τις δικές του συντεταγμένες, δημιουργώντας έτσι την αίσθηση του τρισδιάστατου. Αυτό που θα κάναμε σε αυτό το σημείο είναι η αντιστοίχηση των points με τα δεδομένα που παίρνουμε από τον ήχο προκαλώντας έτσι ένα είδος motion σε όλο το point cloud, επηρεασμένο για παράδειγμα, από τις διαφορετικές ζώνες συχνοτήτων του κομματιού ή από την ένταση. Ένα παρόμοιο project, αλλά με λίγο διαφορετικές δυνατότητες και επιπρόσθετα χαρακτηριστικά, δημιούργησε ο Mike Knuepfel με ονομασία 3d Sensing and Visualization.[10]



41. Η εφαρμογή του Mike Knuepfel 3dSensing and Visualization

Εφαρμόζοντας τα δεδομένα του ήχου που λαμβάναμε μέσω της minim στα διαφορετικά points του point cloud δεν είχε το επιθυμητό αποτέλεσμα. Το πρόβλημα που παρουσιάστηκε ήταν αργό framerate και σε αρκετά σημεία η εφαρμογή αδυνατούσε να "τρέξει". Σε αυτό το σημείο δοκιμάστηκε διαφορετικός renderer, στη θέση του OpenGL. Τον αντικαταστήσαμε με τον P3D ,το προεπιλεγμένο renderer της processing για τρισδιάστατο σχέδιο και μπορούμε να πούμε ότι το αποτέλεσμα βελτιώθηκε αλλά απείχε αρκετά από το επιθυμητό. Στη συνέχεια δοκιμάσαμε να αλλοιώσουμε την ανάλυση του point cloud μειώνοντας τα συνολικά points όπως και τα mapped points, φτάνοντας έτσι την εφαρμογή σε ένα πολύ καλό σημείο αλλά παρόλα αυτά να εξακολουθεί να είναι ασύμφορη για τον υπολογιστή μας.

Σύμφωνα με τους παραπάνω πειραματισμούς διαπιστώσαμε ότι το πρόβλημα βρίσκεται στην FFT και όχι στο point cloud. Μπορεί η FFT να εκτελείται αρκετά εύκολα με τα εργαλεία που μας παρέχει η Processing αλλά είναι μία συνάρτηση που εκτελεί πολύπλοκους υπολογισμούς ανά

[10] http://3dsav.blogspot.gr/

δευτερόλεπτο κάνοντας την έτσι αρκετά ασύμφορη για την εφαρμογή μας. Καταλήγουμε έτσι να δημιουργήσουμε ένα concept χωρίς τη χρήση τρισδιάστατου rendering δοκιμάζοντας διαφορετικές τεχνικές και συναρτήσεις σε πιο απλά εικονικά στοιχεία δυσδιάστατου χώρου.

Κατά την έρευνα πάνω σε δισδιάστατα concept, ανακαλύπτουμε την ανίχνευση ρυθμού (beat detection) και με διαφορετικούς πειραματισμούς πάνω σε visuals δημιουργήσαμε το οπτικό υλικό που βρίσκεται στην εφαρμογή. Κατά την εκτέλεση της εφαρμογής σε αυτό το σημείο διαπιστώθηκε ότι τα δεδομένα ήχου από το input του μικροφώνου, δεν θεωρείται σωστή προσέγγιση όσον αφόρα την ανάλυση του ήχου διότι το εικονικό αποτέλεσμα δεν αντιπροσώπευε το κομμάτι, λόγο φασαρίας(noise) που δημιουργούταν κατά input.

Έτσι ορίσαμε ένα αντικείμενο carrier, που περιέχει μέσα το κομμάτι που επιλέγει ο χρήστης κάθε φορά και πάνω σε αυτό γίνεται όλη η ανάλυση. Συνοψίζοντας τώρα όλα τα παραπάνω, η εφαρμογή κατέληξε στο να πάρει τη μορφή ενός media player, που χειρίζεται ο χρήστης μέσω του kinect και των gestures του. Έτσι δημιουργήθηκε το παρακάτω interface για την εφαρμογή.

5.1.1 Δημιουργία του interface

42. Το αρχικό interface της εφαρμογής

Αρχικά κατά το στήσιμο του interface υπήρχε το σκεπτικό να έχουμε ένα μέρος της οθόνης κατειλημμένο από την εικόνα βάθους(depth image) του χρήστη και στο άλλο μισό να τρέχει το εικονικό υλικό(τα visuals).Η ιδέα για τα controls του χρήστη δεν ήταν ακόμα ξεκάθαρη και η μέθοδος προσέγγισης ήταν αρκετά μέτρια. Πιο συγκεκριμένα ο χρήστης είχε τρεις καταστάσεις που οριζόντουσαν από το ύψος του αριστερού χεριού. Πρώτη κατάσταση ήταν για την ένταση, δεύτερη για την επιλογή του κομματιού και τρίτη για την αλλαγή οπτικού υλικού. Ανάλογα με τη κατάσταση στην οποία βρισκόταν ο χρήστης, το δεξί χέρι λειτουργούσε σαν selector,δηλαδή κατά την κατάσταση της έντασης την αυξομείωνε σε σχέση με το ύψος του(κίνηση στον κάθετο άξονα), κατά τη κατάσταση οπτικού υλικού ,άλλαζε το visual(κίνηση στον οριζόντιο άξονα) και στην αλλαγή κομματιού κίνηση στον οριζόντιο άξονα επίσης.

Η μέθοδος αυτή αποδείχθηκε αρκετά κουραστική και πολύπλοκη καθώς ο χρήστης έπρεπε να έχει μία προκαθορισμένη στάση, όσον αφορά την τοποθεσία των χεριών του, καθ' όλη τη διάρκεια του τρεξίματος. Πέρα από αυτό η αλλαγή κομματιών όπως και visual δεν γινόταν μέσω switches – εντολών, προκαλώντας έτσι μία δυσλειτουργία κατά την περιήγηση του χρήστη σε αυτά.

Σύμφωνα με τα παραπάνω προβλήματα και δοκιμές από άλλους χρήστες έγιναν οι εξής αλλαγές στην εφαρμογή:

- δημιουργία switches – commands
- εισαγωγή καταστάσεων(τύπου modes) για πιο άνετη εναλλαγή λειτουργιών
- δημιουργία interface, εισαγωγή buttons, mode display bar και metadata(ID3) display

Δημιουργία switches εντολών: Μέσω των switches αποφύγαμε τη χρήση των counter που δυσκόλευαν το χειρισμό των διαφορετικών εντολών. Μέσω των Boolean μεταβλητών καταφέραμε μία πιο άνετη και φιλική προσέγγιση στο χειρισμό της εφαρμογής.

Εισαγωγή καταστάσεων: Ο χρήστης έχει την επιλογή μεταξύ τριών διαφορετικών καταστάσεων που ενεργοποιούσε μέσω των buttons που προστέθηκαν στο interface της εφαρμογής. Κατά την ενεργοποίηση της κάθε μίας ο χρήστης χειριζόταν μέσω των gestures του τις διαφορετικές λειτουργίες του player.Έτσι καταφέραμε να μην ”μπλέξουμε’’ τις διαφορετικές λειτουργίες μεταξύ τους.

Δημιουργία interface: Αρχικά εισάγαμε κουμπιά(buttons), για την επιλογή των διαφορετικών καταστάσεων του player στο πάνω μέρος του depth image. Μετά από αρκετές δοκιμές, ορίστηκε η τοποθεσία των κουμπιών, σύμφωνα με τη θέση του χρήστη. Τέλος δημιουργήσαμε display bars, για τα ID3 tags(καλλιτέχνης, είδος μουσικής κ.λπ.) και για την αναφορά των επιλεγμένων κατάστάσεων.

43. Βελτιωμένη εφαρμογή

Το τελικό στάδιο στην ολοκλήρωση της πλατφόρμα ήταν η τελειοποίηση των στοιχείων buttons και display bars, όπου δημιουργήθηκαν αποκλειστικά και μόνο μέσω του **Photoshop**. Επίσης προστέθηκε το στοιχείο δημιουργίας του interface αφού ολοκληρωθεί η ανίχνευση του χρήστη. Μία αρκετά σημαντική αλλαγή είναι η κατάργηση της δεύτερης οθόνης (σε αυτήν δηλαδή που εμφανιζόντουσαν τα visuals) με αποτέλεσμα, στην οθόνη όπου φαίνεται ο χρήστης να τρέχουν και τα visuals. Με τη τελευταία αυτή αλλαγή στο συνολικό interface της εφαρμογής, δώσαμε με ένα έξτρα κουμπί την επιλογή στο χρήστη να απενεργοποιεί την εικόνα βάθους, για να είναι τα visuals πιο ευδιάκριτα. Η τελική μορφή της εφαρμογής φαίνεται στο επόμενο κεφάλαιο.

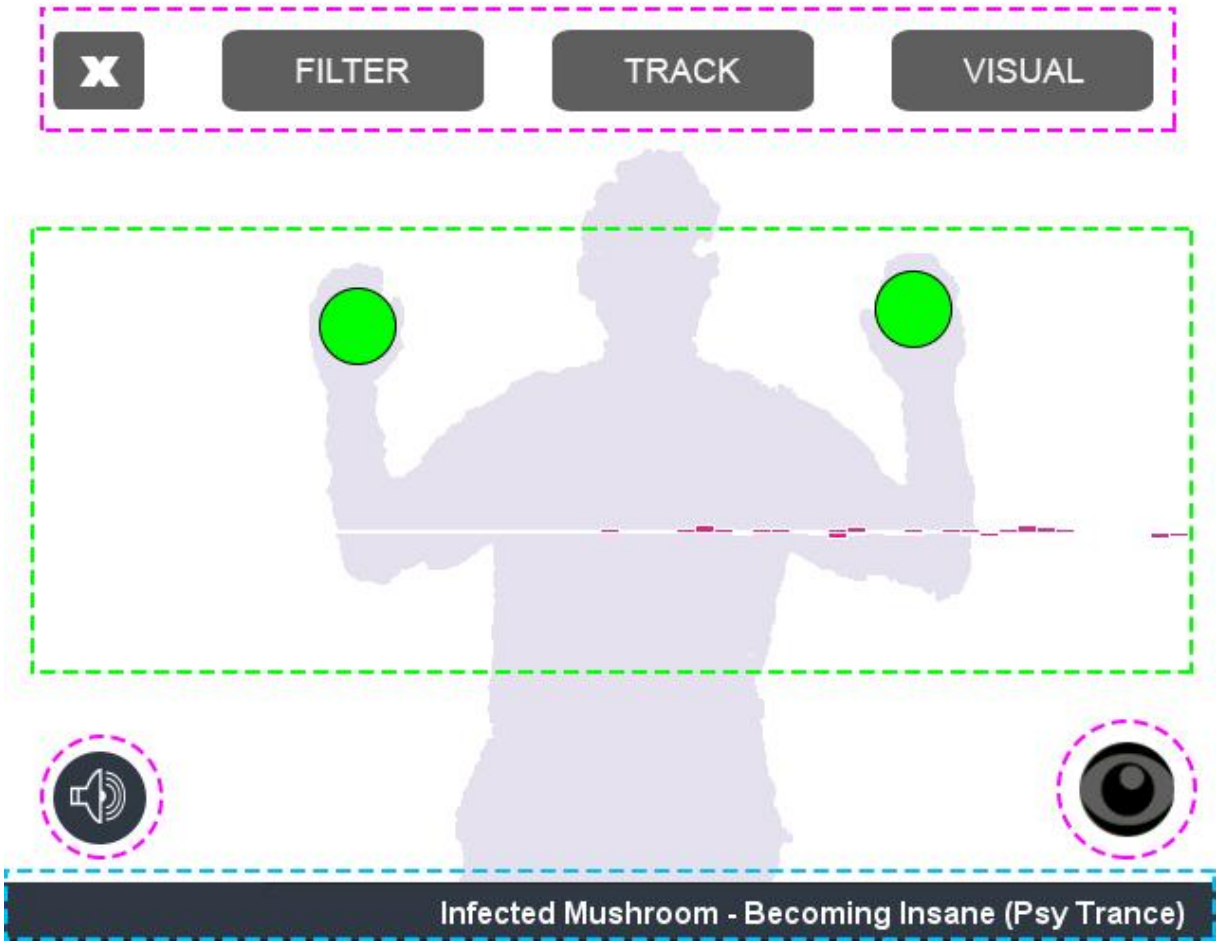
5.1.2 Περιβάλλον εφαρμογής

Τα κύρια στοιχεία που απαρτίζουν το περιβάλλον της εφαρμογής είναι τα εξής,

----- Τα κουμπιά της εφαρμογής, όπου εδώ έχουμε τα κουμπιά στο πάνω μέρος της οθόνης με τις εξής λειτουργίες : (από αριστερά προς τα δεξιά) κλείσιμο εφαρμογής, ενεργοποίηση φίλτρου, mode αλλαγής κομματιού, mode αλλαγής οπτικού υλικού. Τα κουμπιά που βρίσκονται στο κάτω μέρος, το αριστερό ρυθμίζει την ένταση, το δεξί «κρύβει» την εικόνα βάθους του χρήστη.

----- Τη περιοχή όπου εμφανίζεται το τρέχον οπτικό υλικό.

----- Τη κάτω μπάρα όπου εμφανίζεται κάθε φορά το επιλεγμένο μουσικό κομμάτι και επιπρόσθετες πληροφορίες για αυτό, που ανακτούνται από τα ID3 tags μέσω της minim.



44. τελικό interface εφαρμογής

5.2 Παρουσίαση υλοποίησης και κώδικα

5.2.1 Εισαγωγή βιβλιοθηκών, δηλώσεις, ορίσματα

Πρώτα ξεκινάμε με την εισαγωγή των βιβλιοθηκών που θα χρησιμοποιήσουμε. Η ddf.minim. είναι η κύρια βιβλιοθήκη για τη χρήση της minim. Στη συνέχεια εισάγουμε τις ddf.minim.effects και ddf.minim.analysis όπου η πρώτη μας επιτρέπει τη χρήση των φίλτρων και η δεύτερη την ψηφιακή ανάλυση των κομματιών με μεθόδους όπως Fourier κ.α. Τέλος εισάγουμε τη simpleOpenNI για τη χρήση του kinect μέσω της processing.

```
import ddf.minim.analysis.*;
import ddf.minim.*;
import ddf.minim.effects.*;
import SimpleOpenNI.*;

SimpleOpenNI kinect;
PImage userImage;
int userId,i=0,r=0,j=0,selectTrack=0,selectVisual=0,bg=0,selectFilterTrack=0,depthColor=221 ;
int[] userMap;
PImage rgbImage, depthOn, depthOff;
PImage filterButtonOn,filterButtonOff,trackButtonOn,trackButtonOff,visualButtonOn,
       visualButtonOff,volumeButtonOff,volumeButtonOn,playButton,metaBar,exitButton;
boolean trackingUser=false,filterMode=false,triggerVolume=false,volumeButton=false,triggerFilter=false,
        filterButton=false,triggerTrack=false,trackButton=false,triggerVisual=false,visualButton=false,
        hidePlayButton=false,changeTrack=false,changeVisual=false,start=false,changeFilterTrack=false,
        triggerDepth=false, depthButton=false;
float kickSize,snareSize,hatSize,arcSize,volumePercent;

AudioPlayer song1,song2,song3,song4,song5,song6,songCarry;
AudioMetaData meta1,meta2,meta3,metaCarry;
Minim minim;
FFT fft1,fft2,fft3,fft4,fft5,fft6,fftCarry;
BeatDetect beat1,beat2,beat3,carryBeat;
BandPass bpf1,bpf2,bpf3;
```

45. Βιβλιοθήκες,ορίσματα,δηλώσεις

Έπειτα προχωράμε στον ορισμό των αντικειμένων μας. Μέσω του αντικειμένου τύπου SimpleOpenNI θα χειριζόμαστε τα δεδομένα που παίρνουμε από το kinect. Μία σημαντική μεταβλητή είναι ο πίνακας ακεραίων με όνομα **userMap,** όπου σε αυτόν παρακάτω θα αποθηκεύουμε τα διαφορετικά pixels που απαρτίζουν το χρήστη, πετυχαίνοντας έτσι τη διαγραφή του background, εμφανίζοντας μόνο την εικόνα του χρήστη. Στη συνέχεια ορίζουμε τα αντικείμενα minim, και FFT για το μετασχηματισμό Fourier, επίσης τα αντικείμενα song που είναι ορισμένα σαν AudioPlayer θα κρατάνε τα διαφορετικά μουσικά κομμάτια και τα αντικείμενα τύπου BeatDetect για την ανίχνευση beat σε κάθε κομμάτι ξεχωριστά. Τέλος τα BandPass αντικείμενα κρατάνε τα διαφορετικά φίλτρα που στη setup() θα εφαρμόσουμε στα κομμάτια.

5.2.2 Δημιουργία Media player και αντικειμένων

Περνάμε στη setup τώρα και ορίζουμε μέγεθος 640x510 pixel και η εικόνα που λαμβάνουμε από το kinect είναι 640x480. Ο υπόλοιπος χώρος στο ύψος χρησιμοποιείται για τη μπάρα όπου εμφανίζονται τα metadata των κομματιών. Στη συνέχεια δημιουργούμε το αντικείμενο kinect που θα λαμβάνει δεδομένα από τη κάμερα βάθους με την εντολή **kinect.enableDepth()** και ενεργοποιούμε την ανίχνευση χρήστη για skeleton data. Παρακάτω συνεχίζουμε με τη δημιουργία του player.

```
void setup() {

  size(640, 510);
  kinect = new SimpleOpenNI(this);
  kinect.enableDepth();
  kinect.setMirror(true);

  //enable skeleton generation for all joints
  kinect.enableUser(SimpleOpenNI.SKEL_PROFILE_ALL);
```

46. environment and kinect setup

Στο ξεκίνημα της εφαρμογής δηλώσαμε τα αντικείμενα που θα χρησιμοποιήσουμε για τον player. Μέσα στη setup() περνάμε τα μουσικά κομμάτια μέσω του αντικείμενου minim που έχουμε ορίσει, σαν AudioPlayer αντικείμενα με buffersize 512. Στη συνέχεια δημιουργούμε τα αντικείμενα fft για κάθε ένα από αυτά με τα ορίσματα buffer size και sampleRate(ρυθμός δειγματοληψίας) και με παράθυρο **Hamming**. Το αποτέλεσμα της χρήσης ενός παραθύρου είναι να μειωθεί ο θόρυβος στο φάσμα κατά ένα ελάχιστο ποσοστό.

```
//minim setup, songs
minim = new Minim(this);
songl = minim.loadFile("01 Becoming Insane.mp3",512 ); //songs for media player mode
//fourier analysis
fftl = new FFT(songl.bufferSize(), songl.sampleRate());
fftl.window(FFT.HAMMING);
//filter songs
bpfl = new BandPass(440, 20, songl.sampleRate());
songl.addEffect(bpfl);
//beatdetection
beatl = new BeatDetect(songl.bufferSize(), songl.sampleRate());//beat
beatl.setSensitivity(300);//beat
//metaData
metal = songl.getMetaData();
```

47. minim, fft, filters, beats, metadata setup

Η εισαγωγή του φίλτρου σε κάθε κομμάτι γίνεται με την μέθοδο **addEffect(όνομα φίλτρου)** αφού πιο πριν έχουμε υλοποιήσει ένα φίλτρο με παραμέτρους , κεντρική συχνότητα, εύρος ζώνης και ρυθμός δειγματοληψίας. Με την εντολή **getMetaData()** λέμε στη minim να πάρει τα δεδομένα που βρίσκονται στο κάθε κομμάτι όπως όνομα καλλιτέχνη, είδος μουσικής, διάρκεια κ.α. και τα περνάμε στις μεταβλητές meta1,meta2 και meta3.

Δημιουργία αντικειμένων για τη διαδικασία εύρεσης Beat

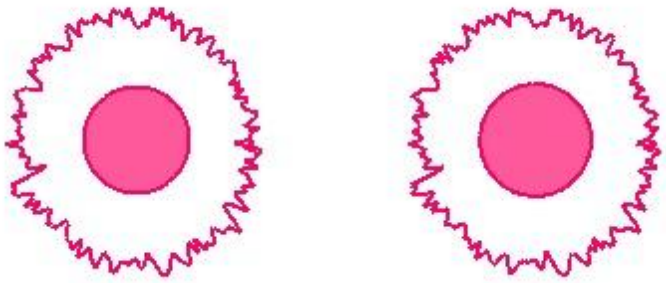
Η κλάση **BeatDetect** μας επιτρέπει να αναλύουμε μια ροή ήχου για beats μέσω των αλγόριθμων που αναφέραμε παραπάνω(ανάλυσης ενέργειας ήχου και συχνότητας ήχου).Στην εφαρμογή έχει χρησιμοποιηθεί ο αλγόριθμος εύρεσης ρυθμού μέσω της συχνότητας. Η Beatdetect κατά το κάλεσμα της μέσα στη draw() δημιουργεί ένα αντικείμενο με όνομα beat και δέχεται δύο ορίσματα, buffersize(μέγεθος buffer) και sample rate(ρυθμός δειγματοληψίας) όπως και η FFT.

Η μέθοδος **setSensitivity** χρησιμοποιείται για να βάλει ένα φρένο στην ανάλυση. Εάν ρυθμίσουμε την ευαισθησία έως 200, κάθε φορά που ο αλγόριθμος ανιχνεύει ένα beat θα περιμένει 200 χιλιοστά του δευτερολέπτου πριν από τη δοκιμή για ένα beat και επιστρέφοντας false στο ενδιάμεσο. Χρησιμοποιούμε αυτή τη μέθοδο εμποδίζουμε τον αλγόριθμο απ το να δίνει πάρα πολλές ψευδής ή θετικές τιμές. Η προεπιλεγμένη τιμή είναι 10, το οποίο είναι όπου δηλαδή δεν υπάρχει απόσβεση. Αν προσπαθήσουμε να ρυθμίσουμε την ευαισθησία σε μια αρνητική τιμή, ένα λάθος θα να αναφερθεί και θα οριστεί σε 10 αντί της αρνητικής τιμής.

Όλα τα παραπάνω που αναφέραμε σε αυτό το κεφάλαιο, όπως παρατηρούμε και στην *εικόνα 47*, γίνονται για κάθε ένα από τα κομμάτια που εισάγουμε στην εφαρμογή μας και βρίσκονται μέσα στη setup() της εφαρμογής μας. Στο παράδειγμα της εικόνας έχουμε ενδεικτικά μόνο για ένα κομμάτι αλλά στην εφαρμογή τα παραπάνω εφαρμόζονται σε όλα μας τα κομμάτια.

Παρακάτω εξηγούμε πιο αναλυτικά πως δουλεύει η εύρεση ρυθμού μέσω της συχνότητας. Στη draw της εφαρμογής μας βρίσκεται ο κώδικας που τρέχει καθ όλη τη διάρκεια της εκτέλεσης του προγράμματος. Πιο συγκεκριμένα στη draw() δημιουργούμε το interface της εφαρμογής(που εμφανίζονται τα metadata των κομματιών, η τοποθεσία των κουμπιών μας, χειρισμός των counters κ.α.).Επίσης μέσω της draw καλούνται και όλες οι συναρτήσεις μας για τη δημιουργία και απεικόνιση των visuals καθώς και τη ανίχνευση του χρήστη μέσω skeleton tracking.

5.2.3 Δημιουργία οπτικού υλικού



48. Waveform/ Beat freq visual

Κυματομορφή και ανίχνευση ρυθμού μέσω συχνότητας

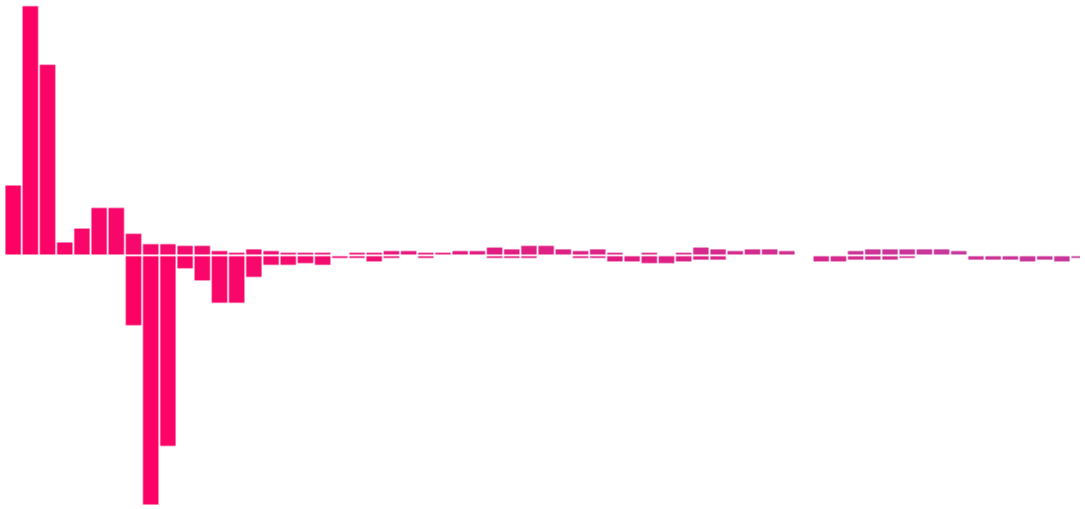
Το συγκεκριμένο visual αναπαριστά τη κυματομορφή για το δεξί και αριστερό κανάλι του τρέχον κομματιού σε κυκλική μορφή. Παράλληλα λειτουργεί η ανίχνευση ρυθμού μέσω ενέργειας του ήχου(βλέπε κεφάλαιο 2.5) όπου και εφαρμόζεται στις δύο ελλείψεις που βρίσκονται στο κέντρο του καθενός από τα δύο κανάλια. Αξιοσημείωτο είναι το πώς η παρέμβαση του φίλτρου επηρεάζει την αποδοτικότητα του beat detection και στο συγκεκριμένο visual. Καθώς κόβουμε ένα ευρύ φάσμα σημαντικών συχνοτήτων παρατηρούμε τη δυσλειτουργία του και καθώς το επαναφέρουμε στην ουδέτερη κατάσταση επανέρχεται πλήρως.

```
void wave(){
  int weight=2;
  if(selectVisual==1){
    beatWave1.detect(songCarry.left);
    beatWave2.detect(songCarry.right);

    float z = map(eRadius1, 20, 80, 60, 255);
    fill(253,2,102, z);
    if ( beatWave1.isOnset() ) eRadius1 = 80; if ( beatWave2.isOnset() ) eRadius2 = 80;
    ellipse(180, 230, eRadius1, eRadius2);
    ellipse(480, 230, eRadius1, eRadius2);
    eRadius1 *= 0.95; eRadius2 *=0.95;
    if ( eRadius1 < 20 ) eRadius1 = 20; if ( eRadius2 < 20 ) eRadius2 = 20;

    stroke(253,2,102);
    strokeWeight(2);
    float a = 0;
    float angle = (2*PI) / 200;
    int step = songCarry.bufferSize() / 200;
    for(int i=0; i < songCarry.bufferSize() - step; i+=step) {
      float x = 100 + cos(a) * (40 * songCarry.left.get(i) + 60); float y = 100 + sin(a) * (40 * songCarry.left.get(i) + 60);
      float x2 = 100 + cos(a + angle) * (40 * songCarry.left.get(i+step) + 60); float y2 = 100 + sin(a + angle) * (40 * songCarry.left.get(i+step) + 60);
      float xx = 100 + cos(a) * (40 * songCarry.right.get(i) + 60); float yy = 100 + sin(a) * (40 * songCarry.right.get(i) + 60);
      float xx2 = 100 + cos(a + angle) * (40 * songCarry.right.get(i+step) + 60); float yy2 = 100 + sin(a + angle) * (40 * songCarry.right.get(i+step) + 60);
      line(x+80,y+130,x2+80,y2+130); line(xx+380,yy+130,xx2+380,yy2+130);
      a += angle;
  }
```

49. Waveform/Beat freq code

FFT φάσμα συχνοτήτων

Η minim όπως έχουμε αναφέρει και πιο πριν παρέχει αρκετά εργαλεία στο χρήστη όσον αφορά την ανάλυση του ήχου. Μία αρκετά σημαντική λειτουργία της είναι ότι υλοποιεί μετασχηματισμό Fourier. Πιο συγκεκριμένα είναι ένας αποτελεσματικός τρόπος να υπολογίσουμε τον αρκετά πολύπλοκο **διακριτό μετασχηματισμό Fourier.** Ο μετασχηματισμός Fourier όπως είπαμε είναι ένας αλγόριθμος που μεταφέρει το σήμα μας από το πεδίο του χρόνου στο πεδίο των συχνοτήτων που συνήθως το αποκαλούμε φάσμα.



50. Fast Fourier Transform Visual

Το φάσμα που δημιουργούμε δεν αποτελείται από ξεχωριστές συχνότητες αλλά από **ζώνες συχνοτήτων** επικεντρωμένες σε συγκεκριμένες συχνότητες. Η κεντρική συχνότητα κάθε ζώνης εκφράζεται συνήθως ως ένα κλάσμα του ρυθμού δειγματοληψίας του σήματος στο πεδίο του χρόνου και ισούται με τον δείκτη της ζώνης συχνοτήτων δια το συνολικό αριθμό των ζωνών. Ο συνολικός αριθμός των ζωνών είναι συνήθως ίσο με το μήκος του σήματος στο πεδίο του χρόνου, αλλά η πρόσβαση παρέχεται μόνο σε ζώνες συχνοτήτων με τους δείκτες λιγότερο από το μισό του μήκους, επειδή αντιστοιχούν σε συχνότητες κάτω από τη **συχνότητα Nyquist**. Με άλλα λόγια, δίνεται ένα σήμα μήκους N, θα υπάρχουν **N / 2 ζωνών στο φάσμα.** Το φάσμα δημιουργείται με τη συνάρτηση fftwave() που καλείται και αυτή μέσα στη draw() όταν ό χρήστης την επιλέξει. Πέρα από το φάσμα έχουν προστεθεί και κάποια άλλα στοιχεία στο σχεδιασμό της αναπαράστασης για να κάνουν το αποτέλεσμα πιο φιλικό.

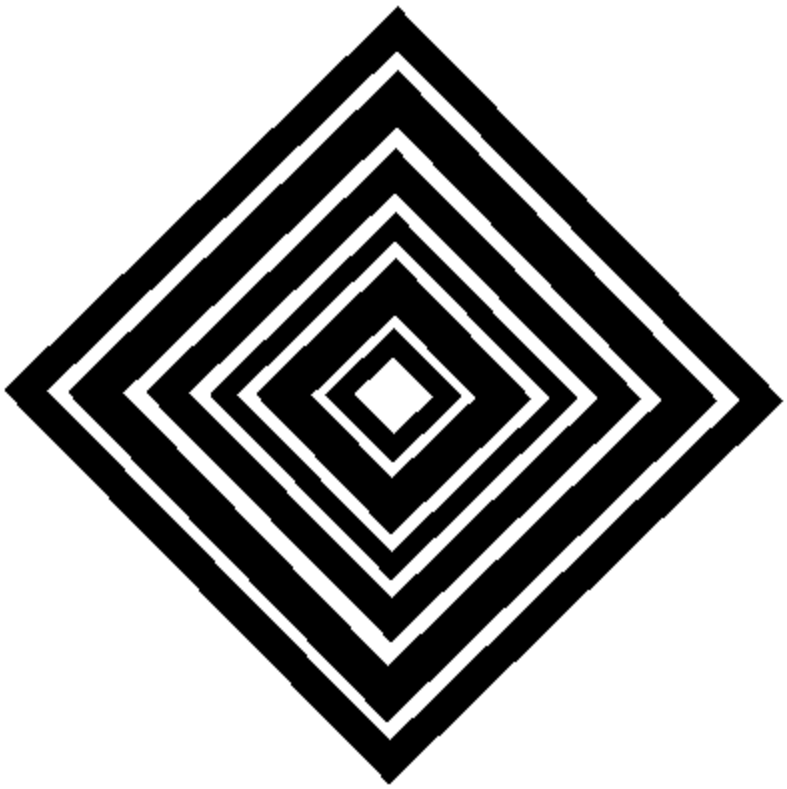
```
void fftwave(){
if(counter==0){
  for(i=0;i< carrierfft.specSize();i++){ //specSize is the number of bands
      stroke(0);                         //we compute the for the fft for each one
      strokeWeight(1);                   //of the bands and map them to the size of
      fill(200,0,50);                    //of each rectangle
      rect(4*i+640,height-carrierfft.getBand(i)*20-250,4,carrierfft.getBand(i)*20);
      stroke(0);
      fill(200,0,50);
      rect(4*i+640,height+carrierfft.getBand(i)*20-250,4,carrierfft.getBand(i)*20);
      }
   }
}
```

51. FFT visual code

Ανίχνευση ρυθμού

Το τελευταίο οπτικό υλικό δημιουργήθηκε με αντικείμενα εύρεσης ρυθμού (beat detection objects).Όπως αναφέραμε και πιο πάνω στη ανάλυση των βασικών αλγορίθμων για την εύρεση του ρυθμού που μας παρέχει η minim δύο είναι οι κύριες κατηγορίες. **Έυρεση ρυθμού μέσω ενέργειας** και **εύρεση ρυθμού μέσω συχνοτήτων.** Στη συγκεκριμένη εφαρμογή χρησιμοποιήσαμε τον δεύτερο αλγόριθμο. Η μέθοδος αυτή χρησιμοποιεί τρεις εμβέλειες συχνοτήτων , isKick(), isSnare() και isHat().

Με την χρήση της isRange() μπορούμε να ορίσουμε εμείς δικές μας ζώνες συχνοτήτων σε περίπτωση που δεν ικανοποιούμασταν από το αποτέλεσμα για τις προεπιλεγμένες ζώνες. Στην εφαρμογή χρησιμοποιήθηκαν οι προεπιλεγμένες ζώνες μιας και συστήνονται για τη χρήση ηλεκτρονικής μουσικής όπως και στην εφαρμογή μας.



52. Beat detection Visual

Το οπτικό υλικό δημιουργείται από τη συνάρτηση beatsquare() που καλείται μέσα από τη draw() όπως και οι προηγούμενες. Η συνάρτηση αυτό που κάνει είναι να δημιουργεί 6 τετράγωνα διαφορετικού μεγέθους. Κάθε ένα από αυτά έχει αντιστοιχηθεί με μία από τις τρεις ζώνες. Δύο τετράγωνα αλληλεπιδρούν με τις ψηλές ζώνες συχνοτήτων(hatSize) δύο με τις μεσαίες (snareSize) και δύο με τις χαμηλές(kickSize).Όταν ανιχνεύεται beat και αναλόγως τη ζώνη συχνότητας αλλάζει το μέγεθος των αντίστοιχων τετραγώνων δημιουργώντας έτσι μία αίσθηση προοπτικής και βάθους. Με τη χρήση της constrain() περιορίζουμε τα μεγέθη σε 2 τιμές που εναλλάσσονται, έτσι όταν ανιχνευτεί beat σε μία ζώνη συχνότητας, αυτόματα τα τετράγωνα που έχουμε αντιστοιχήσει αλλάζουν μέγεθος στιγμιαία και στη συνέχεια επανέρχονται στο αρχικό μέγεθος ομαλά(fade out).

```
void beatsquare(){
  if(selectVisual==2){
    carryBeat.detect(songCarry.mix);
    if ( carryBeat.isKick()==true ) kickSize = 20;
    if ( carryBeat.isSnare()==true) snareSize = 10;
    if ( carryBeat.isHat()==true )  hatSize = 10;
    stroke(0);
    translate(width/2, height/2 + 30);
    rotate(0.8);
    strokeWeight(kickSize);noFill();rect(-19,-19,38, 38);
    strokeWeight(snareSize);noFill();rect(-39,-39,78,78);
    strokeWeight(kickSize);noFill();rect(-60,-60, 120, 120);
    strokeWeight(hatSize);noFill();rect(-80,-80, 160, 160);
    strokeWeight(snareSize);noFill();rect(-105,-105, 210, 210);
    strokeWeight(hatSize);noFill();rect(-130,-130, 260, 260);
    kickSize = constrain(kickSize * 0.95, 10, 20);
    snareSize = constrain(snareSize * 1.05, 10, 20);
    hatSize = constrain(hatSize * 1.05, 10, 20);
    }
  }
```

53. Beat detection code

5.2.4 Skeleton tracking

Παραπάνω εξηγήσαμε συνοπτικά πως λειτουργεί η ανίχνευση του χρήστη μέσω της κάμερας βάθους kinect και εξηγήσαμε πως τα gestures του χρήστη αποτελούν τα input της εφαρμογής.

Πάμε τώρα να δούμε πιο αναλυτικά τα στάδια από τα οποία περνάει το πρόγραμμά μας για να ανιχνεύσει το χρήστη.

Τα τρία βασικά στάδια είναι :

1. εντοπισμός χρήστη(simple user detection)
2. βαθμονόμηση χρήστη(user calibration process)
3. πλήρως ανιχνευμένος χρήστης(fully tracked user)

Η διαδικασία ξεκινά όταν ένας χρήστης εισέρχεται στη σκηνή. Σε αυτό το σημείο η OpenNI εντοπίζει το χρήστη και ελέγχει αν είναι υποψήφιος για εντοπισμό. Από το σκίτσο μας καλείται η συνάρτηση onNewUser**()**.Μέσα από τη συνάρτηση αυτή μπορούμε να ξεκινήσουμε τη διαδικασία παρακολούθησης του χρήστη με τη κλήση της συνάρτησης **startPoseDetection().**Αυτή η συνάρτηση λέει στην OpenNI να παρακολουθεί αν ο χρηστής που μπήκε στη σκηνή έχει λάβει τη στάση για calibration. Η στάση που πρέπει να πάρει ο χρήστης γνωστή και ως **calibration pose(**φαίνεται στη παραπάνω φωτογραφία) λέει στην OpenNI να ξεκινήσει τη βαθμονόμηση του χρήστη. Μέχρι ο χρήστης να λάβει τη παραπάνω στάση η συνάρτηση θα ελέγχει ξανά και ξανά μέσω της startPoseDetection().

Έστω τώρα ότι ο χρήστης λάβει τη προκαθορισμένη στάση η OpenNI θα καλέσει από το σκίτσο μας τη συνάρτηση **onStartPose().**Μετά για να συνεχίσουμε τη βαθμονόμηση καλούμε τη **requestCalibrationSkeleton()** που μέσω αυτής ξεκινάει ουσιαστικά η OpenNI να παρακολουθεί το χρήστη. Με το που τελείωση η βαθμονόμηση καλείται η **onEndCalibration()** από το πρόγραμμα μας. Σε περίπτωση που η βαθμονόμηση ήταν επιτυχής έχουμε πρόσβαση στα μέλη του σκελετού μας με το κάλεσμα της **startTrackingSkeleton()** αλλιώς άμα παρουσιάστηκε κάποιο πρόβλημα με διαδικασία της calibration ξαναγυρνάμε στην startPoseDetection().Στην OpenNI η στάση που παίρνει ο χρήστης αναφέρεται σαν **Psi pose** και έτσι αναφέρεται και στον κώδικα παρακάτω.

```
// user-tracking callbacks!
void onNewUser(int userId) {
println("start pose detection");
kinect.startPoseDetection("Psi", userId);
}
void onEndCalibration(int userId, boolean successful) {
if (successful) {
println(" User calibrated !!!");
kinect.startTrackingSkeleton(userId);
} else {
println(" Failed to calibrate user !!!");
kinect.startPoseDetection("Psi", userId);
}
}
void onStartPose(String pose, int userId) {
println("Started pose for user");
kinect.stopPoseDetection(userId);
kinect.requestCalibrationSkeleton(userId, true);
}
```

54. User tracking callbacks

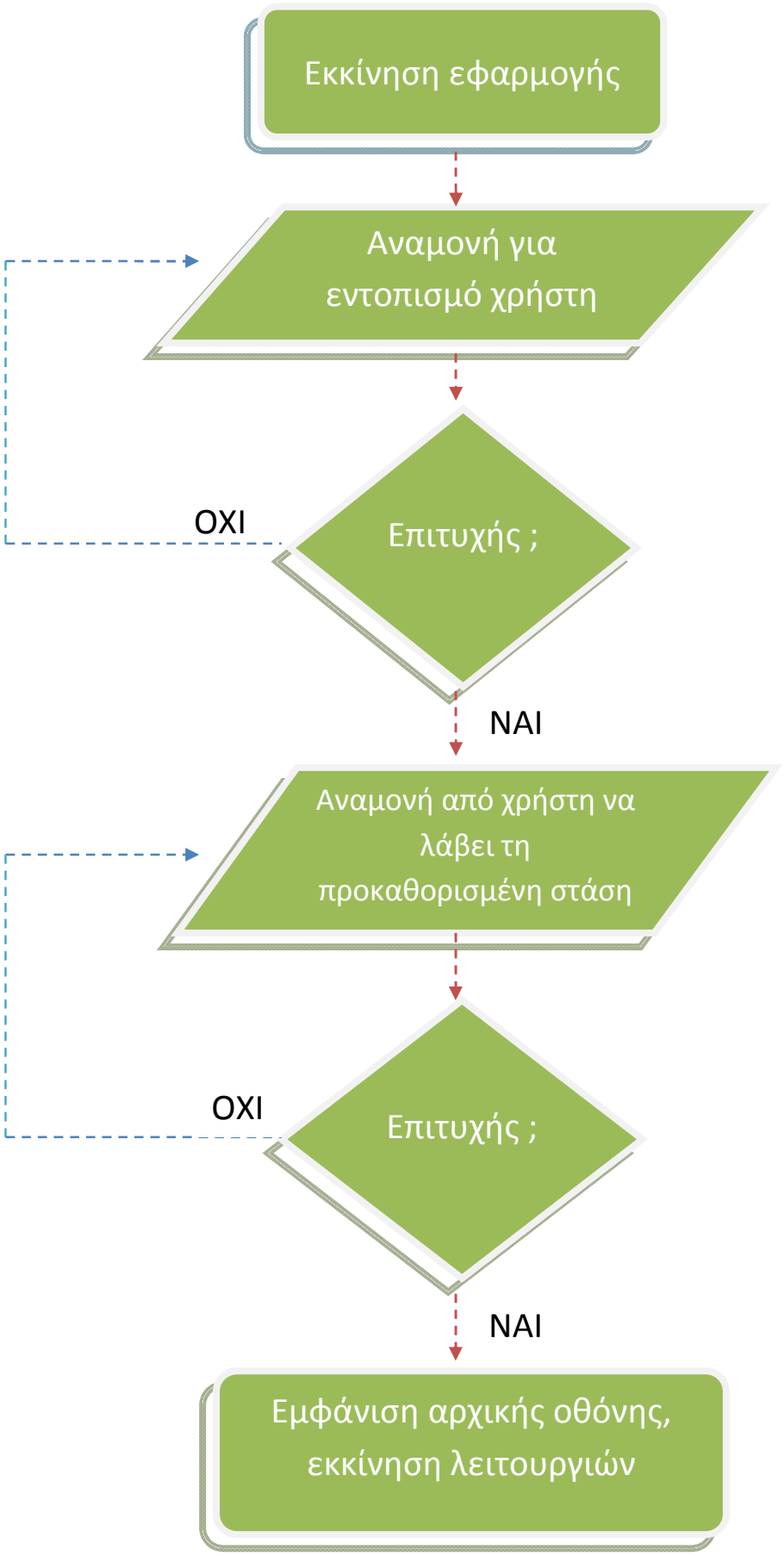
Έτσι αφού έχουμε τελειώσει με τη calibration και ήταν επιτυχής περνάμε στη συνάρτηση **tracking()** που καλείται μέσα της draw και αυτή και περιέχει όλα τα commands για τα gestures μας. Στην εφαρμογή μας τα inputs τα παίρνουμε μέσω του δεξιού και του αριστερού χεριού του χρήστη. Δεν ορίσαμε τα υπόλοιπα μέλη του σώματος μιας και ο χρήστης χειρίζεται καθαρά την πλατφόρμα μέσω της θέσης των χεριών του. Έτσι ξεκινάμε με τα ορίσματα και την αντιστοίχηση για το κάθε μέλος του χρήστη.

```
// if we're successfully calibrated
if ( kinect.isTrackingSkeleton(userId)) {
// make a vector to store the left hand and another for the right hand
PVector leftHand = new PVector();
PVector rightHand= new PVector();
// put the position of the left hand into that vector
kinect.getJointPositionSkeleton(userId,SimpleOpenNI.SKEL_RIGHT_HAND,rightHand);
kinect.getJointPositionSkeleton(userId,SimpleOpenNI.SKEL_LEFT_HAND,leftHand);
// convert the detected hand position
// to "projective" coordinates
// that will match the depth image
PVector convertedLeftHand = new PVector();
PVector convertedRightHand = new PVector();
kinect.convertRealWorldToProjective(leftHand, convertedLeftHand);
kinect.convertRealWorldToProjective(rightHand, convertedRightHand);
```

55. Ορίσματα,αντιστοίχηση μελών και μετατροπή συντεταγμένων

Όπως είπαμε δημιουργούμε τις μεταβλητές για το αριστερό και δεξί μας χέρι και τις ορίζουμε σαν διανύσματα(vectors).Στις επόμενες γραμμές αντιστοιχίζουμε τις συντεταγμένες των χεριών μας με τις μεταβλητές που ορίσαμε. Τέλος με την εντολή convertRealWorldToProjective η SimpleOpenNI μας παρέχει τη δυνατότητα να μετατρέπουμε τις συντεταγμένες του χώρου μας σε προβαλλόμενες συντεταγμένες που θα μπορεί το kinect να τις δέχεται σαν inputs.

Για να κατανοήσουμε καλύτερα τη σειρά των βημάτων της εφαρμογής δημιουργήσαμε το ακόλουθο διάγραμμα ροής .



56. Διάγραμμα ροής εφαρμογής

5.2.5 Επεξήγηση των inputs και modes

Ο χρήστης εναλλάσσει τα βασικά modes της εφαρμογής μέσω των κουμπιών που βρίσκονται στο πάνω μέρος της οθόνης. Τα κουμπιά έχουν δημιουργηθεί με τέτοιο τρόπο έτσι ώστε ανά πάσα στιγμή **ένα και μόνο mode** μπορεί να είναι ενεργοποιημένο. Έτσι από αριστερά προς τα δεξιά έχουμε το κουμπί για τα φίλτρα (filter mode), για την αλλαγή κομματιού (track mode) και κουμπί για την αλλαγή visual (visual mode). Καθώς τα χέρια του χρήστη περνάν από τη θέση του κάθε κουμπιού ενεργοποιείται το αντίστοιχο mode. Επίσης κάθε φορά που ο χρήστης αλλάζει κομμάτι, μέσω του track mode, βλέπουμε πληροφορίες για αυτό(όνομα καλλιτέχνη ,όνομα κομματιού, είδος μουσικής, διάρκεια), στο κάτω μέρος της οθόνης.

Πέρα από τα κουμπιά καταστάσεων που βρίσκονται στο πάνω μέρος της οθόνης έχουμε άλλα δύο, για την **ένταση** και για την **απόκρυψη της εικόνας βάθους** του χρήστη. Στη πρώτη περίπτωση αφού ενεργοποιηθεί η ένταση, με την κατακόρυφη κίνηση του δεξιού χεριού αυξομειώνουμε την ένταση στο κάθε κομμάτι. Η δεύτερη τώρα περίπτωση είναι αρκετά πιο απλή, αλλάζοντας το χρώμα των ενεργών pixel(δηλαδή των pixel που περιέχουν το χρήστη) σε άσπρο. Με αυτή προσθήκη καταφέρνουμε να παρατηρούμε τα visuals πιο καθαρά, τώρα αν ο χρήστης επιθυμεί να απεικονίζεται στην οθόνη έχει την επιλογή να επαναφέρει τα ενεργά pixels στα προηγούμενα χρώματα τους. Ακολουθεί το κομμάτι κώδικα για την ένταση.

```
 //VOLUME CONTROL
if(convertedLeftHand.x<100 && convertedLeftHand.y>380 && triggerVolume==true){
     volumeButton = !volumeButton;
     triggerVolume=false;
     }
 else if(convertedLeftHand.x>100){
     triggerVolume=true;
 }
if (volumeButton==true){
//set gain using the right hand
song1.setGain(map(convertedRightHand.y, 280, 120, -30, 10));
song2.setGain(map(convertedRightHand.y, 280, 120, -30, 10));
song3.setGain(map(convertedRightHand.y, 280, 120, -30, 10));
arcSize=map(convertedRightHand.y, 280, 120, 0, 2*PI);
volumePercent=map(convertedRightHand.y,280, 120,0,99);
}
```

57. gain mode setup

Πάμε τώρα να δούμε τα βασικά modes λίγο πιο αναλυτικά. Αρχικά στη setup δημιουργήσαμε τα εφέ και μετά τα εφαρμόσαμε σε επιλεγμένα κομμάτια. Να σημειώσουμε ότι το φίλτρο που επιλέξαμε είναι το **ζωνοπερατό φίλτρο**(ή αλλιώς bandpass filter στα αγγλ.) και ό λόγος είναι ότι μας επέτρεπε να «παίξουμε» με τις συχνότητες σε ένα μεγαλύτερο βαθμό από τα «απλά» φίλτρα όπως το υψιπερατό ή το χαμηλοπερατό. Στο κεφάλαιο με τα φίλτρα εξηγήσαμε τον τρόπο λειτουργίας του φίλτρου όσον αφορά τις συχνότητες αποκοπής και το εύρος ζώνης. Οι παράμετροι του φίλτρου που εμείς χειριζόμαστε μέσω της minim είναι **το εύρος ζώνης** και τη **ζώνη διέλευσης**(και όχι τις συχνότητες αποκοπής). Μέσω της κεντρικής συχνότητας περιφερόμαστε στο φάσμα συχνοτήτων και παραλλήλως αυξομειώνουμε το μέγεθος του εύρους ζώνης.

Έτσι αφού ενεργοποιήσουμε το φίλτρο(με το αριστερό χέρι) η κάθετη κίνηση του δεξιού ρυθμίζει το εύρος ζώνης και η οριζόντια επιλέγει τη ζώνη διέλευσης.

```
//filter effects
float passBand = constrain(map(convertedRightHand.x,
width/2, 550, 100, 2000),100,2000);//constrain passBand so we can reduce the noise
bpf1.setFreq(passBand);
bpf2.setFreq(passBand);
bpf3.setFreq(passBand);
float bandWidth = constrain(map(convertedRightHand.y,
120, 330, 50, 500),50,500); //constrain bandwidth so we can reduce the noise
bpf1.setBandWidth(bandWidth);
bpf2.setBandWidth(bandWidth);
bpf3.setBandWidth(bandWidth);
```

58. filters setup

Παραπάνω γίνεται η αντιστοίχηση της ζώνη διέλευσης και του εύρος ζώνης με την οριζόντια και κάθετη κίνηση του δεξιού χεριού. Το κομμάτι κώδικα της *εικόνας 58,* εκτελούνται αφού ο χρήστης ενεργοποιήσει το φίλτρο και για όσο παραμένει σε αυτό. Αν τώρα απενεργοποιήσει το φίλτρο για να αλλάξει mode, σταματάει το τρέχον κομμάτι και παίζει πάλι απλά χωρίς εφέ.

Το δεύτερο mode κατά σειρά, όπως εμφανίζεται και στην οθόνη είναι για την περιήγηση στα εισαχθέντα κομμάτια. Η κατάσταση αυτή χειρίζεται τα counters των κομματιών. Πιο συγκεκριμένα αν το δεξί χέρι του χρήστη βρεθεί δεξιά θα πάμε στο επόμενο κομμάτι(αυξάνεται η select κατά ένα) ενώ αν βρεθεί αριστερά γυρνάμε στο προηγούμενο(μειώνεται η select). Ουσιαστικά είναι σαν να χρησιμοποιούμε τα κουμπιά prev και next σε ένα player.

```
//TRACK MODE
if(rightHand.x>-100 && rightHand.x<80 && rightHand.y>250 && trigger3==true){
buttonT=!buttonT;trigger3=false;
}
else if (rightHand.x>80)trigger3=true;
if(buttonT==true){
buttonV=false;buttonG=false;
if(convertedRightHand.x>500 && triggertrack==true){
select++; triggertrack=false;}
else if(convertedRightHand.x<500) triggertrack=true;
fill(0);textFont( loadFont("Bauhaus93-35.vlw") );text("Track mode",190,512);
}
```

59. track mode setup

Τέλος στο **Visual mode** ο χρήστης αλλάζει το οπτικό υλικό. Κάθε visual παίρνει πληροφορία για το κομμάτι και αλληλεπιδρά με αυτό σε πραγματικό χρόνο. Όπως και με το track mode έτσι και εδώ έχουμε μία μεταβλητή counter η οποία αυξάνεται και αυτή κατά ένα αν το δεξί χέρι βρεθεί στο σημείο που έχουμε ορίσει.

```
//VISUAL MODE
if(rightHand.x>400 && rightHand.y>250 && trigger2==true){
buttonV=!buttonV; trigger2=false;
}
else if(rightHand.x<400)trigger2=true;

if(buttonV==true){
buttonG=false;buttonT=false;
if(convertedRightHand.x>500 && triggervisual==true){        //visual mode on
counter++;
triggervisual=false;}
else if(convertedRightHand.x<500) triggervisual=true;
fill(0);textFont( loadFont("Bauhaus93-35.vlw") );text("Visual mode",190,512);
}
```

60. visual mode setup

5.3 Η εφαρμογή στη πράξη

Έχοντας δει στα προηγούμενα κεφάλαια τα βήματα που ακολουθήσαμε για την ολοκλήρωση της εφαρμογής και τα κομμάτια κώδικα από τα πιο κρίσιμα σημεία της, θα ρίξουμε τώρα μία πιο αναλυτική ματιά. Με τη χρήση στιγμιότυπων που τραβήχτηκαν κατά το τρέξιμο της εφαρμογής θα εξηγήσουμε ακριβώς πως λειτουργεί με τη χρήση των διαφορετικών gestures.

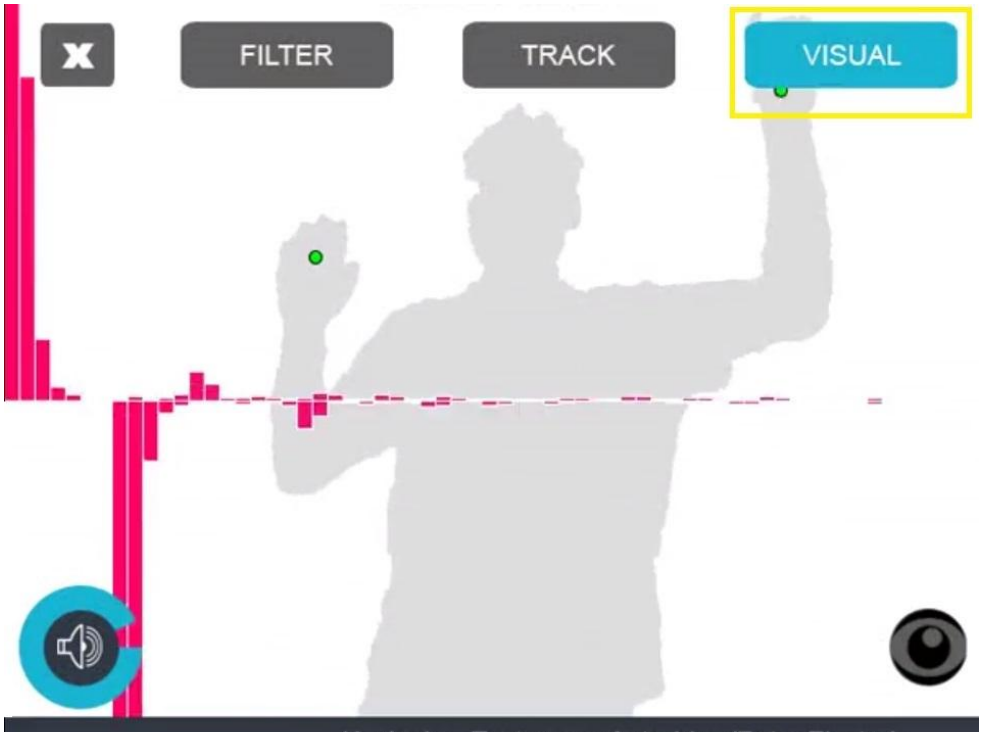
5.3.1 Αρχική οθόνη

Τρέχοντας την εφαρμογή το πρώτο πράγμα που παρατηρούμε είναι μία **άσπρη οθόνη**. Αυτή η κατάσταση ισχύει μέχρι το kinect να εντοπίσει κάποιο χρήστη στο οπτικό του πεδίο. Αφού ο χρήστης εισέλθει στη σκηνή αναγράφεται η φιγούρα του στην οθόνη με γκρίζο χρώμα και αυτό που περιμένει η εφαρμογή για να ξεκινήσει τη βασική λειτουργία της, είναι να πάρει ο χρήστης τη προκαθορισμένη στάση, ή αλλιώς τη **PSY pose**, ή οποία φαίνεται στη παρακάτω εικόνα.

61. Instance of the PSY pose

Έτσι μόλις το kinect εντοπίσει τη στάση αυτή ξεκινάει και η εφαρμογή με το βασικό interface της.

5.3.2 Ενεργοποιώντας τις διαφορετικές λειτουργίες

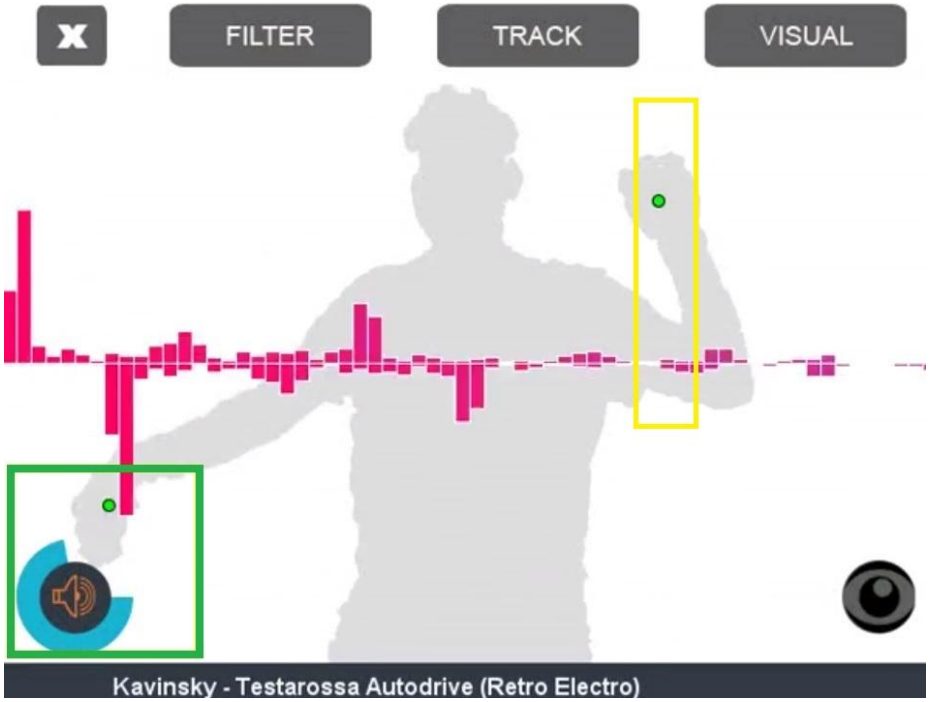


62. Ενεργοποιώντας το visual mode

Αφού τώρα έχει ξεκινήσει να λειτουργεί η εφαρμογή και ο χρήστης είναι βαθμονομημένος και παρακολουθείται, περιμένει από εμάς κάποια ενέργεια. Η επιλογές που έχουμε είναι είτε οι τρεις στο πάνω μέρος της οθόνης(δηλ. filter, track, visual), είτε οι δύο στο κάτω μέρος(volume control, η απενεργοποίηση της εικόνας βάθους). Στη παραπάνω εικόνα παρατηρούμε πως ενεργοποιείται το visual mode με το δεξί χέρι και έτσι ακριβώς ενεργοποιούνται και τα υπόλοιπα modes που βρίσκονται στο πάνω μέρος της οθόνης.

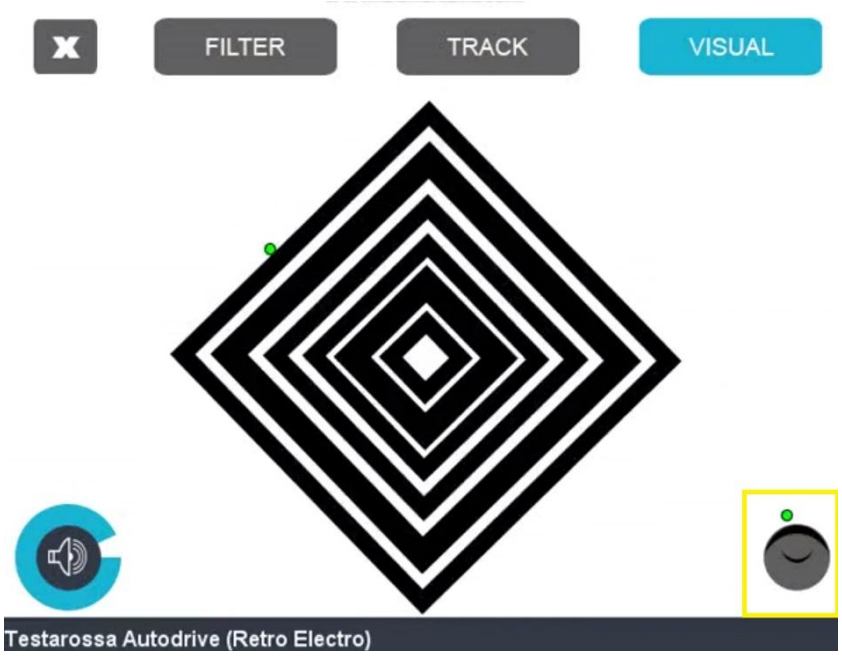
63. Το visual mode σε λειτουργία

Στην *εικόνα 63* έχουμε στο ενεργοποιημένο **visual mode**. Από τι στιγμή που ενεργοποιηθεί ένα από τα κουμπιά δημιουργούνται οι περιοχές δράσεις (ή στα αγγλ. area of effect), όπου με το δεξί η το αριστερό χέρι εκτελούνται οι επιλεγμένες λειτουργίες. Η πράσινη είναι η περιοχή δράσης για το αριστερό χέρι και η κίτρινη για το δεξί. Στη περίπτωση των visual και track mode που λειτουργούν με τον ίδιο τρόπο το αριστερό χέρι μειώνει τον counter κατά ένα και το δεξί χέρι τον αυξάνει, πετυχαίνοντας έτσι την πλοήγηση των εικονικών στοιχείων και μουσικών κομματιών αντίστοιχα.



64. ρυθμίζοντας την ένταση

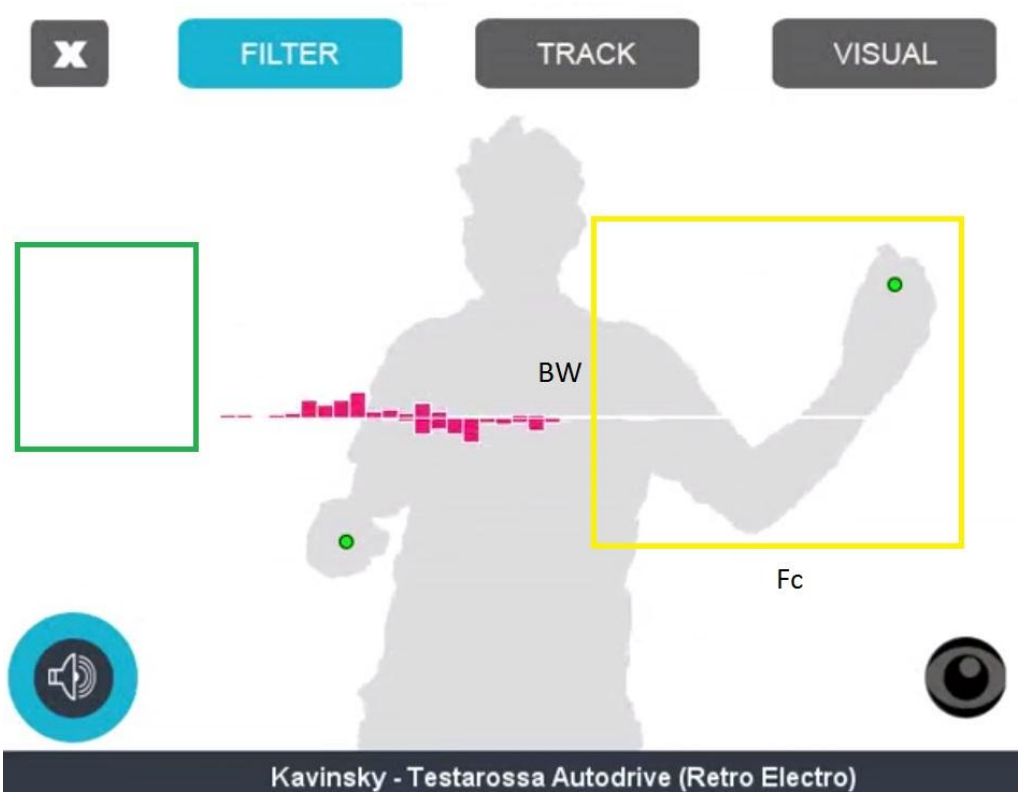
Στο mode της έντασης τώρα, το οποίο ενεργοποιείται όπως βλέπουμε και παραπάνω με το αριστερό χέρι(και το κουμπί αλλάζει χρώμα από άσπρο σε πορτοκαλί), αυτόματα αντιστοιχεί(mapping) το ύψος του δεξιού χεριού με την ποσοστιαία ένταση των κομματιών. Σε αυτή τη περίπτωση δεν μας ενδιαφέρει η τοποθεσία του δεξιού χεριού στον οριζόντιο άξονα αλλά μόνο στο κάθετο.



65. Αλλάζοντας χρώμα στα ενεργά pixels

Τα ενεργά pixels τα έχουμε ορίσει σαν τα pixel που απαρτίζουν το χρήστη τα οποία είναι αποθηκευμένα στην εφαρμογή μας στο πίνακα userMap []. Στο στιγμιότυπο της *εικόνας 65* όπου ενεργοποιούμε το **depthOff button**, δεν απενεργοποιούμε τα pixels όπως λέει το όνομα αλλά αλλάζουμε το χρώμα τους σε 255(δηλ. άσπρο) δίνοντας έτσι την επιλογή στο χρήστη να χειρίζεται την εφαρμογή χωρίς τη βοήθεια της εικόνας του παρά μόνο με τις κουκίδες που είναι mapped στο κάθε χέρι. Το button αυτό όπως και της έντασης είναι σχεδιασμένα για να μπορεί ο χρήστης να τα ενεργοποιεί ανά πάσα στιγμή ασχέτως σε ποια κατάσταση βρίσκεται. Αυτό δεν ισχύει για τα κουμπιά που βρίσκονται στο πάνω μέρος, που ένα μόνο μπορεί να είναι ενεργοποιημένο καθ' όλη τη διάρκεια.

Τέλος περνάμε στο πιο ενδιαφέρον mode της εφαρμογής, το **filter mode**.



66. filter mode in action

Το φίλτρο που χρησιμοποιούμε στην εφαρμογή όπως αναφέραμε και στα παραπάνω κεφάλαια είναι το **ζωνοπερατό φίλτρο**. Το mode αυτό επιλέγεται μέσω του αριστερού χεριού και δημιουργεί δύο περιοχές δράσεις, που φαίνονται και στην *εικόνα 66*. Το αριστερό χέρι μας επιτρέπει να αλλάζουμε κομμάτια όσο βρισκόμαστε στο mode αυτό και το δεξί χέρι χειρίζεται τις δύο παραμέτρους του φίλτρου, τη ζώνη διέλευσης και το εύρος της ζώνης αυτής. Όπως φαίνεται και στην εικόνα η περιοχή δράσης του δεξιού χεριού είναι η κίτρινη, όπου οι κάθετες συντεταγμένες(Y- άξονα) είναι mapped με το εύρος της ζώνης και οι οριζόντιες(X - άξονα) είναι με mapped με τη τοποθεσία της ζώνης διέλευσης. Έτσι όπως παρατηρούμε και στο visual, το δεξί χέρι βρίσκεται σε μεγάλο εύρος ζώνης και στις υψηλές συχνότητες αφήνοντας έτσι να περνούν ένα ευρύ φάσμα υψηλών συχνοτήτων κόβοντας παράλληλα τις χαμηλές.

Μία ενδιαφέρουσα λειτουργία του mode αυτού είναι ότι μπορεί να λειτουργήσει σαν ένα είδους **crossfader** για την αλλαγή μεταξύ κομματιών. Καθώς το δεξί χέρι βρίσκεται τέρμα δεξιά αλλάζουμε κομμάτι και σταδιακά το επαναφέρουμε αριστερά για να περάσουν οι βασικές του συχνότητες. Με αυτή τη μέθοδο μπορούμε να κάνουμε ομαλές αλλαγές μεταξύ των κομματιών, σαν ένα εργαλείο DJ.

5.4 Σύνοψη Πρακτικού Μέρους

Σε αυτό το κεφάλαιο είδαμε αναλυτικά τα βήματα που ακολουθήσαμε για την δημιουργία και ολοκλήρωση της εφαρμογής. Ποια ήταν η αρχική ιδέα στο πρακτικό κομμάτι της πτυχιακής και πως αυτή εξελίχθηκε και πήρε την τελική μορφή της. Στη συνέχεια και καθώς περάσαμε στην επεξήγηση των διαφορετικών λειτουργιών της εφαρμογής, συμπεριλάβαμε τα πιο κρίσιμα κομμάτια του συνολικού κώδικα, παρουσιάζοντας τα και εξηγώντας τον τρόπο λειτουργίας τους. Καθώς η εφαρμογή εξελισσόταν το κεφάλαιο αυτό συμβάδιζε με τη πορεία αυτή, έτσι η προτεραιότητα στα διαφορετικά κεφάλαια είναι άμεσα συνδεδεμένη και με τη ροή της υλοποίησης. Στο παρακάτω κεφάλαιο το οποίο είναι και το τελευταίο, αναφέρουμε διαφορετικές προσθήκες του πρακτικού μέρους, που θα μπορούσαν να βελτιώσουν τη εφαρμογή σε πολλούς τομείς.

6. Τελικό μέρος πτυχιακής

Στο τελευταίο κεφάλαιο συνοψίζουμε τα συμπεράσματα του ερευνητικού και του πρακτικού μέρους της πτυχιακής και αξιολογούμε τη συνολική διαδικασία. Επίσης, προσθέτουμε νέα στοιχεία και ιδέες για μελλοντικές επεκτάσεις όσον αφορά την επίδοση και την πρακτικότητα της εφαρμογής.

6.1 Αποτελέσματα και συμπεράσματα

Η εφαρμογή είχε σαν σκοπό να δώσει στο χρήστη μία πλήρη εικόνα για το πως τα διαφορετικά χαρακτηριστικά ενός ήχου μπορούν να απεικονιστούν ,τι δεδομένα λαμβάνουμε από αυτή τη διαδικασία και πως μπορούμε να τα εκμεταλλευτούμε. Επίσης πώς εξυπηρετεί η εικονική αναπαράσταση του ήχου ερευνητικούς και καλλιτεχνικούς τομείς, με ποιες μεθόδους και ποιες τεχνικές.

Πέρα από το ερευνητικό μέρος που αναλύσαμε, το σκεπτικό πίσω από το σχεδιασμό αλγορίθμων πάνω στην εύρεση ρυθμού και των τεχνικών οπτικοποίησης, δημιουργήσαμε και μία εφαρμογή με βάση τα παραπάνω.

Διαπιστώσαμε ότι οι εφαρμογές πάνω στην απεικόνιση του ήχου σε πραγματικό χρόνο μπορεί να αποδειχθούν αρκετά επιβαρυντικές για την μνήμη, αν δεν διαχειριστούμε σωστά τα δεδομένα ενός ηχητικού κομματιού. Το μέγεθος του buffer(buffer size) και ο ρυθμός δειγματοληψίας(sample rate) του ψηφιακού μας ήχου πρέπει ρυθμιστούν έτσι ώστε να μην προκαλούν συμφόρηση κατά το output, όπως και να μην αλλοιώνουν κατά πολύ το ακουστικό αποτέλεσμα. Μπορεί το υψηλό buffer size να προσφέρει καλύτερη ποιότητα στον ήχο, αλλά είναι σημαντικό να γνωρίζουμε το κατά πόσο επιβαρύνει την εφαρμογή μας, γιατί μιλάμε για ψηφιακή ανάλυση σε πραγματικό χρόνο.

Μπορούμε να δημιουργήσουμε με μετασχηματισμό Fourier ελκυστικά και δημιουργικά αποτελέσματα, αρκετά εύκολα, αφού μας παρέχονται όλα έτοιμα μέσω συναρτήσεων. Το πρόβλημα όμως με τον FFT είναι ότι αποτελείται από μία σειρά πολύπλοκων υπολογισμών, που γίνονται ανά δευτερόλεπτο, γι αυτό θα πρέπει τα οπτικά που δημιουργούμε, βάση του μετασχηματισμού, να είναι απλά και να μην προκαλούν “κολλήματα”, glitches ή καθυστέρηση στη ροή του visual.

Όσον αφορά, τώρα, τους αλγόριθμους πάνω στη εύρεση ρυθμού που αναλύσαμε, είχαμε διαφορετικά αποτελέσματα, όπως θα περιμέναμε για διαφορετικά είδη μουσικής. Οι τεχνικές πάνω σ' αυτόν τον τομέα είναι αρκετά παραμετροποιήσιμες για να μπορούν να εφαρμόζονται σε κάθε μουσικό είδος. Τα αποτελέσματα που λάβαμε εμείς για την εφαρμογή τους πάνω στην ηλεκτρονική μουσική ήταν αρκετά ικανοποιητικά.

Τέλος με την δημιουργία της εφαρμογής διαπιστώσαμε το πόσο εύκολα μπορεί ένας χρήστης να δημιουργήσει ,με τη χρήση κατάλληλων εργαλείων, μία πληθώρα από visuals αρκεί να κατανοήσει συγκεκριμένες έννοιες πάνω στο ψηφιακό ήχο. Η γλώσσα προγραμματισμού processing σε σχέση με C# ή Java ,είναι ιδανική καθώς προσφέρει αμέτρητα εργαλεία πάνω στην ανάπτυξη πολυμεσικών εφαρμογών όπως επίσης και στον χειρισμό ηχητικών δεδομένων.

6.2 Επεκτάσεις, ιδέες, βελτιστοποίηση εφαρμογής

Η εφαρμογή από μόνη της περιέχει τα βασικά στοιχεία πάνω στη οπτικοποίηση ηχητικών δεδομένων πράγμα που σημαίνει ότι μπορούμε να εφαρμόσουμε ένα σημαντικό αριθμό επεκτάσεων όσον αφορά τη πολυπλοκότητα των οπτικών στοιχείων.

Ιδέες για νέα εικονικά στοιχεία σε δισδιάστατο χώρο

Random ellipses(με χρήση Beat detect):Τυχαίοι κύκλοι δημιουργούνται στο χώρο όταν ανιχνεύεται beat που και ανάλογα τη ζώνη συχνότητας του beat(ανάμεσα από τρεις ζώνες) θα έχει και την αντίστοιχη απόχρωση. Σε κάθε κύκλο εφαρμόζεται ένα σταδιακό fade που δίνει στο τελικό αποτέλεσμα μία ψυχεδελική αίσθηση. Το μέγεθος κάθε κύκλου στην οθόνη ορίζεται από την ένταση του beat.

Equalizer(με χρήση FFT):Δημιουργία δισδιάστατου equalizer με κλειστά χρώματα για χαμηλές ζώνες και ανοιχτά για υψηλές ζώνες.

Ιδέες για νέα εικονικά στοιχεία σε τρισδιάστατο χώρο και P3D renderer

3D Equalizer(με χρήση FFT): Δημιουργία equalizer σε τρισδιάστατο χώρο με την προσθήκη περιήγησης μέσα στο χώρο ή αυτόματη αλλαγή προκαθορισμένων λήψεων.

3D Star(με χρήση FFT):Ένα point στο χώρο όπου έχουμε δημιουργήσει γραμμές να ανοίγονται ξεκινώντας από αυτό και εφαρμόζονται σε αυτές ζώνες συχνοτήτων δίνοντας την έτσι την αίσθηση ενό αστεριού που τρεμοπαίζει.

Βελτίωση interface: Αρκετές βελτιώσεις χωράνε στο interface της εφαρμογής για να μπορέσουμε να πετύχουμε ένα πιο φιλικό και κατανοητό περιβάλλον προς το χρήστη. Καλύτερος σχεδιασμός των buttons ,πιο προσεγμένη επιλογή χρωμάτων για τα παράθυρα της εφαρμογής ή διαφορετική προσέγγιση όσον αφορά τη διαχείριση των modes.

Χειρισμός εικονικών στοιχείων σε πραγματικό χρόνο(manipulation mode)

Τέλος μία προσθήκη που θα μπορούσε να δώσει στην εφαρμογή μας ένα πιο δημιουργικό χαρακτήρα να δίνουμε τη δυνατότητα στο χρήστη να επέμβει ο ίδιος στο visual. Μέσα στο visual mode της εφαρμογής θα μπορεί ο χρήστης με το visual που θα είναι επιλεγμένο να το παραλλάσσει καθώς αυτό θα τρέχει. Αυτή η λειτουργία θα μπορούσε να προστεθεί και σαν διαφορετικό mode(manipulation mode) σε περίπτωση που θεωρηθεί αρκετά πολύπλοκο το visual mode. Χαρακτηριστικά όπως το μέγεθος χρώμα του ή το χρώμα του background στο visual θα αλλάζουν σύμφωνα με την τοποθεσία των χεριών του χρήστη.

Προσθήκη Παραγωγικών Εικονικών στοιχείων (generative visuals): Στο θεωρητικό μέρος εξηγήσαμε την έννοια των παραγωγικών visuals. Η προσθήκη ενός τέτοιου visual θα ήταν μία μεγάλη αναβάθμιση για την εφαρμογή καθώς στη συνέχεια θα μπορούσε πέρα από το κομμάτι να αναπτύσσεται και μέσω κινήσεων του χρήστη που θα λαμβάνονται από το kinect. Έτσι θα έχουμε ορίσει διαφορετικά στοιχεία να εξελίσσονται μέσω των συχνοτήτων των κομματιών και διαφορετικά στοιχεία μέσω των gestures του χρήστη δημιουργώντας έτσι ένα πλήρως διαδραστικό παραγωγικό concept.

Υποστήριξη της εφαρμογής για αριστερόχειρες

Κατά το ξεκίνημα της εφαρμογής θα θέτεται το ερώτημα στο χρήστη για το αν είναι αριστερόχειρας ή δεξιόχειρας. Κάθε επιλογή θα έχει τις δικές τις ρυθμίσεις που θα είναι πιο βολική ανάλογα με το χρήστη. Η εφαρμογή αυτής της προσθήκης είναι αρκετά εύκολη καθώς το μόνο που αλλάζει πέρα από την οθόνη εισαγωγής, θα είναι οι ακριβώς αντίθετες ρυθμίσεις της αρχικής.

7. Βιβλιογραφία

Aaron Koblin. (n.d.). Ανάκτηση από http://www.aaronkoblin.com/work/rh/: http://www.aaronkoblin.com/work/rh/

Borenstein, C. *Making things see.*

Fede, D. D. (n.d.). *compartmental*. Ανάκτηση από compartmental: http://code.compartmental.net/tools/minim/quickstart/

flight404. (n.d.). *www.flight404.com.* Ανάκτηση από http://www.flight404.com/blog/?p=143.

Fry & Reas, C. a. *Getting started with Processing.*

Miller, R. *Digital Art Painting with Pixels.*

Patin, F. (n.d.). Beat detection Algorithms.

Pramerdorfer, C. An Introduction to Processing and Music Visualization.

Rouse, M. (n.d.). *bandpass filter.* Ανάκτηση από http://whatis.techtarget.com/definition/bandpass-filter

Stearns, L. (n.d.). Introduction to Music Production(Coursera course).

wikipedia. (n.d.). Ανάκτηση από http://en.wikipedia.org/wiki/Digital_filter

ΦΙΛΤΡΑ, Τ. Λ. (n.d.). *ΦΙΛΤΡΑ ΜΕ ΠΑΘΗΤΙΚΑ ΣΤΟΙΧΕΙΑ.* Ανάκτηση από http://www.eln.teilam.gr/sites/default/files/10%20FILTRA.pdf

IST LAB 1st kinect workshop Consumer Depth sensors and Natural User Interaction

Links

http://code.compartmental.net/

http://www.matthiasdittrich.com/projekte/narratives/visualisation/

http://www.synthtopia.com/content/2008/10/18/

generative-music-visualization-radioheads-bodysnatchers/

http://www.aaronkoblin.com/work/rh/

http://vimeo.com/21773900

http://www.gamedev.net/page/resources/_/technical/math-and-physics/beat-detection-algorithms-r1952

http://shiffman.net/p5/kinect/

http://www.badlogicgames.com/wordpress/?p=99

http://www.youtube.com/watch?v=bFp2zZlFgv4

https://play.google.com/store/apps/details?id=com.psperl.projectM&hl=en

https://itunes.apple.com/en/app/xa1/id304070013?mt=8

http://glennmarshall.wordpress.com/2008/10/18/bodysnatchers-zeno-music-visualiser/

http://www.openprocessing.org/sketch/5989

http://technabob.com/blog/2007/08/24/atari-video-music-forgotten-1970s-tech/

http://www.flight404.com/blog/?p=143

http://code.compartmental.net/minim/examples/AudioEffect/

http://searchcio-midmarket.techtarget.com/definition/bandpass-filter

Wikipedia links

http://en.wikipedia.org/wiki/Digital_audio

http://en.wikipedia.org/wiki/Frequency_spectrum

http://en.wikipedia.org/wiki/Short-time_Fourier_transform

http://en.wikipedia.org/wiki/Music_visualization

http://en.wikipedia.org/wiki/Spectrogram

http://en.wikipedia.org/wiki/Media_player_(application_software)

http://el.wikipedia.org/wiki/Tempo

http://en.wikipedia.org/wiki/Processing_(programming_language)

http://en.wikipedia.org/wiki/Real-time_analyzer

http://en.wikipedia.org/wiki/Mapping

http://en.wikipedia.org/wiki/Interactive_art

http://en.wikipedia.org/wiki/New_media_art

http://en.wikipedia.org/wiki/Band-pass_filter

http://en.wikipedia.org/wiki/Electronic_filter

http://en.wikipedia.org/wiki/Analog_circuit

http://en.wikipedia.org/wiki/Digital_filter

el.wikipedia.org/wiki/Ψηφιακή_επεξεργασία_σήματος

el.wikipedia.org/wiki/Επεξεργασία_σήματος