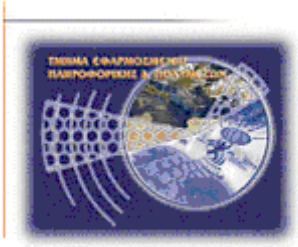
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων



Πτυχιακή εργασία

Τίτλος: Ανάπτυξη Πλατφόρμας Μετατροπής Τρισδιάστατης Εικόνας σε Ήχο
(3D Image-to-Sound Mapping)

Εμμανουήλ Ποτετσιανάκης (ΑΜ: 1698)

Επιβλέπων καθηγητής : Δρ. Γεώργιος Τριανταφυλλίδης

Επιτροπή Αξιολόγησης : Νικόλαος Βιδάκης
Τσαμπίκος Κουναλάκης

Ημερομηνία παρουσίασης: 30 Αυγούστου 2012

Ευχαριστίες

Για την πτυχιακή αυτή, θα ήθελα να ευχαριστήσω την Σοφία, για την συνεισφορά της στο πειραματικό μέρος της εργασίας. Τους γονείς μου και τον αδερφό μου, που ανέχτηκαν όλον αυτόν τον θόρυβο κατά τις δοκιμές. Και τον Ιωάννη για τις γενικές συμβουλές που προσέφερε και τα brainstorming sessions που είχαμε.

Abstract

This Diploma Thesis aims at creating an interactive multimedia application for real-time 3D image data-to-sound mapping. Utilizing a depth-sensor camera, the user, through the application can perform various sound-oriented actions.

To be more specific, there will be a sound synthesis mode, in which the data obtained from the sensor will be transformed into soundwaves. A second mode, will reproduce a sound signal, and after the 3D data mapping the signal will be sent to the output, edited according to the user's movements. The third -and final- mode is going to utilize the platform as an interpreter that will translate the user's movements into signals and then send them to an external audio program.

Throughout the analysis stage, various current audiovisual application development techniques will be studied and evaluated. Then, after the application study, new prospects will be presented. Prospects that are based and became feasible by this Thesis.

Despite the research-oriented nature of the project, the functions offered by the platform are entertaining and the environment plain and simple – accessible to users with no previous experience. This way, besides the new prospects that will be examined, we are going to have a complete multimedia application, ready to use. In order for the application to be even more accessible, the camera device selected is widely used (Microsoft's Kinect) and the programming language Processing, is platform indepented.

A demo video of the platform can be found at http://vimeo.com/convoitise/thesis

Σύνοψη

Η πτυχιακή αυτή αποσκοπεί στη δημιουργία μίας πολυμεσικής διαδραστικής εφαρμογής για μετατροπή -σε πραγματικό χρόνο- δεδομένων τρισδιάστατης εικόνας, σε ήχο. Χρησιμοποιώντας μια κάμερα με αισθητήρα βάθους, ο χρήστης, μέσω της εφαρμογής, μπορεί να πραγματοποιήσει διαφορετικές ενέργειες σχετικές με τον ήχο.

Πιο συγκεκριμένα θα υπάρξει μία λειτουργία σύνθεσης ήχου, στην οποία τα δεδομένα από τον αισθητήρα θα μετατρέπονται σε ηχητικά κύματα. Μία δεύτερη λειτουργία, θα δέχεται ένα ηχητικό σήμα και κατόπιν αντιστοίχησης των 3D δεδομένων θα έχει σαν έξοδο το σήμα, επεξεργασμένο, ανάλογα με τις κινήσεις του χρήστη. Η τρίτη και τελευταία λειτουργία που θα εξεταστεί, θα χρησιμοποιεί την πλατφόρμα σαν διερμηνέα, ο οποίος θα μεταφράζει τις κινήσεις του χρήστη και θα στέλνει αντίστοιχα σήματα σε εξωτερική εφαρμογή σχετική με ήχο.

Κατά τη διάρκεια του σταδίου της ανάλυσης, θα μελετηθούν και θα αξιολογηθούν διάφορες υπάρχουσες τεχνολογίες ανάπτυξης οπτικοακουστικών εφαρμογών. Κατόπιν, αφού γίνει η μελέτη της εφαρμογής, θα παρουσιαστούν νέες προοπτικές οι οποίες ξεκλειδώνονται, με βάση τη πτυχιακή αυτή εργασία.

Παρόλο που το θέμα είναι ερευνητικού χαρακτήρα, οι λειτουργίες της πλατφόρμας είναι ψυχαγωγικές και το περιβάλλον της απλό και λιτό - προσβάσιμο σε χρήστες με μηδενική εμπειρία. Κατ αυτόν τον τρόπο, πέρα από τις επικείμενες προοπτικές που θα αναλυθούν, έχουμε έτοιμη για χρήση μια ολοκληρωμένη πολυμεσική εφαρμογή. Για να γίνει η εφαρμογή ακόμη πιο εύκολα προσβάσιμη , η κάμερα η οποία επιλέχθηκε είναι ευρέως διαδεδομένη (Microsoft's Kinect) και η γλώσσα προγραμματισμού Processing, στην οποία έγινε η ανάπτυξη, ανεξάρτητη λειτουργικού (platform indepented).

Ένα βίντεο επίδειξης της τελικής πλατφόρμας υπάρχει εδώ: http://vimeo.com/convoitise/thesis

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Λίστα Πινάκων

1 Εισαγωγή

Ο τομέας της Πληροφορικής ασχολείται με την συλλογή, τη διαλογή, την επεξεργασία και την παρουσίαση των δεδομένων. Ένας από τους επικρατέστερους τρόπους για να το πετύχουμε αυτό είναι χρησιμοποιώντας Πολυμέσα, τα οποία αποτελούνται από εικόνες και οι ήχους.

Ο μέσος χρήστης έχει συνηθίσει το ένα μέσο να συνοδεύει ή να συμπληρώνει το άλλο. Σε αυτή τη πτυχιακή ξεδιπλώνονται νέες προοπτικές για την παρουσίαση της πληροφορίας και τα σύνορα ανάμεσα στην εικόνα και τον ήχο γίνονται λιγότερο διακριτά.

Χρησιμοποιώντας τεχνικές αντιστοίχισης (mapping) μετατρέπουμε την εικόνα (στην ψηφιακή μορφή της) σε ήχο. Η πτυχιακή αυτή εργασία ασχολείται με μία αφηρημένη μετατροπή, αλλά με βάση αυτή την εργασία, δεν αποκλείονται μελλοντικές υλοποιήσεις για τεχνικούς, καλλιτεχνικούς και επιστημονικούς σκοπούς.

1.1 Περίληψη

Σκοπός της πτυχιακής αυτής εργασίας είναι η κατασκευή μίας καινοτόμα πλατφόρμας για αντιστοίχηση δεδομένων 3D εικόνας σε ήχο. Για να πραγματοποιηθεί αυτό, χρειάζεται μία κάμερα με αισθητήρα βάθους. Η κάμερα που επιλέχθηκε γι αυτόν τον σκοπό είναι η Kinect της Microsoft και η επιλογή αυτή έγινε με βάση τη δημοτικότητα της κάμερας (και σε πωλήσεις, αλλά και σε υποστήριξη).

Στη συνέχεια επιλέχθηκε η Processing σαν γλώσσα προγραμματισμού, καθώς πρόκειται για μία γλώσσα η οποία έχει πολυμεσικό προσανατολισμό, ταχεία ανάπτυξη και βασίζεται πάνω στη πολύ σταθερή και εδραιωμένη αντικειμενοστραφή γλώσσα Java.

Για να επιτευχθεί η αντιστοίχηση, αρχικά γίνεται αναγνώριση Gestures του χρήστη, χρησιμοποιώντας τα δεδομένα βάθους. Στη συνέχεια ανάλογα με τη λειτουργία που έχει επιλεχθεί, γίνεται το mapping.

Οι λειτουργίες που θα προσφέρει η πλατφόρμα είναι τρεις. Η πρώτη είναι η λειτουργία σύνθεσης ήχου, στην οποία ο χρήστης δημιουργεί ήχο. Η δεύτερη επεξεργασίας ήχου, στην οποία ο χρήστης αλλοιώνει σε πραγματικό χρόνο, ένα προεπιλεγμένο ηχητικό απόσπασμα. Η τρίτη έχει να κάνει με τη χρήση της πλατφόρμας, σαν μέσο διασύνδεσης με εξωτερικές εφαρμογές εξειδεικευμένες σε λειτουργίες σχετιζόμενες με τον ήχο.

Όλα αυτά υλοποιούνται γύρω από έναν κοινό πυρήνα, κάτω από ένα απέριττο interface, με σκοπό την θέσπιση βάσεων για περαιτέρω εξέλιξη στον τομέα. Έτσι, αφού αναλυθούν οι λειτουργίες και οι προοπτικές της πλατφόρμας, θα υπάρξει και μία σύντομη εξέταση τυχών εναλλακτικών τρόπων προσέγγισης του θέματος.

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας

Σε προσωπικό επίπεδο, το κίνητρο για την επιλογή αυτού του θέματος ήταν η χρόνια ενασχόληση με το sound engineering, σε συνδυασμό με την επιθυμία μου για εξοικείωση με αλγορίθμους αντιστοίχισης. Επίσης τα τελευταία χρόνια έχω αναπτύξει ενδιαφέρον για αρχές αναγνώρισης προτύπων και τεχνητής όρασης. Τα παραπάνω, λαμβάνοντας υπ όψιν και τη ραγδαία ανάπτυξη των τεχνολογιών απεικόνισης χώρου, με οδήγησαν στην επιλογή του συγκεκριμένου θέματος. Η συγκεκριμένη πτυχιακή υλοποιήθηκε καθώς αποσκοπεί να δείξει μία νέα οπτική στον τρόπο αντιμετώπισης των Πολυμέσων.

1.3 Σκοπός και Στόχοι Εργασίας

Σκοπός αυτής της πτυχιακής είναι να προσδώσει μία "πλαστικότητα" στην διασύνδεση της εικόνας με τον ήχο. Σε αυτή την εργασία ο ήχος δεν αντιμετωπίζεται σαν μέσο καταγραφής/αναπαραγωγής της πληροφορίας, αλλά σαν μέσο μετατροπής αυτής. Έτσι ο χρήστης μπορεί πρακτικά να ακούσει τις κινήσεις του.

Στόχος της εργασίας είναι, μετά το πέρας αυτής, να υπάρχει μια θεμελιώδης πλατφόρμα αντιστοίχησης, η οποία θα θέτει βάσεις για περαιτέρω ανάπτυξη του τομέα διασύνδεσης εικόνας-ήχου.

1.4 Δομή Εργασίας

Για την υλοποίηση αυτής της πτυχιακής αρχικά θα αναλυθεί η μορφή της ψηφιακής εικόνας, συμπεριλαμβανομένων των δεδομένων τρισδιάστατης εικόνας, και στη συνέχεια του ψηφιακού ήχου. Κατόπιν θα μελετηθούν οι ήδη υπάρχοντες αλγόριθμοι που εξυπηρετούν τον ίδιο ή κάποιο αντίστοιχο σκοπό (state of the art). Στη συνέχεια θα σχεδιαστούν νέοι τρόποι αντιστοίχησης εικόνας σε ήχο. Θα αναλυθεί η αποδοτικότητα και η χρηστικότητα τους και στη συνέχεια θα επιλεχθούν οι καταλληλότεροι για την εργασία. Οι επιλαχόντες αλγόριθμοι θα υλοποιηθούν στη γλώσσα προγραμματισμού Processing και θα ενσωματωθούν σε μία ενιαία πλατφόρμα. Τέλος, θα γίνει και η πειραματική αξιολόγηση τους η οποία θα καταγραφεί στην τελική αναφορά.

2 Παρουσίαση Δομικών Συστατικών Δεδομένων

Παρακάτω παρουσιάζονται τα βασικά είδη πληροφοριών τα οποία χρησιμοποιεί η πλατόφρμα. Κυρίως είναι οι τύποι δεδομένων που έχουμε σαν είσοδο, για επεξεργασία ή σαν έξοδο. Η γνώση της δομής τους είναι ουσιώδες για την επεξήγηση του τρόπου λειτουργίας της πλατφόρμας.

2.1 Ψηφιακή Εικόνα

Στις ψηφιακές εικόνες, έχουμε δύο τύπους. Τις "διανυσματικές (vector)" ή αλλιώς δυναμικές και τις "στατικές (raster)". Εμείς θα ασχοληθούμε με τη δεύτερη κατηγορία εικόνων. Η επιλογή αυτή έγινε επειδή η μετατροπή δυναμικής σε στατικής εικόνας είναι αρκετά εύκολη, ενώ το αντίστροφο είναι τις περισσότερες φορές πολύ δύσκολο. Συν τις άλλοις, οι περισσότερες εικόνες που υπάρχουν σήμερα είναι στατικές και οι δυναμικές συνήθως εξυπηρετούν συγκεκριμένους σκοπούς.

Από εδώ και στο εξής λοιπόν, για ευκολία κατανόησης όταν αναφερόμαστε σε ψηφιακή εικόνα εννοείται ότι αυτή είναι στατική.

Κάθε ψηφιακή εικόνα είναι στην ουσία ένας πίνακας ο οποίος κβαντίζει την εικόνα σε ελάχιστες μονάδες εικόνας (εικονοστοιχεία - "pixels"). Κάθε εικονοστοιχείο λοιπόν απεικονίζει το μικρότερο δυνατό κλάσμα το οποίο μπορεί να απεικονιστεί, και ταυτόχρονα είναι ένα κελί (δεδομένων) του πίνακα με τα στοιχεία που απαρτίζουν αυτή την εικόνα.

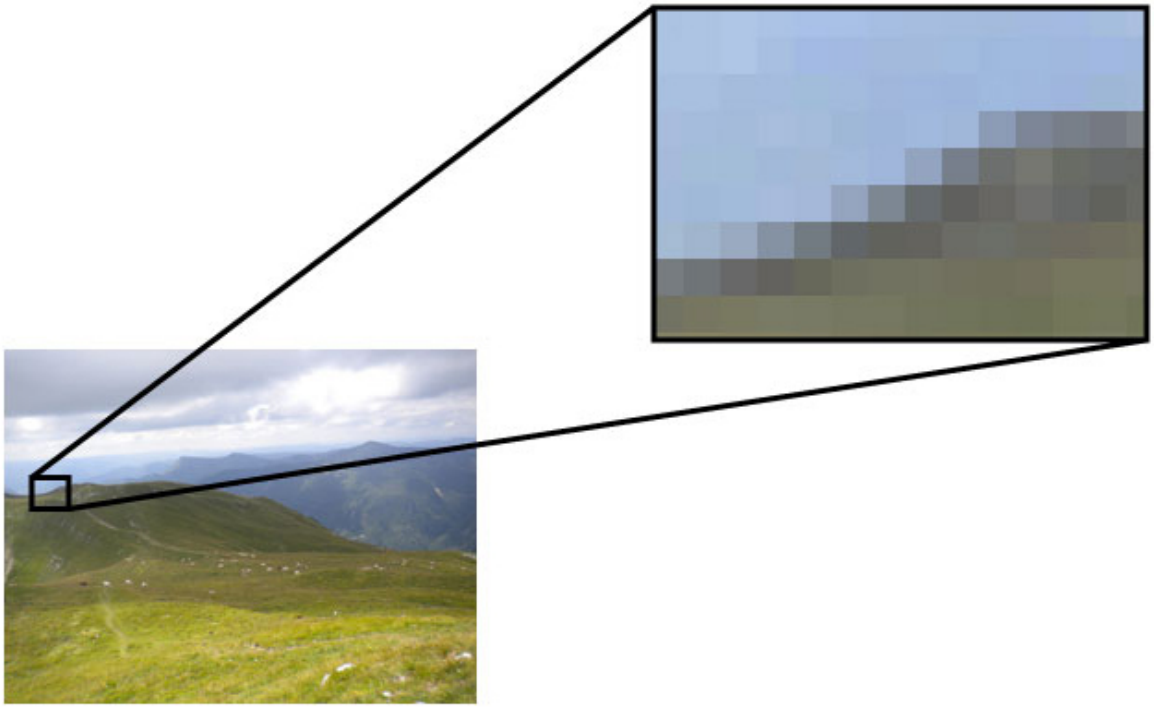
Όταν αναφερόμαστε στο μέγεθος, κάθε εικονοστοιχείου μπορεί να αναφερόμαστε στο εμβαδόν το οποίο καλύπτει αυτό στην εικόνα μας. Το οποίο δεν εξαρτάται από το εικονοστοιχείο αυτό καθ' αυτό, αλλά από την ανάλυση στην οποία απεικονίζεται η εικόνα μας. Για παράδειγμα μπορούμε να έχουμε μία εικόνα η οποία έχει 10 εικονοστοιχεία ανά μονάδα εμβαδού, και μία άλλη η οποία να έχει 100 εικονοστοιχεία ανά την ίδια μονάδα εμβαδού. Είναι προφανές λοιπόν ότι στη δεύτερη περίπτωση τα εικονοστοιχεία είναι 10 φορές περισσότερα και ταυτόχρονα έχουν 10 φορές μικρότερο εμβαδόν, άρα και ακριβέστερη πιστότητα εικόνας. Στην περίπτωση λοιπόν όπου αυτές οι εικόνες δεν έχουν κάποια μέθοδο συμπίεσης δεδομένων, είναι προφανές ότι η πρώτη εικόνα θα χρειαστεί τη 10πλάσια μνήμη για να αποθηκευτεί.

Συνήθως όμως όταν μας ενδιαφέρει το εμβαδόν του εικονοστοιχείου, χάριν ευκολίας, αναφέρεται η "ανάλυση". Είτε σαν συνολική ανάλυση εικόνας (image resolution) είτε σαν στοιχεία ανά ίντσα (pixels/dots per inch – ppi/dpi). Με την ανάλυση λοιπόν, έχουμε ουσιαστικά τον βαθμό λεπτομέρειας της εικόνας.

Φέρνοντας σαν παράδειγμα την παρακάτω εικόνα, έχουμε συνολικά εικονοστοιχεία: 256x192 pixels, με διαστάσεις εικόνας: 3.556x2.667inches. Άρα μπορούμε να συμπεράνουμε ότι η εικόνα είναι στα 72 dpi.

1: Sample Image

Παρακάτω παραθέτουμε την ίδια εικόνα, με ένα κομμάτι της μεγεθυνμένο κατά 10 φορές. Τα pixels σε αυτό το σημείο είναι ευδιάκριτα με γυμνό μάτι. Αυτό οφείλεται στο ότι η ανάλυση της εικόνας (άρα και η ακρίβεια της), είναι αντιστρόφως ανάλογη, της -τυχόν- μεγέθυνσης της.



2: Sample Image [x10]

Επειδή συνήθως ζητάμε την ανάλυση για την πιστότητα της εικόνας και όχι το εμβαδόν του εικονοστοιχείου, με το "μέγεθος εικονοστοιχείου", είναι πιο πιθανό να αναφερόμαστε στην μνήμη στην οποία καταλαμβάνει το κάθε εικονοστοιχείο για την αποθήκευση του. Συνήθως αναφέρεται σαν "βάθος χρώματος" (colour depth) με άτυπη μονάδα μέτρησης αυτού να είναι το bits per pixel (bpp).

Η αναφορά σαν "βάθος χρώματος" προκύπτει από την παραδοχή, ότι η πληροφορία η οποία αποθηκεύεται σε κάθε pixel, είναι η πληροφορία που χρειαζόμαστε να απεικονίσουμε το χρώμα το οποίο εμπεριέχει. Έτσι λοιπόν, αναπαράγοντας όλα τα εικονοστοιχεία με το σωστό τους χρώμα έκαστο, και τοποθετόντας τα στην σωστή σειρά/θέση, γίνεται η ανασύνθεση της εικόνας.

Για να δημιουργήσουμε την παλέτα χρωμάτων που χρειαζόμαστε για την αναπαράσταση κάποιας εικόνας, χρησιμοποιούμε συνδυασμούς τριών χρωμάτων: κόκκινο (Red), πράσινο (Green), μπλέ (Blue). Αυτή η μεθοδολογία απεικόνισης ονομάζεται RGB (από τα αρχικά των χρωμάτων).

Έτσι λοιπόν, ανάλογα με τα bits per pixel τα οποία διαθέτουμε, έχουμε και μέγεθος χρωματικής παλέτας. Ένας γενικός κανόνας είναι ότι για n bits έχουμε 2^n διαθέσιμα χρώματα. Παρακάτω αναφέρονται κάποια παραδείγματα.

Στην περίπτωση που έχουμε 1 bit για κάθε pixel είναι προφανές ότι, θα μπορέσουμε να αντιστοιχίσουμε σε 2 τιμές [0 και 1], οι οποίες θα είναι μαύρο και άσπρο αντίστοιχα. Παρακάτω έχουμε την εικόνα-δείγμα στην αντίστοιχη μορφή.

3: Sample Image [1bit Color Depth]

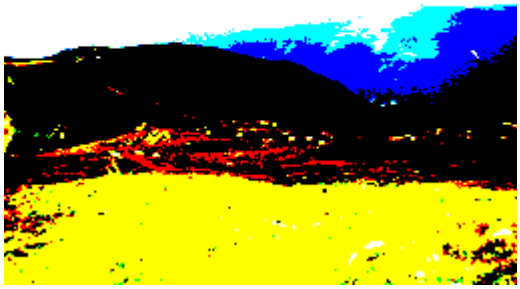
Εκ πρώτης όψης θα μπορούσε κάποιος εύκολα να υποθέσει ότι δεν είναι δυνατό να γίνει αναπαράσταση εικόνας με 1 μόνο bit ανά εικονοστοιχείο, εκτός και αν πρόκειται για κάποιο λιτό σκίτσο ή για κείμενο. Για να ξεπεραστεί αυτός ο τεχνικός περιορισμός απεικόνισης, οι ασπρόμαυρες εικόνες (καθώς και οι πρώιμες έγχρωμες) χρησιμοποιούσαν μια τεχνική η οποία αποκαλείται "dithering". Παρακάτω παραθέτουμε την εικόνα-δείγμα, ξανά σε αναπαράσταση μαύρου-άσπρου, αλλά με χρήση dithering.

4: Sample Image [1bit Color Depth - Dithering]

Παρατηρούμε αισθητή βελτίωση στην απεικόνιση. Αυτό ωφελείται σε τεχνικές χειραγώγησης της ανθρώπινης αντίληψης, κυρίως στην ανάμειξη λευκών εικονοστοιχείων με μαύρων, σε διάφορες αναλογίες, για επίτευξη διαφόρων "γκρι" αποχρώσεων. Ουσιαστικά αυτό που γίνεται είναι η ελεγχόμενη προσθήκη "θορύβου" στην εικόνα. Αλλά αυτό είναι κάτι που μπορούμε να το εξετάσουμε αργότερα.

Στα 2 bit, υπάρχει η ασπρόμαυρη αναπαράσταση (άσπρο, γκρι απόχρωση 1, γκρι απόχρωση 2 και μαύρο) καθώς και μία ψευδό-έγχρωμη αναπαράσταση, η οποία χρησιμοποιόταν παλιότερα (κυρίως από τη Mac και την ATARI).

Στα 3 bit, αρχίζουμε να έχουμε ουσιαστική έγχρωμη αναπαράσταση, καθώς έχουμε τη δυνατότητα να διαθέσουμε 1 bit για κάθε χρώμα και να πετύχουμε 8 συνολικές αποχρώσεις. Παρακάτω μπορούμε να δούμε μια αναπαράσταση της εικόνας σε 3 bit (με και χωρίς, dithering).



5: Sample Image [3bits per pixel]

6: Sample Image [3bpp - Dithering]

Αντίστοιχα λοιπόν είχαμε για 4bits (1 bit για κάθε χρώμα και άλλο ένα με χρήση "switch" φωτεινότητας), για 5 (2 bit για κάθε χρώμα εκτός του μπλε όπου έχει 1), για 6 (όπου έχουμε 64 συνολικά χρώματα).

Στα 8 λόγο της χρηστικότητας (8 bits = 1 byte) αναπτύχθηκαν διάφορες τεχνικές αναπαράστασης για αυτά τα 256 δυνατά χρώματα. Συμπεριλαμβανομένης φυσικά του μέχρι τώρα indexing, όπου είχαμε 3 bits για κόκκινο-πράσινο και 2 bits για μπλέ. Πολύ συνηθισμένη ήταν και η Web παλέτα (η οποία όμως είχε μόνο 216 χρώματα) και χρησιμοποιείται ακόμη και σήμερα σε κάποιες ειδικές περιπτώσεις.

7: Sample Image [8bit – Mac OS]

8: Sample Image [8bit – Windows]

9: Sample Image [8bit – Web]

10: Sample Image [8bit – Web - Dithering]

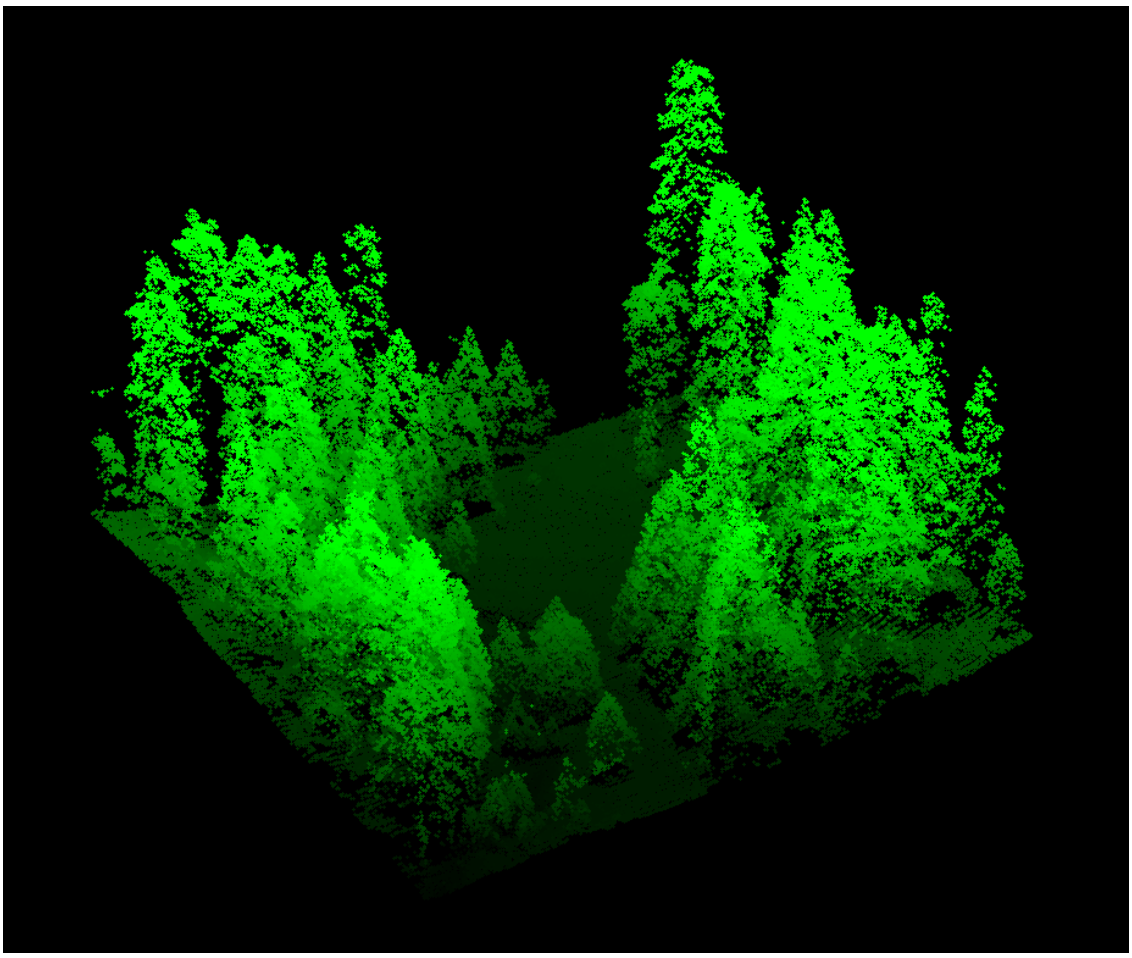
Στις μέρες μας η πιο διαδεδομένη μορφή είναι των 24bits [1byte ανά χρώμα] όπου έχουμε συνολικά 16.777.216 χρώματα. Η οποία επίσης αποκαλείται και "truecolor" καθώς πιστεύεται ότι το ανθρώπινο μάτι μπορεί να αναγνωρίσει μέχρι -περίπου- 10 εκατομμύρια χρώματα. Σε αυτό το βάθος χρώματος ήταν η αρχική μας εικόνα-δείγμα και αυτή τη ποιότητα εικόνας θα έχουμε σαν βάση στη συνέχεια.

Κάπου εδώ πρέπει να αναφερθεί ότι υπάρχουν και άλλα χρωματικά μοντέλα, όπως είναι για παράδειγμα το CMYK, αλλά σε αυτά θα αναφερθούμε εκτενέστερα αργότερα, καθώς τα χρησιμοποιούμε.

2.2 Τρισδιάστατη Ψηφιακή Εικόνα

Με τον όρο "τρισδιάστατη ψηφιακή εικόνα" αναφερόμαστε στα δεδομένα τα οποία συλλέγονται σχετικά με την θέση των αντικειμένων στον χώρο. Ο "πρέπον" όρος που θα έπρεπε να χρησιμοποιείτο θα ήταν "εικόνα βάθους" (depth map). Αυτό οφείλεται στο γεγονός ότι στη πραγματικότητα δεν έχουμε αποθηκευμένη μία τρισδιάστατη εικόνα αυτή καθ' αυτή, αλλά στοιχεία σχετικά με την απόσταση σημείων από τον αισθητήρα μας.

Έτσι λοιπόν, όπως στην ψηφιακή εικόνα έχουμε ουσιαστικά έναν δισδιάστατο πίνακα, στον οποίο σε κάθε θέση του έχουμε δεδομένα σχετικά με το χρώμα του κάθε στοιχείου, αντίστοιχα στην τρισδιάστατη εικόνα έχουμε έναν δισδιάστατο πίνακα στον οποίο κάθε στοιχείο του είναι ένα νούμερο το οποίο αντιπροσωπεύει την απόσταση του σημείου από τον αισθητήρα/κάμερα (point cloud). Στην εικόνα παρακάτω μπορούμε να δούμε μία απεικόνιση ενός point cloud από μία LIDAR κάμερα.



11: Point Cloud Visualization

Πέραν αυτού υπάρχουν και άλλες ομοιότητες με την ψηφιακή εικόνα. Για παράδειγμα, όπως το μέγεθος του πίνακα είναι άρρητα συνδεδεμένο με την ποιότητα/μέγεθος της εικόνας, έτσι και όταν μιλάμε για έναν depth map όσο μεγαλύτερος είναι ο πίνακας, τόσο περισσότερα στοιχεία έχουμε. Επίσης ανάλογα με το πόσα bits αφιερώνουμε σε κάθε στοιχείο του πίνακα τόσο μεγαλύτερη ακρίβεια μπορούμε να έχουμε στην καταγραφή των στοιχείων.

Μία ειδοποιός διαφορά μεταξύ των δύο τύπων εικόνων είναι ότι ενώ στην χρωματική εικόνα, το εύρος τιμών που μπορεί να έχει ένα στοιχείο είναι η αντιστοίχηση του στο εύρος συχνοτήτων του ορατού φωτός, ενώ στην εικόνα με τρισδιάστατα δεδομένα ένα στοιχείο μπορεί να έχει τιμές αντίστοιχες με της δυνατότητες του αισθητήρα μας. Για παράδειγμα ο αισθητήρας της συσκευής Kinect -την οποία χρησιμοποιούμε- μπορεί να καταγράψει τιμές για αντικείμενα τα οποία βρίσκονται σε απόσταση μεγαλύτερη των 120cm και μικρότερη των 350cm.

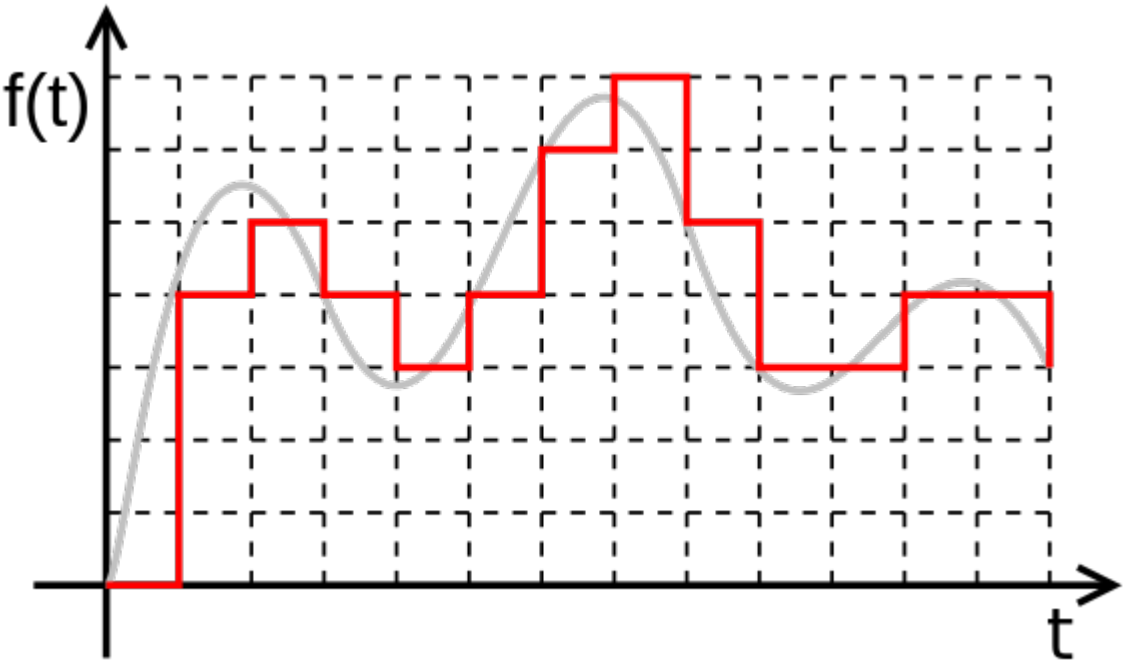
2.3 Ψηφιακός Ήχος

Ως γνωστό, ο ήχος προκαλείται είτε από ξαφνική διαφορά πυκνότητας (π.χ. έκρηξη βόμβας) σε ένα μέσο, είτε από ταλαντώσεις κάποιας επιφάνειας (μεμβράνη ηχείου, φωνητικές χορδές κλπ), και κατόπιν μεταδίδεται σε κύματα δια κάποιου μέσου (αέριο, στερεό, υγρό), και γίνεται αντιληπτός από τις ταλαντώσεις τις οποίες αυτά προκαλούν (σε μικρόφωνο, στο ακουστικό τύμπανο κλπ). Το ανθρώπινο αυτί μπορεί να αντιληφθεί κύματα συχνότητας από 20Hz μέχρι 20000Hz (κατά προσέγγιση).

Για να καταφέρουμε να καταγράψουμε (και στη συνέχεια να αναπαράγουμε) τον ήχο είναι

προφανές ότι θα πρέπει να καταγράψουμε τα κύματα τα οποία αυτός δημιουργεί. Για να γίνει αυτό με αναλογικό τρόπο είναι -σχετικά- εύκολο, καθώς μπορούμε να αναπαραστήσουμε την -συνεχή- ταλάντωση που αυτά δημιουργούν (π.χ. δίσκος βινυλίου). Επειδή όμως στον ψηφιακό κόσμο δεν είναι δυνατό να υπάρξει συνέχεια, όπως τον αναλογικό, ο ήχος πρέπει να κβαντιστεί σε διακριτές τιμές για να μπορέσει να καταγραφεί.

Η λογική με την οποία αυτό γίνεται είναι αντίστοιχη με την ψηφιακή εικόνα. Προσπαθούμε να χωρίσουμε το δείγμα μας σε όσον το δυνατόν περισσότερα "κομμάτια" (samples), για να πετύχουμε την ψευδαίσθηση της συνέχειας, και αυτά τα "κομμάτια" να είναι αντιστοιχισμένα σε όσο το δυνατόν περισσότερο ακριβή τιμές για να υπάρχει πιστότητα κατά την αναπαραγωγή. Αυτό έχει σαν αποτέλεσμα, όταν θελήσουμε να αναπαράγουμε τον ήχο ο οποίος έχει καταγραφεί/δημιουργηθεί να παράγουμε τους ήχους που αντιστοιχίζονται σε αυτά τα samples αρκετά γρήγορα.



12: Αναλογικό & Ψηφιακό Ηχητικό Σήμα

Στην εικόνα παραπάνω μπορούμε να δούμε ένα αναλογικό σήμα (με γκρι) και πως θα ήταν το αντίστοιχο ψηφιακό (με κόκκινο). Παρατηρούμε ότι για κάθε δείγμα θα χρειαζόμασταν 3bits και ότι το συνολικό σήμα είναι χωρισμένο σε 13 δείγματα. Ο ρυθμός με τον οποίο γίνεται η καταγραφή/αναπαραγωγή ψηφιακών ηχητικών σημάτων συνηθίζεται να είναι 44100Hz, υπακούοντας στη συνθήκη Niquist.

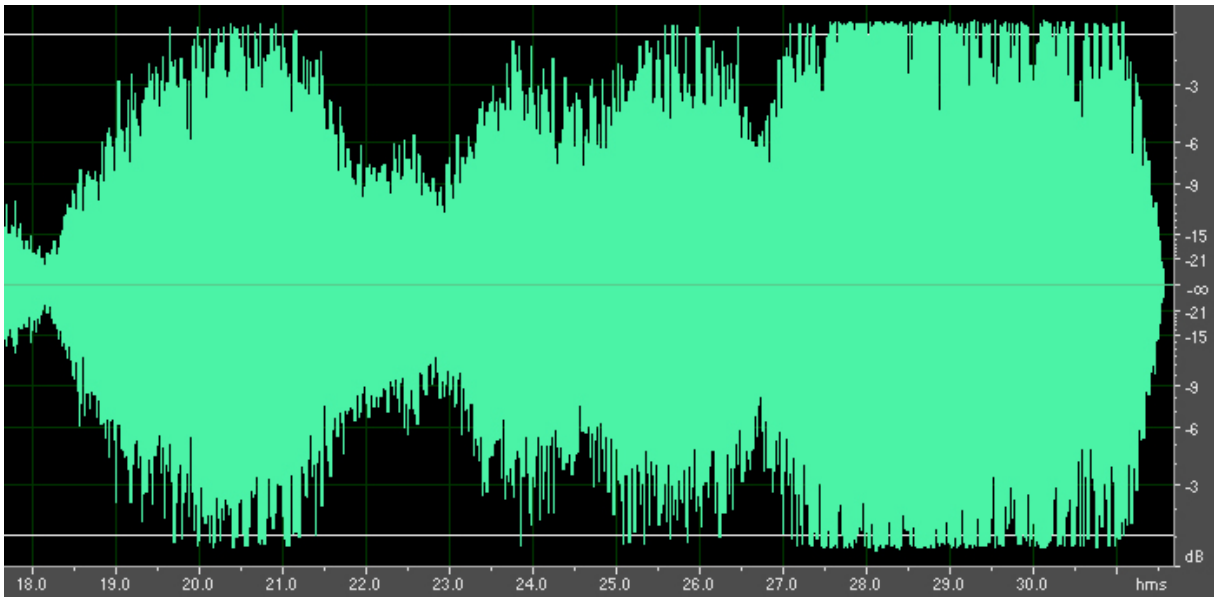
3 Σχέδιο Δράσης Για Την Εκπόνηση Της Εργασίας

Σε αυτό το κεφάλαιο, θα αναλύσουμε ήδη υπάρχουσες τεχνολογίες στον χώρο και θα επεξηγηθεί ο συγκεκριμένος τρόπος προσέγγισης της πτυχιακής.

3.1 State of The Art

Στο τομέα της αντιστοίχισης εικόνας-ήχου αυτή τη στιγμή υπάρχουν αρκετές εφαρμογές. Η πιο γνωστή είναι η απεικόνιση της κυματομορφής (waveform) του ήχου, την οποία κατά πάσα πιθανότητα έχουμε συναντήσει όλοι κάπου. Αυτό που κάνει αυτη η εφαρμογή είναι να αναπαριστά τον ήχο (ως προς το χρόνο) σχηματίζοντας σε εικόνα την κυματική μορφή που θα είχε ο ήχος καθώς αναπαράγεται.

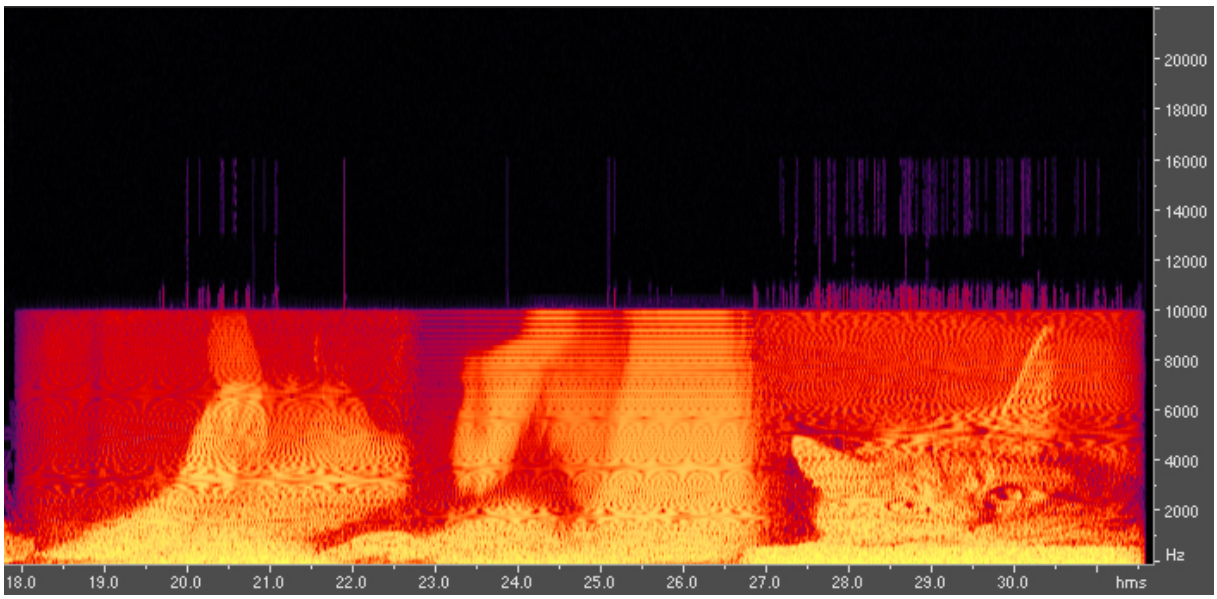
Όπως βλέπουμε και στην παρακάτω εικόνα στον κάθετο άξονα έχουμε την ένταση (σε dB) του ήχου, ως προς τον χρόνο (οριζόντιος άξονας).



13: Waveform of "Look" by Venetian Snares [18"-31"]

Ένας άλλος τρόπος αναπαράστασης του ήχου, ο οποίος είναι κάπως πιο εξειδικευμένος είναι η αναπαράσταση της φασματικής ανάλυσης του ήχου (spectrogram). Με αυτό τον τρόπο, ο ήχος περνάει μέσα από Φασματική Ανάλυση Fourier και αυτό που προκύπτει είναι μία απεικόνηση του φάσματος του.

Η παρακάτω εικόνα έχει το ίδιο δείγμα ήχου σε φασματική αναπαράσταση, δηλαδή στον κάθετο άξονα έχουμε συχνότητα (Hz).



14: Spectrogram of "Look" by Venetian Snares [18"-31"]

Υπάρχουν και άλλοι τρόποι για μετατροπή ήχου σε εικόνα, όχι όμως τόσο ευρέως γνωστοί και αρκετά εξειδικευμένοι. Για παράδειγμα, υπάρχουν προγράμματα τα οποία εμφανίζουν την διαφορά της φασματικής ανάλυσης (αντί για την απόλυτη τιμή του φάσματος) ή εξομοιώνουν την πραγματική ένταση του ήχου (αποκόπτουν συχνότητες που δεν είναι αντιλήψημες στον άνθρωπο για ψυχοακουστικούς λόγους).

Στον τομέα της μετατροπής εικόνας-σε-ήχο, τα πράγματα είναι λίγο πιο σύνθετα. Παίρνοντας το φάσματογράφημα του "Look" από τον καλλιτέχνη Venetian Snares (εικόνα b) παρατηρούμε ότι εμφανίζονται κάποιες γάτες. Στο συγκεκριμένο παράδειγμα, αυτός ήταν ο σκοπός. Έτσι, ο καλλιτέχνης χρησιμοποιόντας μία ασπρόμαυρη εικόνα από τα κατοικίδια του σαν φασματογράφημα, έκανε την αντίστροφη μετατροπή και παρήγαγε τον ήχο που θα είχε αυτό το φάσμα.

Ο αλγόριθμος μετατροπής είναι πολύ απλός, καθώς αναλύει τη φωτογραφία και τη δέχεται σαν φασματογράφημα. Στη συνέχεια αντιστρέφοντας τη φωτεινότητα της την περνάει σαν είσοδο. Έτσι, στα σημεία που εμφανίζεται κάτι στη φωτογραφία έχουμε μία εικονική φασματική τιμή η οποία με αντίστροφη μετατροπή Fourier πάει στην έξοδο σαν ήχος.

Αυτή λοιπόν είναι η πιο διαδεδομένη μέθοδος mapping από εικόνα σε ήχο (επιπλέον καλλιτέχνες που το έχουν υλοποιήσει σε κομμάτια τους είναι Nine Inch Nails, Aphex Twin, Francisco Lopez κ.α.).

Το πρόβλημα με αυτή τη μέθοδο όμως είναι ότι αυτό που γίνεται δεν είναι ακριβώς αντιστοίχηση. Από τη μία έχουμε το στοιχείο ότι κάθε δείγμα της φωτογραφιας αντιστοιχεί σε ένα μοναδικό φάσμα αρα και σε έναν μοναδικό ήχο, αλλά η αντιστοίχιση γίνεται με συγκεκριμένες φωτογραφίες, έχοντας περιορισμένες δυνατότητες, καλύπτοντας μικρό εύρος αποτελέσματος και εισόδου. Το αποτέλεσμα είναι ότι έχουμε μία μετατροπή για χάρη της μετατροπής και όχι ένα εργαλείο αντιστοίχισης.

Στό ίδιο κομμάτι όμως δε θα μπορούσε να γίνει η μετατροπή χρησιμοποιόντας τη κυματομορφή του (εικόνα α). Αυτό οφείλεται στο γεγονός ότι σαν δεδομένα έχουμε την συγκεκριμένη τιμή της έντασης του ήχου για κάθε δεδομένη στιγμή. Η ένταση όμως δεν είναι ένα μοναδικό χαρακτηριστικό του ήχου. Έτσι δε θα μπορούσε να υπάρξει ποτέ κάποια πλατφόρμα μετατροπής εικόνας σε ήχο χρησιμοποιόντας την εικόνα σαν κυματομορφή (και μόνο).

Άλλες πλατφόρμες μετατροπής εικόνας σε ήχο, έχουν ακόμη πιο θολή την ένοια του "mapping". Για παράδειγμα το πρόγραμμα RGB MusicLab, παίρνει μία εικόνα και αποσυνθέτοντας τη στα τρία βασικά χρώματα (Κόκκινο, Πράσινο, Μπλε) δημιουργεί μία μουσική κλίμακα την οποία και αναπαράγει. Έαν λόγου χάρη είχαμε σε ένα pixel μια RGB τιμή 120 τότε θα είχαμε τη νότα "Ντο" και ανάλογα με τη φωτεινότητα του, θα είχαμε τη διάρκεια. Εαν είχαμε R = 0, G = 0, B = 0 δεν θα υπήρχε

κάποιος ήχος.

Βλέπουμε λοιπόν ότι αυτό το πρόγραμμα εξυπηρετεί έναν συγκεκριμένο σκοπό. Ο οποίος είναι η δημιουργία ήχων από χρωματικές κλίμακες.

Μία άλλη εφαρμογή (η οποία δουλεύει μόνο με ασπρόμαυρες εικόνες) είναι η vOICe η οποία δέχεται μια εικόνα σαν είσοδο και η αντιστοίχιση γίνεται ως εξής:

1. Η κάθετη θέση του pixel μετατρέπεται σε συχνότητα
2. Η οριζόντια σε στιγμή (ουσιαστικά έτσι καθορίζεται η σειρά με την οποία αναπαράγονται τα δείγματα)
3. Η φωτεινότητα σε πλάτος συχνότητας (στον ήχο αντιστοιχεί σε ένταση).

Αυτή η εφαρμογή (παρόλο που δουλεύει μόνο για ασπρόμαυρες εικόνες) παράγει ένα πιο χαρακτηριστικό παράδειγμα αντιστοίχισης, καθώς κάθε pixel είναι μοναδικό, έχουμε δεδομένα για τη σειρά των pixel και εκτός από τη συχνότητα έχουμε και το πλάτος αυτής.

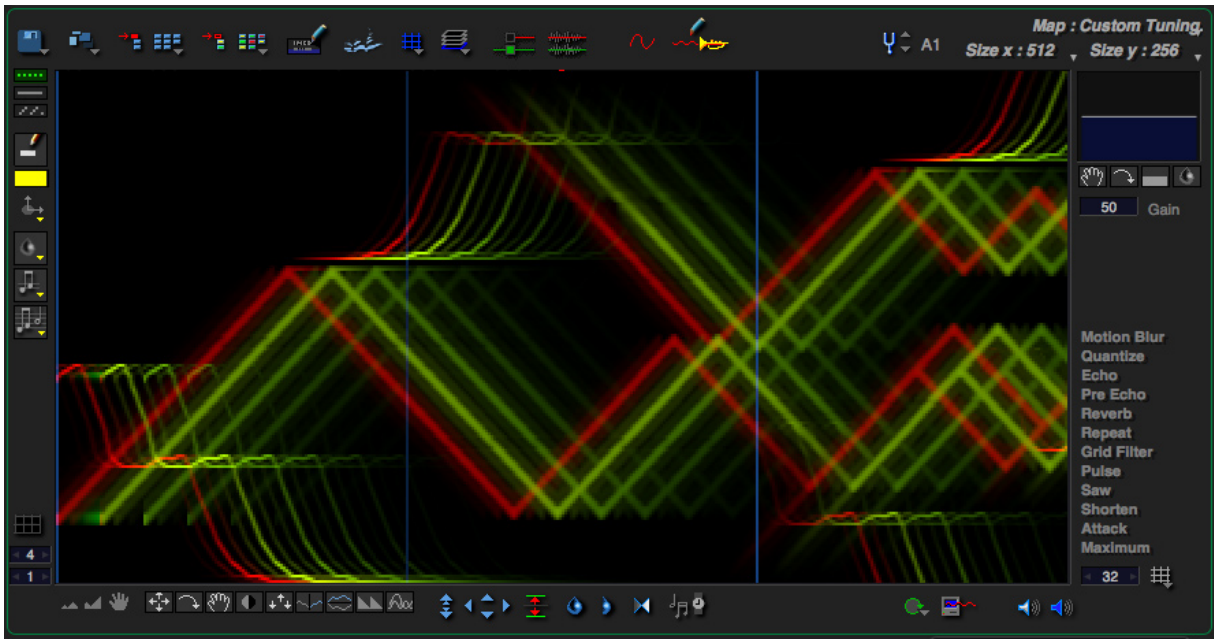
Παραταύτα, το αποτέλεσμα και οι δυνατότητες του προγράμματος, δε διαφέρουν και πολύ από ότι εαν κάναμε μια απλή αντιστροφή φασματογράμματος.

Η πιο διαδεδομένη ίσως ολοκληρωμένη πλατφόρμα η οποία κάνει πραγματική αντιστοίχηση εικόνας σε ήχο pixel-by-pixel είναι η MetaSynth της UI Software.

Στο κεντρικό panel της εφαρμογής (εικόνα c) έχουμε ένα σχεδιαστικό πρόγραμμα στο οποίο ζωγραφίζουμε την εικόνα που επιθυμούμε. Στη συνέχεια γίνεται η μετατροπή της σε ήχο.

Η μηχανή αντιστοίχησης δουλεύει με τις εξής παραδοχές:

1. Η οριζόντια θέση του εικονοστοιχείου αντιστοιχεί στον χρόνο (σειρά) στην οποία θα τοποθετηθεί το δείγμα μας.
2. Η κάθετη θέση αντιπροσωπεύει τη χροιά (pitch) που θα έχει ο παραγόμενος ήχος.
3. Η εφαρμογή παράγει stereo δείγματα και το χρώμα του εικονοστοιχείου αντιστοιχεί στην θέση του στον χώρο (stereo balance).
4. Τέλος, η φωτεινότητα του είναι κατ' αναλογία η ένταση η οποία θα έχει το παραχθέν δείγμα.



15: Στιγμιότυπο από το περιβάλλον εργασίας του MetaSynth

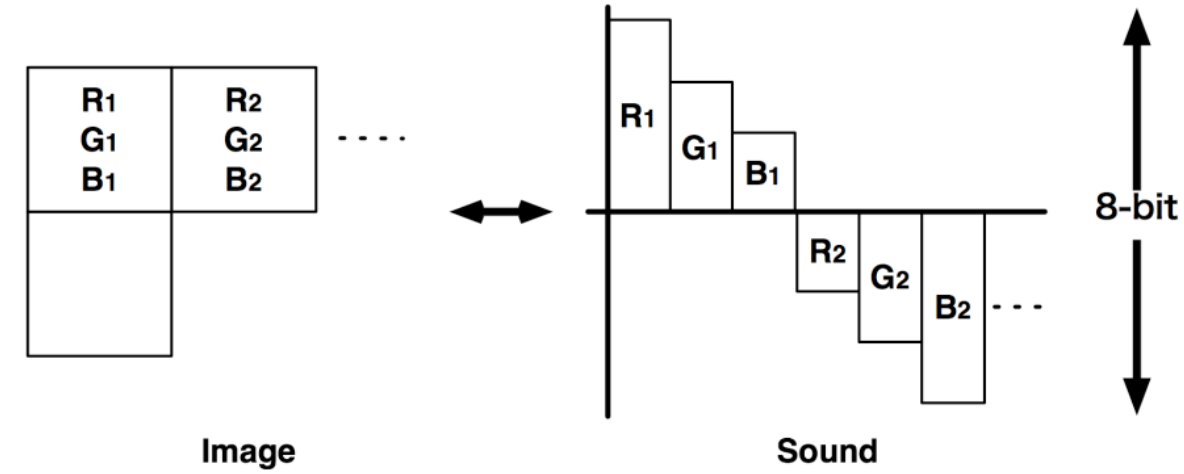
Μία λιγότερο γνωστή εφαρμογή η οποία κάνει και αυτή αντιστοίχηση είναι η Monalisa. Αυτή η εφαρμογή (υπάρχει μόνο για OS X και iOS) είναι βασισμένη πάνω στο λογισμικό Max/MSP της Cycling 74 και στο UPIC του Ιωάννη Ξενάκη.

Η εφαρμογή αυτή λειτουργεί με δύο τρόπους.

Στον πρώτο, αντλεί από κάθε εικονοστοιχείο τις τιμές του κάθε καναλιού χρώματος χωριστά και τις τριάδες των 8bit που αντλεί τις μετατρέπει σε 3άδες ηχητικών δειγμάτων των 8bit και

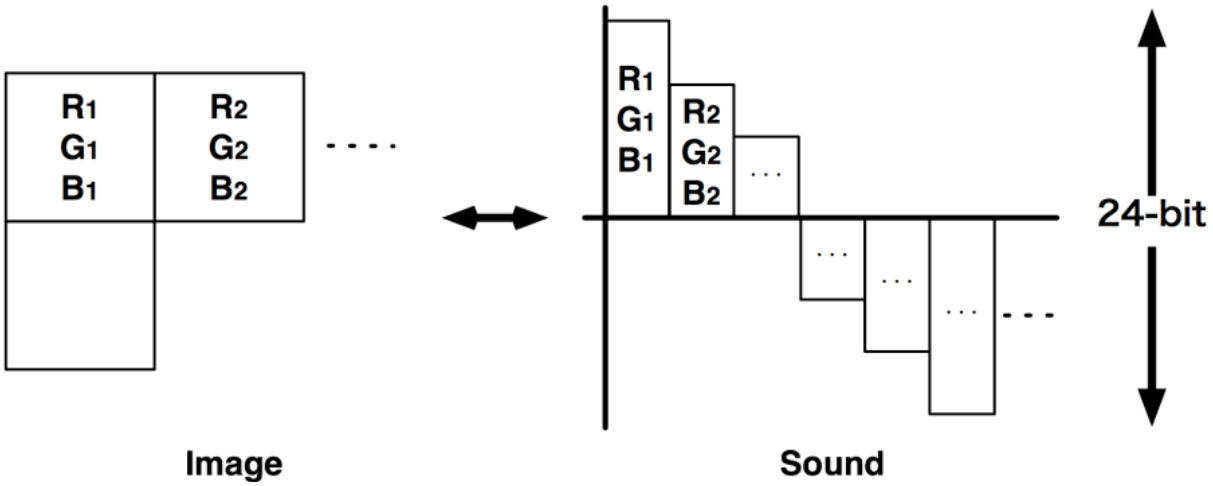
αντίστροφα (εικόνα d).

Όπως μπορούμε να συμπεράνουμε σε αυτή τη λειτουργία η πλατφόρμα έχει σαν είσοδο εικόνες με βάθος χρώματος 24bit και έξοδο ήχο με πιστότητα 8bit. Αντίστοιχα συμπεραίνουμε ότι τα δείγματα του ήχου εξόδου είναι 3 φορές περισσότερα από το πλήθος των εικονοστοιχείων της εικόνας εισόδου.



16: Monalisa (8bit function)

Κατά τον δεύτερο τρόπο λειτουργίας της πλατφόρμας, η πληροφορία που αντλείται από κάθε εικονοστοιχείο μεταφράζεται σε πληροφορία κάθε δείγματος. Πιο αναλυτικά, η κάθε 3αδα από 8bit του κάθε εικονοστοιχείου μεταφράζεται σε 1 ηχητικό δείγμα των 24bit (εικόνα e).



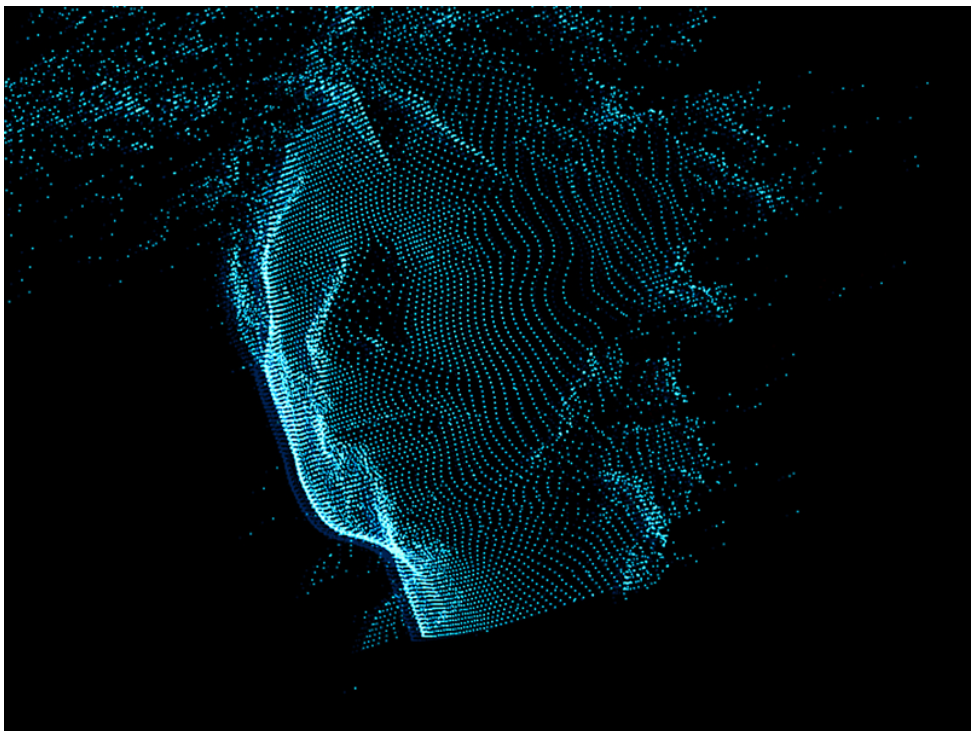
17: Monalisa (24bit function)

Παρατηρούμε εδώ λοιπόν ότι κάθε εικονοστοιχείο (με βάθος χρώματος 24bit) αντιστοιχίζεται σε ένα ακριβώς δείγμα ήχου (με πιστότητα 24bit). Άρα έχουμε τον ίδιο αριθμό ηχητικών δειγμάτων και εικονοστοιχείων.

Όσον αφορά την τρισδιάστατη εικόνα και τη μετατροπή της σε ήχο, οι σχετικοί υπάρχοντες αλγόριθμοι είναι ελάχιστοι, οι περισσότεροι εκ των οποίον είναι σε στάδιο ανάπτυξης. Μέχρι στιγμής δε μπορεί κάποιος να μιλάει για ακριβή αντιστοίχηση.

Το πιο δημοφιλές παράδειγμα μέχρι στιγμής, είναι η εφαρμογή (επίσης γραμμένη σε Processing) η οποία δημιουργήθηκε από το συγκρότημα Radiohead για την εξυπηρέτηση αναπαράστασης τρισδιάστατων δεδομένων αποσκοπώντας στη δημιουργία του μουσικού βίντεο για το τραγούδι τους House of Cards (από τον δίσκο τους In Rainbows). Αυτό όμως δεν πρόκειται για μία

εφαρμογή αντιστοίχησης, καθώς έγινε η απεικόνιση μέσω του προγράμματος και ο συγχρονισμός της εικόνας με τον ήχο έγινε εξωτερικά και σε δεύτερο χρόνο.



18: Thom Yorke (Radiohead) – House of Cards Instance

Άλλα γνωστά παραδείγματα – ολοκληρωμένες εφαρμογές δεν υπάρχουν. Μετά από αρκετή έρευνα εντοπίστηκαν κάποια παρεμφερή, όπως για παράδειγμα μία πλατφόρμα (επίσης σε Processing) η οποία αναγνωρίζει κίνηση μέσω ενός επιταχυντή τον οποίο κρατά ο χρήστης και γίνεται mapping της κίνησης αυτής ούτως ώστε να κινηθεί ένας ρομποτικός βραχίονας και να πατήσει κάποια πλήκτρα που παράγουν ήχο,·η όλη διαδικασία απεικονίζεται σε μία οθόνη χρησιμοποιώντας μία κάμερα. Αυτό πρόκειται για μέρος διπλωματικής μίας φοιτήτριας, και όπως μπορούμε να συμπεράνουμε δεν εμπίπτει στην σφαίρα του 3D Image-to-Sound mapping.

Πρέπει να σημειωθεί πάραυτα ότι παρότι διαφαίνεται απουσία εφαρμογών σε αυτόν τον τομέα, υπάρχει έργο το οποίο είναι σε εξέλιξη. Επιπλέον, ήδη υπάρχουσες πολυμεσικές πλατφόρμες -όπως για παράδειγμα το IanniX- μπορούν να διαμορφωθούν ή να διασυνδεθούν με άλλες πλατφόρμες ούτως ώστε να στηρίξουν την ανάπτυξη τέτοιων εφαρμογών.

3.2 Σημαντικοί Στόχοι Για την Ολοκλήρωση της Πτυχιακής

Ακολουθεί μία λίστα με τους συνολικούς στόχους για την επίτευξη της πτυχιακής εργασίας με τη βαρύτητα του καθενός χωριστά (σε ποσοστό επί τοις εκατό) ως προς το χρόνο και την προσωπική εργασία που εκτιμάται ότι θα απαιτήσουν.

- Έρευνα state of the art 5
- Αναζήτηση υλοποιημένων τεχνικών 5
- Ανάλυση Ψηφιακής Εικόνας και Ψηφιακού Ήχου 10
- Καταγραφή προτεινώμενων αλγορίθμων αντιστοίχισης 10
- Αξιολόγιση (a-prior) και επιλογή των πιο κατάλληλων 5

- Υλοποίηση των επιλεγμένων αλγορίθμων αντιστοίχισης 20
- Αξιολόγιση (a-posterior) και επιλογή των τελικών αλγορίθμων 5
- Βελτιστοποίηση αυτών 5
- Ενσωμάτωση τους σε μία ενιαία πλατφόρμα 15
- Έλεγχος σφαλμάτων της συνολικής πλατφόρμας 5
- Καταγραφή μετρήσεων/αποτελεσμάτων 5
- Συγγραφή Αναφοράς 5
- Δημιουργία Τελικής Παρουσίασης 5

4 Κύριο Μέρος Πτυχιακής Μέρος Πτυχιακής (Requirement Analysis)

Σε αυτό το κεφάλαιο αρχικά θα γίνει μία ανάλυση του στόχου που προσπαθούμε να πετύχουμε, στη συνέχεια μια παρουσίαση του τρόπου προσέγγισης και τέλος θα παρουσιαστεί ή πραγματοποίηση αυτού.

4.1 Ανάλυση Προβλήματος

Σκοπός αυτής της πτυχιακής είναι η αντιστοίχιση πληροφοριών τρισδιάστατης εικόνας σε ήχο. Εξ ορισμού, αντιστοίχιση σημαίνει ότι έχεις κάποια δεδομένα εισόδου (οπτικά) και τα μετατρέπεις σε κάποια δεδομένα εξόδου άλλης μορφής (ηχητικά). Αρχικά λοιπόν θα πρέπει να φέρουμε την εικόνα -η οποία αποτελεί την είσοδο στην πλατφόρμα- σε κάποια/ες μορφή/ες όπου εξυπηρετούν την αντιστοίχιση.

Μέχρι τώρα δόθηκε έμφαση στο χρωματικό μοντέλο RGB. Αυτό όμως έγινε κυρίως επειδή οι υπάρχουσες εφαρμογές λειτουργούν με εικόνες 2-διαστάσεων. Στην προκειμένη πτυχιακή εργασία, θα χρησιμοποιήσουμε κυρίως δεδομένα τρισδιάστατης εικόνας για την αντιστοίχηση, και τα χρωματικά μοντέλα θα μελετηθούν σε θεωρητικό επίπεδο. Έτσι λοιπόν σαν είσοδο θα έχουμε μια μηχανή η οποία θα αναγνωρίζει τις κινήσεις (gestures) του χρήστη.

Στην έξοδο έχουμε ηχητικά δεδομένα. Για να ανακτήσουμε αυτά, θα πρέπει να "συναρμολογήσουμε" τα δείγματα (samples) τα οποία θα είναι και το προϊόν της αντιστοίχισης. Στη συγκεκριμένη φάση ανάπτυξης, η εφαρμογή παρέχει πιστότητα δείγματος 1024bits με συχνότητα 44100Hz. Σε περίπτωση που επιθυμούσαμε να ηχογραφήσουμε τα ηχητικά αποτελέσματα. η ηχητική μορφή εξόδου η οποία θα επιδιώκαμε να παράγουμε θα ήταν *.wav, καθώς πρόκειται για μορφή με χαρακτηριστικά τα οποία ταιριάζουν στην παρούσα έξοδο και χωρίς ψυχοακουστικές απώλειες (λόγω συμπίεσης) οι οποίες όταν μιλάμε για δεδομένα -και όχι για τέχνη- είναι ανεπίτρεπτες.

Καταλήγουμε λοιπόν ότι το πρώτο βήμα θα είναι να μετρήσουμε το εύρος των επιλογών μας στην είσοδο. Στην συνέχεια θα υπολογίσουμε το πλήθος των διαφορετικών samples που θα μπορούσαμε να εξάγουμε. Κατόπιν θα δημιουργήσουμε την μηχανή αντιστοίχησης η οποία θα είναι και η "καρδιά" της πλατφόρμας με γνώμονα τις Απαιτήσεις Συστήματος (βλέπε επόμενη ενότητα). Τέλος, θα γίνει η υλοποίηση της μηχανής γύρω από την οποία θα έχουμε την πλατφόρμα.

4.2 Απαιτήσεις Συστήματος

Υπάρχουν διαφορετικές πιθανές χρήσεις της πλατφόρμας και ανάλογα με τον σκοπό τον οποίο θα εξυπηρετεί θα υπάρξει και υλοποίηση. Οι κυριότερες χρήσεις είναι:

- Για **καλλιτεχνικούς σκοπούς**, όπου το προϊόν της πλατφόρμας δεν είναι απλά ηχητικά δεδομένα, αλλά στοχεύει στη παραγωγή μουσικής, ηχητικών τοπίων κλπ.
- Για **τηλεπικοινωνιακούς σκοπούς**. Στη προκειμένη περίπτωση η εικόνα και ο ήχος δεν είναι οπτικά και ακουστικά ερεθίσματα αντίστοιχα, αλλά μορφές ανταλλαγής πληροφορίας.
- Για **επεξεργασία εικόνας-ήχου**. Εδώ σκοπεύουμε στην επεξεργασία εικόνας χρησιμοποιώντας ηχητικά εφέ και τεχνικές επεξεργασίας ήχου (και αντίστροφα).

Εμείς θα έχουμε σαν γνώμονα κατά τον σχεδιασμό την δημιουργία μίας ευέλικτης και πολυχρηστικής πλατφόρμας, και όχι τη δημιουργία μίας αποκλειστική μηχανή αντιστοίχησης. Για να το πετύχουμε αυτό θα πρέπει να έχουμε ακριβέστατη αντιστοίχηση. Δηλαδή το σύστημα πρέπει να μην αφήνει περιθώρια εσφαλμένης αντιστοίχησης ή παράλειψης δεδομένων.

Επίσης σημαντικό είναι η εφαρμογή να έχει δυνατότητες διασύνδεσης με άλλες πλατφόρμες (μεταφοράς ήχου, επεξεργασίας εικόνας κλπ) καθώς επίσης και δυνατότητες μεταποίησης και εξέλιξης

μετά την υλοποίησης της.

Τέλος, σημαντικό κριτήριο κατά τον σχεδιασμό θα είναι η ευκολία χρήσης της, καθώς πρόκειται για μια πολυμορφική εφαρμογή που απευθύνεται σε χρήστες με διαφορετικό background.

4.3 Σχεδιασμός Υλοποίησης

Ο σχεδιασμός υλοποίησης θα γίνει ακολουθώντας τα εξής βήματα:

1. Ανάλυση και επιλογή υλικού ανάπτυξης της πλατφόρμας.
2. Ανάλυση χρωματικών μοντέλων και τρισδιάστατων δεδομένων.
3. Σχεδιασμός μηχανής παραγωγής/επεξεργασίας ηχητικής εξόδου.
4. Σχεδιασμός μηχανής αντιστοίχησης (mapping).
5. Ένωση μηχανών mapping και ήχου, σε μία ενιαία πλατφόρμα.
6. Ανάλυση και βελτιστοποίηση αλγόριθμων.

Αφού πραγματοποιηθούν τα παραπάνω βήματα θα γίνει μία γενική ανάλυση της -ολοκληρωμένης πλέον- πλατφόρμας όπου θα παρουσιαστούν τα αποτελέσματα αυτής και θα μελετηθούν οι επικείμενες προοπτικές του τομέα.

4.4 Ανάλυση και Επιλογή Υλικού Ανάπτυξης Πλατφόρμας

Όπως αναφέρθηκε παραπάνω η εφαρμογή θα πρέπει να είναι ευέλικτη, σταθερή, αποδοτική και εύκολη στη χρήση. Έχοντας αυτά τα χαρακτηριστικά, σε συνδυασμό με τις απαιτήσεις του συστήματος έγινε η επιλογή των παρακάτω υλικών για ανάπτυξη της εφαρμογής.

Processing (Γλώσσα Προγραμματισμού)

Για την ανάπτυξη της πλατφόρμας η γλώσσα προγραμματισμού η οποία κρίθηκε καταλληλότερη είναι η Processing. Πρόκειται για έναν wrapper ο οποίος είναι πάνω στη γλώσσα προγραμματισμού Java. Παρακάτω επισημαίνονται τα κυριότερα κριτήρια επιλογής:

- **Πολυμεσικός Προσανατολισμός:** Εξ ορισμού η Processing δημιουργήθηκε για την ανάπτυξη οπτικοακουστικών εφαρμογών. Έτσι, πέρα από τις έτοιμες λειτουργίες που προσφέρει στον σχετικό τομέα, μπορούν να βρεθούν πρόσθετες εξωτερικές βιβλιοθήκες για πιο εξειδικευμένες χρήσεις. Επιπρόσθετα η δομή της γλώσσας είναι τέτοια που παρόλο που είναι καθαρά αντικειμενοστραφής (Java) έχει σύνταξη τέτοια ώστε να δίνεται έμφαση στα στοιχεία ήχου/εικόνας
- **Ταχεία Ανάπτυξη (Rapid Developing):** Η γλώσσα αυτή προσφέρεται για γρήγορη ανάπτυξη εφαρμογών. Έχοντας πολλές απλουστευμένες ή/και διαφοροποιημένες λειτουργίες (από τηJava) η υλοποίηση ιδεών γίνεται εύκολα και γρήγορα.
- **Ταχεία Εξέλιξη (Rapid Development):** Πρόκειται για μια -σχετικά- καινούργια γλώσσα προγραμματισμού για την οποία αναπτύσσονται πολλές βιβλιοθήκες και επιπρόσθετα η γλώσσα αυτή καθ αυτή εξελίσσεται ταχύτατα.

Αξίζει να σημειωθεί ότι η Processing μπορεί να λειτουργήσει με σειριακό προγραμματιστικό μοντέλο, event-driven, αλλά κυρίως (από default) χρησιμοποιώντας μια διαδικασία Initialization η οποία τρέχει πριν το πρόγραμμα και μετά τρέχοντας συνεχώς συναρτήσεις (οι οποίες βρίσκονται μέσα στην draw). Αυτό το μοντέλο λειτουργίας φέρνει πολύ στην γλώσσα προγραμματισμού C# της Microsoft με την οποία διαφέρει στο ότι πέραν της draw υπάρχει και η update.

19: Processing Splash Screen

Minim (βιβλιοθήκη)

Το Minim είναι η βιβλιοθήκη για Processing η οποία επιλέχθηκε για τις λειτουργίες σχετικά με τον ήχο. Η βιβλιοθήκη αυτή είναι πάνω στη Javasound. Αυτό την κάνει πολύ σταθερή και αξιόπιστη. Επιπλέον της προσφέρει αξιοπιστία.

Πέραν τούτου προσφέρει επιπλέον λειτουργίες οι οποίες -αν και δεν είναι αδύνατο- θα ήταν πολύ δύσκολο να υλοποιηθούν χωρίς αυτή. Αναφέρονται ενδεικτικά μερικές:

- AudioPlayer: Ενσωματωμένη μηχανή αναπαραγωγής ήχου.
- AudioOutput: Ενσωματωμένη μηχανή δημιουργίας ήχου.
- BeatDetection: Ανίχνευση beats – χρήσιμο για εύρεση tempo, το οποίο χρειάζεται για καλύτερη ρύθμιση των effect ήχου.
- FFT: Δυνατότητα εφαρμογής μετασχηματισμού Fourier, ο οποίος είναι πολύ χρήσιμο εργαλείο για την ανάλυση ήχου.
- AudioRecorder: Μηχανή ηχογράφισης ήχου, η οποία υποστηρίζει μορφή αρχείων *.wav , σε περίπτωση που χρειαστεί.

NextText (βιβλιοθήκη)

Μία από τις λειτουργίες η οποία είναι λίγο δύσκολη στον χειρισμό στην Processing (δυσκολία η οποία κληρονομήθηκε από τη Java), είναι η επεξεργασία/εμφάνιση κειμένου. Με την χρήση αυτής της βιβλιοθήκης νέες λειτουργίες ξεκλειδώνονται.

Αρχικά αναπτύχθηκε για Java και αργότερα έγινε port για Processing. Ενώ λοιπόν υπάρχουν κάποιες λειτουργίες για χρήση στη Processing όλες οι δυνατότητες της Java είναι ξεκλείδωτες και διαθέσιμες για αξιοποίηση. Μπορεί αυτή η πλατφόρμα να μην απαιτεί -για την ώρα- εξελιγμένες λειτουργίες κειμένου. Αλλά ακόμη και τα βασικά είναι αρκετά πιο εύκολα στη χρήση, και επικείμενη εξέλιξη και αξιοποίηση των πιο προηγμένων δυνατοτήτων είναι πραγματοποιήσιμη.

Kinect (αισθητήρας/κάμερα)

Το Kinect της Microsoft είναι η συσκευή η οποία απαιτείται για να χρησιμοποιήσει ο χρήστης τη πλατφόρμα. Συμπεριλαμβάνει πομπό υπέρυθρων, μία ToF κάμερα/αισθητήρα, μία χρωματική κάμερα, καθώς και ένα array μικροφώνων. Η επιλογή της έγινε κυρίως για τους εξής λόγους:

I. Πρόκειται για την πιο διαδεδομένη κάμερα αυτού του τύπου. Σύμφωνα με το Guiness Book of Records είναι η ηλεκτρονική συσκευή με τον πιο ταχύ ρυθμό πώλησης όλων των εποχών. Οι περισσότεροι χρήστες εφαρμογών πολυμέσων, είναι ήδη κάτοχοι της.
II. Υπάρχει βιβλιοθήκη για Processing (βλ. παρακάτω), η οποία είναι εξαιρετικά σταθερή.
III. Ο σχεδιασμός της είναι υψηλών προδιαγραφών, διατηρώντας όμως χαμηλή τιμή και ευκολία στη χρήση.

20: Η συσκευή Kinect

Σε αυτό το σημείο αξίζει να γίνει μία σύντομη αναφορά στα βασικά χαρακτηριστικά της συσκευής ένα-προς-ένα.

Depth Sensor: Ο αισθητήρα βάθους αποτελείται από δύο μέρη. Το πρώτο είναι ένας προβολέας υπέρυθρων ακτίνων, ενώ το δεύτερο ένας CMOS αισθητήρας για υπέρυθρη ακτινοβολία. Με αυτά τα δύο και χρησιμοποιώντας την αρχή Time-of-Flight (εξού και το όνομα αυτού του τύπου κάμερας) μπορεί να υπολογιστεί η απόσταση ενός σημείου από τους αισθητήρες.

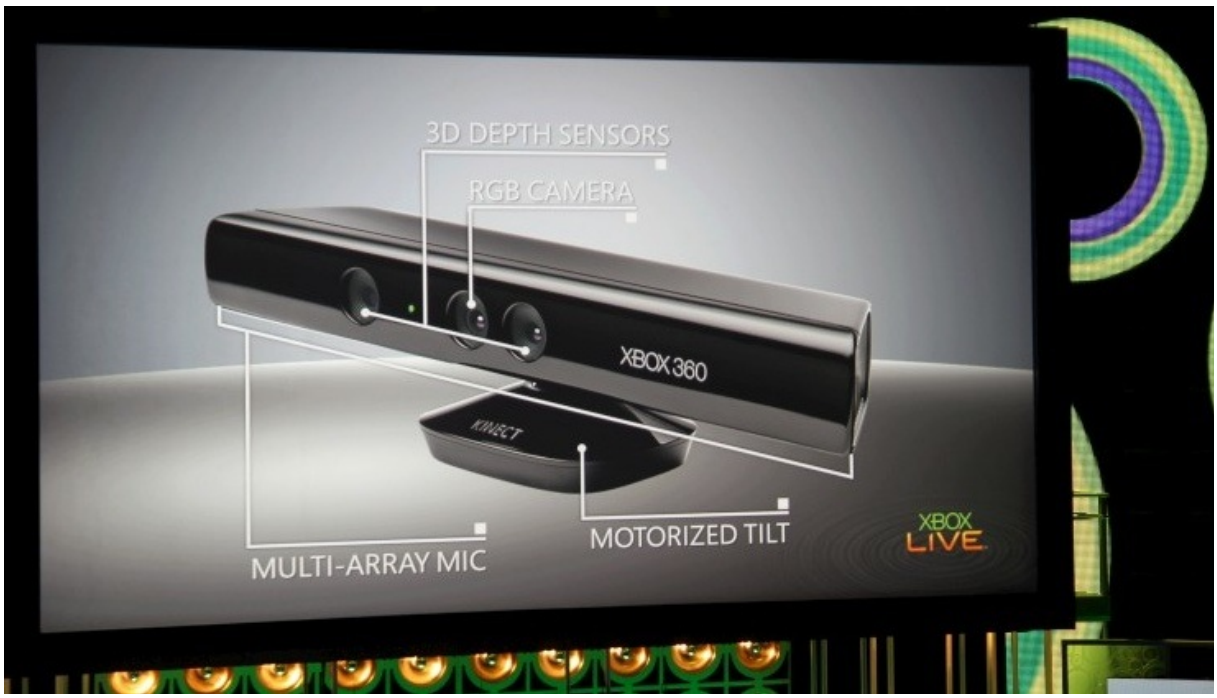
Χρησιμοποιώντας τεχνολογία υπερύθρων υπάρχει το σημαντικό πλεονέκτημα ότι η κάμερα βάθους μπορεί να λειτουργήσει κάτω από οποιονδήποτε φωτισμό. Μόνο ελάττωμα είναι ότι δε μπορούν να χρησιμοποιηθούν δύο κάμερες ταυτόχρονα στον ίδιο χώρο, καθώς δεν μπορεί να γίνει διαχωρισμός των ακτινών.

Από τεχνικής άποψης η κάμερα βάθους μας προσφέρει πιστότητα τρισδιάστατης εικόνας αρκετά ακριβή. Συγκεκριμένα, η ανάλυση της είναι VGA (640pixels πλάτος επί 480pixels ύψος), ενώ το κάθε στοιχείο χρησιμοποιεί 11bit παρέχοντας 2048 επίπεδα βάθους. Ο ρυθμός ανανέωσης της εικόνας είναι στα 30Hz. Το εύρος της κάμερας είναι κατά προσέγγιση από 120cm μέχρι 350cm.

RGB Sensor: Η κάμερα του Kinect έχει χαρακτηριστικά τα οποία να εναρμονίζονται με τον αισθητήρα βάθους. Έτσι λοιπόν μπορεί να προσφέρει επίσης ανάλυση VGA με 30 καρέ ανά δευτερόλεπτο. Η πιστότητα εικόνοστοιχείου είναι στα 8bit (μιας και πρόκειται για RGB κάμερα). Τέλος, και οι δύο αισθητήρες έχουν οπτικό πεδίο 57º οριζοντίως και 43º καθέτως.

Microphone Array: Η συσκευή είναι εφοδιασμένη με μια σειρά τεσσάρων μικροφώνων. Παρόλο που δε θα χρησιμοποιηθεί, καθώς είναι περιττή στη προκειμένη φάση ανάπτυξης αξίζει να αναφερθεί καθώς πρόκειται για εξαιρετικής κατασκευής. Πέραν τούτου το επίσημο SDK της Microsoft μπορεί να προσφέρει λειτουργίες φωνητικής αναγνώρισης σε διάφορες γλώσσες καθώς επίσης και εντοπισμού ηχητικής πηγής. Από προηγούμενη προσωπική σχετική εμπειρία μου, μπορώ να εκφράσω την άποψη ότι παρέχονται υπηρεσίες ακριβέστατης ηχητικής αξιοπιστίας.

Motor: Τέλος, η συσκευή έχει τοποθετημένο στη βάση της έναν κινητήρα, ο οποίος μπορεί να αλλάξει την κλίση της σε εύρος 27º. Ο κινητήρας αυτός ρυθμίζεται από το software.



21: Τα διαφορετικά μέρη της συσκευής

Στη φωτογραφία παραπάνω (από την έκθεση E3) μπορεί κάποιος να δει τα σημεία στα οποία βρίσκονται τα βασικά λειτουργικά μέρη της συσκευής.

SimpleOpenNI (βιβλιοθήκη)

Παρόλο που η SimpleOpenNI εδώ αναφέρεται σαν βιβλιοθήκη, θα μπορούσε να χαρακτηριστεί σαν ολοκληρωμένο πακέτο ή/και σαν project σχετικά με το Kinect.

Το πακέτο συμπεριλαμβάνει τον **NITE** installer ο οποίος επιτρέπει να εγκατασταθούν οι οδηγοί της συσκευής από τη **PrimeSense.** Τέλος, αφού εγκατασταθούν οι drivers μπορεί να τοποθετηθεί το **OpenNI** το οποίο μας προσφέρει ουσιαστικά τις λειτουργίες του Kinect.

Ο τρόπος λειτουργίας της βιβλιοθήκης είναι σχετικά απλός.

1. Αρχικά κάνουμε import τη βιβλιοθήκη στο project μας *import SimpleOpenNI.*;*
2. Δημιουργούμε ένα αντικείμενο του τύπου SimpleOpenNI *SimpleOpenNI example;*
3. Αρχικοποιούμε *example = new SimpleOpenNI(this);*
4. Ενεργοποιούμε τον αισθητήρα που θέλουμε *example.enableRGB();*
5. Τώρα μπορούμε να προχωρήσουμε σε περαιτέρω λειτουργίες με τα δεδομένα από τον ενεργοποιημένο αισθητήρα/ες.

Περισσότερα παραδείγματα κώδικα θα ακολουθήσουν, μέσα από snipets του προγράμματος.

4.5 Ανάλυση Χρωματικών Μοντέλων και Τρισδιάστατων Δεδομένων

Σε κάθε χρωματικό μοντέλο θα γίνει ανάλυση βάσει: των συνιστώσεων για τη δημιουργία του χρώματος, την πιστότητα με την οποία αποδίδεται η εικόνα και την δημοτικότητα αυτού. Επίσης η ανάλυση θα γίνεται με έμφαση στις εικόνες με βάθος χρώματος τα 24bit, ενώ για υπόλοιπες τιμές θα γίνεται μία πιο επιφανειακή εξέταση.

RGB

Για αυτό το χρωματικό μοντέλο έχουμε αναφερθεί εκτενέστατα παραπάνω. Συνοψίζοντας έχουμε τα εξής δεδομένα:

i. Το χρώμα δημιουργείται βάσει 3 συνισταμένων των 8bit. Αυτές είναι: Κόκκινο (Red), Πράσινο (Green), Μπλε (Blue). Κάθε συνιστώσα παίρνει τιμές 0-255 και συνολικά αποδίδονται 2^{24} αποχρώσεις.
ii. Οι 16.777.216 που έχουμε στη διάθεση μας, είναι περισσότερες από όσες μπορεί να διακρίνει το ανθρώπινο μάτι. Άρα και κρίνονται αρκετές (από πλευράς οπτικής).
iii. Είναι το πιο διαδεδομένο χρωματικό μοντέλο ψηφιακής εικόνας εδώ και αρκετά χρόνια.

Σύμφωνα με τα παραπάνω μπορούμε να συμπεράνουμε ότι σε περίπτωση που χρησιμοποιηθεί το RGB μοντέλο με 1:1 (1 προς 1) αντιστοίχιση, θα μπορέσουμε να έχουμε στην έξοδο μας 2^{24} διαφορετικές τιμές ή -πάλι χωρίς αλλοίωση- τρείς συνιστώσες των 2^8 τιμών.

CMYK

Αυτό είναι ένα "αφαιρετικό" χρωματικό μοντέλο, καθώς η δημιουργία χρωμάτων συνίσταται από τα χρώματα τα οποία απορροφά κάθε χρώμα, ενώ το RGB από τα χρώματα που αντανακλώνται. Για παράδειγμα το κυανό απορροφά κόκκινο φως, αλλά αντανακλά το πράσινο και το μπλέ.

i. Το χρώμα δημιουργείται βάσει 4 συνισταμένων. Αυτές είναι: Κυανό (Cyan), Ματζέντα (Magenta), Κίτρινο (Yellow), Μαύρο (Key).
ii. Το CMYK -θεωρητικά- μπορεί να παράγει λιγότερες αποχρώσεις από ότι το RGB (κρατώντας σταθερή τη μνήμη που καταλαμβάνει μία εικόνα).
iii. Χρησιμοποιείται κατά κόρον σε συσκευές εκτύπωσης εικόνων.

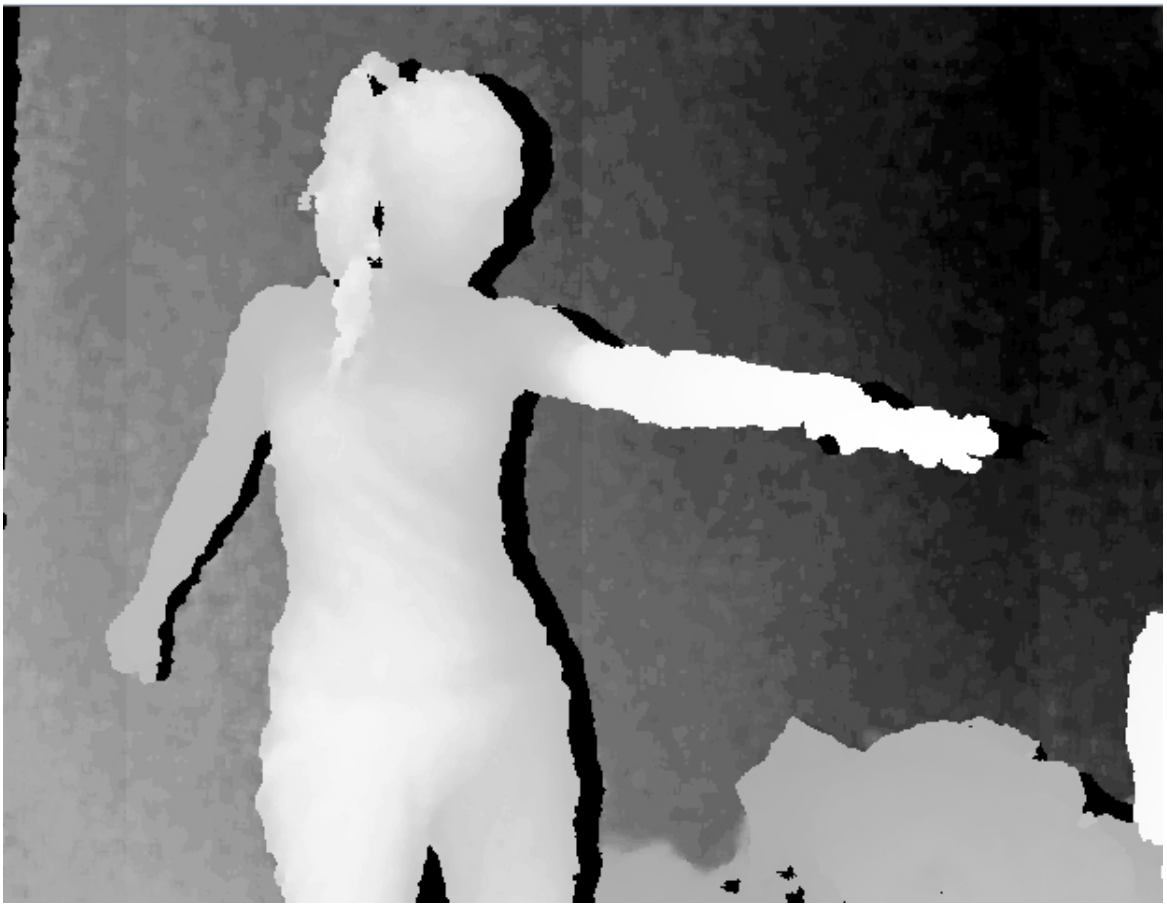
Από τα παραπάνω καταλήγουμε στο συμπέρασμα ότι παρόλο που μια CMYK εικόνα έχει περισσότερες συνισταμένες (άρα έχουμε και μεγαλύτερη ευελιξία στον τρόπο χρήσης του) καλό θα ήταν να αποφευχθεί η χρήση του καθώς θα έχουμε απώλειες κατά τη μετατροπή.

Όπως αναφέρθηκε στην αρχή της πτυχιακής αναφοράς, τα χρωματικά μοντέλα αναλύθηκαν και επεξηγήθηκαν καθώς θα εξετάσουμε αργότερα θεωρητικές χρήσεις τους και για να μπορέσει ο χρήστης να κατανοήσει τις ήδη υπάρχουσες εφαρμογές image-sound mapping οι οποίες τα χρησιμοποιούν. Στη προκειμένη εφαρμογή, η αντιστοίχηση γίνεται μέσω των **δεδομένων βάθους** τα οποία μας προσφέρει το Kinect. Οι δύο κυριότερες μορφές τους επεξηγούνται παρακάτω.

Depth Map (Point Cloud)

Αυτή είναι η μορφή των δεδομένων έτσι όπως τις στέλνει η συσκευή. Πρόκειται για τα τρισδιάστατα δεδομένα όπως τα περιγράψαμε σε προηγούμενα μέρη της πτυχιακής. Συνοψίζοντας λοιπόν γνωρίζουμε ότι πρόκειται για "εικόνες" οι οποίες έχουν στοιχεία για την απόσταση των αντικειμένων από τον αισθητήρα, ανάλυσης 640x480. Κάθε ένα από τα στοιχεία της εικόνας μπορεί να πάρει μία εκ των 2048 τιμών που είναι διαθέσιμες. Οι εικόνες αυτές ανανεώνονται 30 φορές ανά δευτερόλεπτο.

Σε περίπτωση που γίνει απλή αναπαράσταση του point cloud έχουμε την απεικόνιση του χώρου. Αυτό μπορεί να γίνει είτε τοποθετώντας τα στην αντίστοιχη θέση σε μία τρισδιάστατη εικόνα *(κεφάλαιο "Τρισδιάστατη Ψηφιακή Εικόνα")*, είτε κάνοντας αντιστοίχηση κάθε τιμής σε μία χρωματική τιμή οπότε και έχουμε μια δισδιάστατη χρωματική απεικόνιση του χώρου (όπως στην εικόνα παρακάτω).



22: Grayscale Depth Mapping

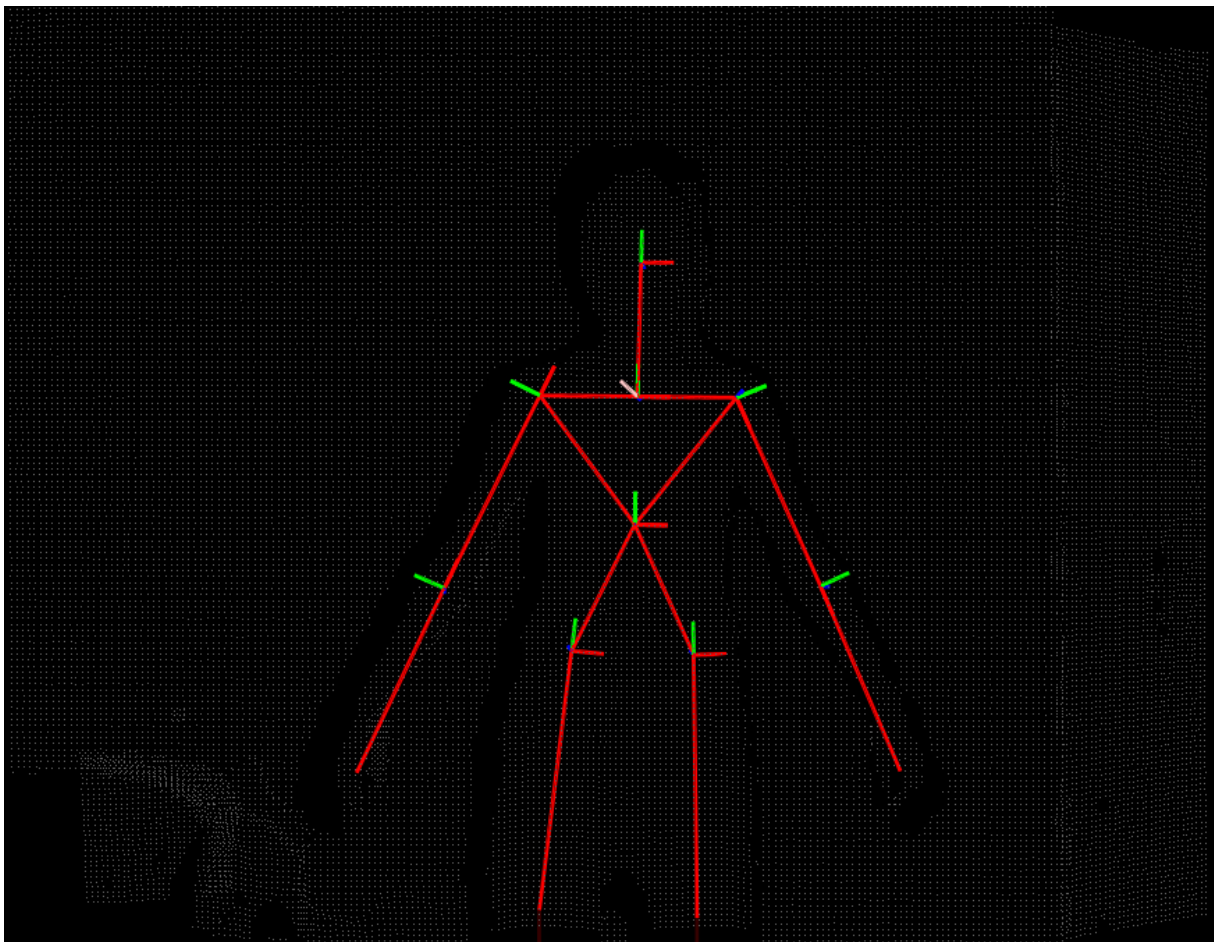
Σε αυτή την εικόνα η φωτεινότητα δείχνει πόσο κοντά βρίσκεται ένα σημείο στον αισθητήρα. Έτσι μπορούμε να παρατηρήσουμε ότι ο χρήστης έχει το αριστερό του χέρι παρατεταμένο προς τη κατεύθυνση της κάμερας, ενώ το δεξί στην αντίθετη κατεύθυνση (πίσω από το σώμα του).

Skeleton Tracking

Σε αυτού του τύπου δεδομένα, αντί για να έχουμε τιμή για κάθε pixel του χώρου, γίνεται εντοπισμός του χρήστη (ο οποίος είναι στο πεδίο της κάμερας) και μας επιστρέφονται τιμές για συγκεκριμένα σημεία του σώματος του. Για παράδειγμα ο προγραμματιστής μπορεί να ζητήσει να "τραβήξει" μόνο τη θέση στον χώρο του αριστερού ώμου του χρήστη. Καθ' αυτόν τον τρόπο σώνεται πολύς χρόνος, επειδή δε χρειάζεται να προγραμματιστεί ο εντοπισμός του χρήστη, αντί αυτού, ο προγραμματιστής μπορεί να ασχοληθεί κατευθείαν στις ουσιαστικές λειτουργίες του προγράμματος.

Αυτού του τύπου δεδομένα χρησιμοποιούνται στην συντριπτική πλειοψηφία των εφαρμογών που κάνουν χρήση gestures και αυτά τα δεδομένα θα χρησιμοποιηθούν και σε αυτή τη πλατφόρμα.

Παρακάτω ακολουθεί μία εικόνα στην οποία έχουν γίνει visualize τα σημεία του χρήστη για τα οποία αντλούνται δεδομένα μέσω του Skeleton Tracking. Τα σημεία αυτά αποτυπώνονται πάνω στην εικόνα του χώρου (όπως γίνεται απεικόνιση μέσα από τα Point Clouds).



23: Skeleton Tracking

4.6 Σχεδιασμός & Υλοποίηση(Design & Implementation)

Προτού ξεκινήσει ο σχεδιασμός της πλατφόρμας έγινε μία προκαταρκτική μελέτη πάνω στη Processing και προέκυψαν τα ακόλουθα συμπεράσματα τα οποία και έπαιξαν σημαντικό ρόλο στον σχεδιασμό που ακολούθησε:

i. **Ταχύτητα επεξεργασίας** Η Java είναι μία γλώσσα που χωλαίνει όταν προκύπτουν σύνθετες μαθηματικές πράξεις. Αυτό είναι ένα πρόβλημα το οποίο έχει και η Processing (όπως αναφέρθηκε παραπάνω). Αποφασίστηκε λοιπόν, ο αριθμός των περίπλοκων μαθηματικών πράξεων να περιοριστεί σε ένα ελάχιστο.
ii. **Μηχανή Γραφικών** Η Processing μπορεί να υποστηρίξει γραφικά τύπου OpenGL (τα οποία είναι πιο αξιόπιστα). Μετά από κάποια tests που έτρεξαν όμως, φάνηκε ότι όταν τα γραφικά είναι τύπου P3D υπάρχει ταχύτερη απόδοση. Επειδή λοιπόν έμφαση δίνεται στην όραση και όχι στην απεικόνιση γραφικων, θα χρησιμοποιηθούν τα P3D.
iii. **Μέθοδος Προγραμματισμού** Όπως αναφέρθηκε παραπάνω, η Processing μπορεί να υποστηρίξει event-driven και σειριακό προγραμματισμό. Παραταύτα η setup-draw μέθοδος είναι πολύ πιο αποδοτική και αυτή επιλέχθηκε.
iv. **Περιβάλλον Προγραμματισμού** Αρχικά η υλοποίηση του project ξεκίνησε σε περιβάλλον Linux. Επειδή όμως προέκυπταν συνεχώς προβλήματα και ασυμβατότητες (με το Kinect, τα γραφικά κ.α.) έγινε αλλαγή σε Windows. Επίσης ενώ αρχικά το development ήταν σε Eclipse (το οποίο είναι ένα περιβάλλον που προσφέρει πολλά χρήσιμα εργαλεία), έγινε αλλαγή στο IDE της Processing – κυρίως επειδή το export δεν είναι ολοκληρωμένο.

Εκτός από τα συμπεράσματα που προκύπτουν παραπάνω αποφασίστηκε αρχικά να σχεδιαστεί / υλοιποιηθεί η μηχανή παραγωγής επεξεργασίας ήχου. Κατόπιν να γίνει μία αξιολόγησή της, και ύστερα να αναπτυχθεί η μηχανή αντιστοίχησης. Τέλος αφού γινόταν η ένωση σε μία ενιαία πλατφόρμα θα επέλθει το debugging (παράλληλα με τη βελτιστοποίηση) και η τελική αξιολόγηση της πλατφόρμας.

4.7 Σχεδιασμός Μηχανής Παραγωγής/Επεξεργασίας Ήχου

Η αρχική σύλληψη ήταν η πλατφόρμα να έχει τρία mode λειτουργίας. Το πρώτο θα προσέφερε λειτουργίες **σύνθεσης ήχου**. Το δεύτερο θα αποσκοπούσε στην **on-the-fly επεξεργασία ήχου**, μέσω διαφόρων effects. Το τρίτο θα ήταν ουσιαστικά ένα **interface διασύνδεσης με εξωτερικές πλατφόρμες ήχου** προσφέροντας λειτουργίες controller.

Μίας και η διασύνδεση με το Kinect θα γίνει σε αργότερο στάδιο, αρχικά όλες οι μηχανές ήχου δημιουργήθηκαν και δοκιμάστηκαν με βασικό χειρισμό από το ποντίκι και το πληκτρολόγιο του υπολογιστή. Για να επιτευχθεί αυτός ο χειρισμός έγινε ανάγνωση της σχετικής -ως προς την πάνω αριστερά γωνία της οθόνης- θέσης του δείκτη του ποντικιού και γράφτηκε κάποιος κώδικας για ανάγνωση του πληκτρολογίου.

```
//KEYBOARD input handling
void keyPressed() {
  if (key=='P'||key=='p') {
    if (!player.isPlaying())   player.play();
    else player.pause();
    //play-pause
  }
  else if (key=='S'||key=='s') {
    player.cue(0);
    player.pause();
    //stop
  }
}
```

24: Κώδικας για χειρισμό keyboard inputs

Αρχικά λοιπόν (για να μπορέσει να υλοποιηθεί οποιαδήποτε ενέργεια σχετική με ήχο), φτιάξαμε ένα αντικείμενο από τη βιβλιοθήκη Minim και ενεργοποιήσαμε την ηχητική έξοδο.

```
//Create Minim  (object for sound)
Minim minim;
//Create Audio Output
AudioOutput output;


//INITIALIZING AUDIO
void InitAudio() {
  //Creating Minim Object
  minim = new Minim(this);
  //Output type:Stereo/Mono, buffer size, sample rate, bitdepth
  output=minim.getLineOut(Minim.STEREO, 2048, 44100, 16);
  println("Audio Initialized");
}
```

25: Κώδικας για δημιουργία ηχητικής εξόδου

4.7.1 Synthesis Mode

Όσον αφορά τη σύνθεση ήχου, πρώτη σκέψη υλοποίησης ήταν μία μηχανή η οποία θα εκμεταλλεύεται τα δύο βασικά χαρακτηριστικά του ήχου – ένταση και χροιά. Όπως ήδη γνωρίζουμε κάθε ήχος προέρχεται από κάποια ταλάντωση. Η πιο σύνηθες μορφή είναι ένα κύμα ημιτονοειδής μορφής. Το πλάτος του κύματος είναι ανάλογο της έντασης του ήχου (μικρό πλάτος -> χαμηλή ένταση), ενώ η συχνότητα του είναι ανάλογη της χροιάς του (χαμηλή συχνότητα -> μπάσος ήχος).

Πρώτο λογικό βήμα λοιπόν, είναι η **δημιουργία ενός ημιτονοειδές κύματος.** Στη συνέχεια αυτό το σήμα περάστηκε σαν ηχητικό σήμα στην έξοδο από το αντικείμενο minim. Τέλος, για να μπορούμε να διαμορφώσουμε το ημιτονοειδές κύμα το κάναμε mapping με τις συντεταγμένες του ποντικιού.

Το πεδίο αντιστοίχησης ήταν οι διαστάσεις της οθόνης. Στη συνέχεια η οριζόντια θέση του

ποντικιού στην οθόνη αντιστοιχίστηκε στη συχνότητα του κύματος. Συγκεκριμένα, ξεκινώντας από το μηδέν (τέρμα αριστερά στην οθόνη), έφτανε μέχρι τα 22kHz (τέρμα δεξιά στην οθόνη) όπου και είναι -περίπου- η μέγιστη συχνότητα που μπορεί να αντιληφθεί το ανθρώπινο αυτί.

Κάπως έτσι (αντιστρέφοντας όμως τους άξονες, για λόγους ευχρηστίας) έγινε και η αντιστοίχηση του πλάτους στο κύμα μας. Εννοώντας ότι στον κάθετο άξονα το μηδέν βρισκόταν τέρμα κάτω στην οθόνη, ενώ το μέγιστο πλάτος (τιμή: ένα) ήταν στην κορυφαία θέση στην οθόνη.

Τελευταίο βήμα ήταν κάποιες μικρο-επεμβάσεις στον τρόπο λειτουργίας. Για παράδειγμα το ηχητικό σήμα ενεργοποιούταν μόνο όταν ήταν πατημένο το αριστερό κουμπί του ποντικιού (για να ακούγεται μόνο όταν το επιθυμούμε). Επίσης, προστέθηκε ένα smoothing 20ms στις αλλαγές των χαρακτηριστικών για πιο ευχάριστο αισθητικό αποτελέσματα. Και για λόγους testing, δίπλα από τον δείκτη του ποντικιού φαινόταν η θέση του στην οθόνη.

```
//SINE WAVE AUDIO
void CreateAudioWave(float frequency, float amplitude) {
  //Creating the Sine Wave  (freq, amp, sample rate)
  sine = new SineWave(frequency, amplitude, 44100f);
  //Smoothing 20ms
  sine.portamento(200);
  //Adding Sine Wave to the actual Output
  output.addSignal(sine);
  println("Sine Created + Added. Freq=  "+frequency);
}

void UpdateAudioWave() {
  sine.setFreq(map(mouseX, 0, 1024, 0, 20000));
  sine.setAmp(map((height-mouseY), 0, 600, 0, 1));
}
```

26: Κώδικας για δημιουργία και επεξεργασία ημιτονοειδές ηχητικού κύματος

Αφού ολοκληρώθηκε και δοκιμάστηκε το ημιτονοειδές κύμα, δημιουργήθηκαν άλλοι δύο τύποι ηχητικών σημάτων. Σήμα **τετραγωνικού παλμού** και **τριγωνικό σήμα**. Αξίζει να σημειωθεί ότι -εν αντιθέσει με το ημιτονοειδές- αυτού του τύπου αντικείμενα δεν υπάρχουν έτοιμα στην βιβλιοθήκη και δημιουργήθηκαν χειροκίνητα.

Για να γίνει αυτό, φτιάξαμε ένα δικό μας τύπου αντικειμένου (κλάση), ο οποίος κληρονομεί χαρακτηριστικά από την κλάση AudioSignal της Minim.

Παρακάτω παρατίθεται ο κώδικας της κλάσης για το τριγωνικό σήμα (το οποίο είναι και αυτό το οποίο κρατήθηκε τελικά (έναντι του τετραγωνικού παλμού).

```
class SawWave implements AudioSignal
{
  void generate(float[] samp)
  {
    //Amplitude
    float amp = map((height-mouseY), 0, height, 0, 1);
    //Freq
    float frequency = map(mouseX, 0, width, 1, 2000);
    //Peaks
    float peaks = frequency/21.5332;
    //Interpolation
    float inter = float(samp.length) / peaks;
    for ( int i = 0; i < samp.length; i += inter )
    {
      for ( int j = 0; j < inter && (i+j) < samp.length; j++ )
      {
        samp[i + j] = map(j, 0, inter, -amp, amp);
      }
    }
  }

  //create mono signal
  void generate(float[] left, float[] right)
  {
    generate(left);
    generate(right);
  }
}
```

27: Κώδικας της κλάσης τριγωνικού ηχητικού σήματος

Κάπως έτσι ολοκληρώθηκε ο πυρήνας της μηχανής δημιουργίας ήχου. Αργότερα θα γίνει μετατροπή του ούτως ώστε να δέχεται σαν inputs δεδομένα από τον αισθητήρα βάθους του Kinect.

4.7.2 Effects Mode

Η δεύτερη λειτουργία της μηχανής ήχου είναι αυτή η οποία θα προσφέρει στον χρήστη δυνατότητες επεξεργασίας ενός ηχητικού σήματος κατά την διάρκεια την οποία αυτό αναπαράγεται.

Προτού ξεκινήσει η δημιουργία της μηχανής, φορτώνουμε ένα κομμάτι στο αντικείμενο player της βιβλιοθήκης (το οποίο είναι συνδεδεμένο στο output). Αυτό επιτυγχάνεται με τον εξής κώδικα (αφού πρώτα δημιουργήσουμε το αντικείμενο): *player = minim.loadFile("PWSteal.mp3");* *όπου PWSteal.mp3* είναι ο τίτλος του αρχείου.

Το συγκεκριμένο αρχείο είναι ένα μουσικό κομμάτι με τον τίτλο “PWSteal.Ldpinch.D” από τον καλλιτέχνη Richard James, γνωστός με το ψευδώνυμο Aphex Twin. Αυτό επιλέχθηκε επειδή είναι ένα λιτό και ρυθμικό κομμάτι, οπότε είναι πιο εύκολο να παρατηρηθούν ακόμη και μικρές αλλαγές στον ήχο.

Το πρώτο effect το οποίο δοκιμάστηκε ήταν ένα **Oscillation Effect** στην ένταση. Ο τρόπος με τον οποίο λειτουργεί έχει ως εξής:

- Αρχικά δημιουργείται ένα περιοδικό σήμα (στη δική μας περίπτωση δημιουργήθηκε ένα ημιτονοειδές σήμα).
- Στη συνέχεια το σήμα αυτό προστίθεται ή -πιο σύνηθες- πολλαπλασιάζεται, κατά πλάτος, με το ηχητικό σήμα. Αυτή η διαδικασία, στη δική μας περίπτωση, γίνεται άνα ηχητικό sample και η πράξη η οποία λαμβάνει χώρα είναι πολλαπλασιασμός.
- Τέλος, το διαμορφωμένο ηχητικό σήμα οδηγείται στην έξοδο.

Ένας άλλο τρόπος χρήσης του Oscillation Effect είναι όταν έχουμε εκτός από το ηχητικό σήμα, ενεργοποιημένο πάνω του ένα ηχητικό Effect (για παράδειγμα Echo), και ανάλογα με την τιμή του Oscillator αλλάζουμε μία παράμετρος του (Ένταση, Διάρκεια, Χρόνος κλπ.) - αντί της εκτέλεσης κάποιας μαθηματικής πράξης. Αλλά στη παρούσα εργασία, η συγκεκριμένη περίπτωση δε θα εξεταστεί.

Για το ημιτονοειδές σήμα, αρχικά χρησιμοποιήθηκε η ίδια μεθοδολογία που ακολουθήσαμε στο Synthesis Mode. Επειδή όμως παρατηρήθηκε σοβαρή επιβάρυνση στο συνολικό σύστημα μία νέα μέθοδος χρησιμοποιήθηκε.

Αρχικά δημιουργήθηκε μία κλάση παρόμοιου τύπου με την SawWave, έχοντας σαν βασική διαφορά ότι αντί για να δημιουργεί samples, δέχεται σαν είσοδο τα samples πριν φτάσουν στην συνολική έξοδο. Στην συνέχεια γίνεται πολλαπλασιασμός με το ημίτονο (το οποίο βρίσκεται με τον τρέχον χρόνο – ακρίβεια χιλιοστού του δευτερολέπτου). Τέλος, αντικαθιστούμε με τα νέα samples και το στέλνουμε στην έξοδο.

```
class OscillatorEffect implements AudioEffect
{
  void process(float[] samp)
  {
    for (int j = 0; j < (samp.length -1); j++)
      samp[j] = samp[j]*(sin(TWO_PI*(millis()*1000)));
  }

  void process(float[] left, float[] right)
  {
    process(left);
    process(right);
  }
}
```

28: Κώδικας Oscillator Effect

Δυστυχώς ακόμη και με αυτόν τον τρόπο -παρόλο που είναι αισθητά πιο γρήγορος- η Processing δε μπορεί να αντεπεξέλθει και προκαλείται clipping στον ήχο που δημιουργείται. Ακόμη δύο τύποι effect δοκιμάστηκαν, αλλά και αυτοί είχαν το ίδιο -αρνητικό- αποτέλεσμα.

Το πρώτο ήταν Delay, στο οποίο τα samples αναπαράγονταν σε μία συνεχές διαδικασία μαζί με τον original σήμα έχοντας μία σταδιακή αποδυνάμωση.

Το δεύτερο ήταν μιας μορφής Overdrive, στο οποίο μία ομάδα από samples ανάλογα με την

ισχύς τους προκαλούσε παραμόρφωση στα γειτονικά samples.

Το effect που χρησιμοποιήθηκε στην τελική μορφή της πλατφόρμας είναι ένα φίλτρο ζώνης (Band-Pass Filter). Η επιλογή αυτή έγινε επειδή βρίσκεται ενσωματωμένο στην minim οπότε δε παρουσιάζει προβλήματα, αλλά ταυτόχρονα είναι παραμετροποιήσιμο, οπότε μπορεί να γίνει mapping.

Αρχικά είχε δοκιμαστεί ένα χαμηλοπερατό φίλτρο (Low-Pass Filter). Η θέση του ποντικιού στην οθόνη καθόριζε και τη συχνότητα αποκοπής (συχνότητες χαμηλότερες αυτής αποκόπτονται από την τελική έξοδο). Παρακάτω παρατίθεται δείγμα κώδικα με την δημιουργία/αρχικοποίηση του φίλτρου.

```
//LOW PASS FILTER EFFECT
void AddLowPassFilter() {
  //Creating filter and setting parameters (cutoff frequency, sample rate)
  lpFilter = new LowPassFS(map(mouseY,0,600,0,20000), player.sampleRate());
  //enabling filter
  player.addEffect(lpFilter);
}
```

29: Low Pass Filter, Code Snippet

Αυτό το φίλτρο χρησιμοποιήθηκε καθ' όλη τη διάρκεια του αρχικού σχεδιασμού (μέχρι και την ολοκλήρωση) της μηχανής ήχου. Στη συνέχεια όμως, όταν και υλοποιήθηκε η μηχανή αντιστοίχησης, αντί του χαμηλοπερατού φίλτρου, χρησιμοποιήθηκε φίλτρο ζώνης. Η επιλογή αυτή έγινε καθώς μπορεί ο χρήστης να επέμβει σε δύο (αντί ενός) παραμέτρους. Επιπλέον, με σωστό προγραμματισμό και χρήση της πλατφόρμας, το φίλτρο ζώνης, μπορεί να μετατραπεί σε χαμηλοπερατό αλλά και υψιπερατό φίλτρο.

Ο κώδικας του Band-Pass Filter είναι παρόμοιος με αυτόν του χαμηλοπερατού. Για παράδειγμα η αρχικοποίηση έχει ως εξής:

bpFilter = new BandPass(frequency, bandwidth, player.sampleRate());

Όπως παρατηρείται, δέχεται δύο παραμέτρους, αντί της cutoff frequency. Αυτό ωφείλεται στο γεγονός ότι η minim για να δημιουργήσει ένα BandPass Filter απαιτεί μία κεντρική συχνότητα και ένα πλάτος συχνότητας. Κατ' αυτόν τον τρόπο, το φίλτρο αντί για να ορίζεται μέσω δύο (μίας άνω και μίας κάτω) συχνοτήτων αποκοπής, ορίζεται χρησιμοποιώντας την κεντρική και αφήνοντας να περάσουν συχνότητες μέσα στο δοσμένο πλάτος (από τη κεντρική).

Περισσότερες λεπτομέρειες σχετικά με την λειτουργία του φίλτρου και τη διασύνδεση του με τον χρήστη, αναφέρονται σε ακόλουθο κεφάλαιο.

4.7.3 Audio Interface Mode

Το τελευταίο κομμάτι της μηχανής έχει να κάνει με τη χρήση της πλατφόρμας σαν διεπαφή χρήστη για ηχητικές εφαρμογές. Παρόλο που είχε γίνει μια προκαταρκτική αξιολόγηση/μελέτη, δε πρόλαβε να υλοποιηθεί στα χρονικά πλαίσια της πτυχιακής εργασίας. Παρακάτω αναφέρονται κάποια σημεία-κλειδιά της μελέτης.

Αρχικά, η λειτουργία η οποία θα εξυπηρετούσε θα ήταν να υπήρχαν κάποια triggers τα οποία θα ενεργοποιούταν από κινήσεις του χρήστη και κάποιες μεταβλητές παράμετροι οι οποίες επίσης θα λειτουργούσαν με ανάλογο τρόπο. Έτσι θα μπορούσαμε να έχουμε ένα virtual pad (ή κάποιου άλλου είδους χειριστήριο) το οποίο θα ήταν διαμορφώσιμο ανάλογα με την εκάστοτε εφαρμογή διασύνδεσης.

Για να γίνει αυτή η διασύνδεση, έπρεπε να επιλεγεί ένα πρωτόκολλο δικτύου, πάνω στο οποίο θα βασιζόταν το mode. Αυτό που είχε κριθεί καταλληλότερο για αυτή τη χρήση είναι το OSC (Open Sound Control), το οποίο αναπτύχθηκε από το εργαστήριο CNMAT στο πανεπιστήμιο του Berkeley, California. Τα βασικά κριτήρια για τα οποία έγινε αυτή η επιλογή ήταν τα εξής:

- Βασίζεται πάνω σε UDP/IP – Ethernet, τα οποία είναι εξαιρετικά γρήγορες τεχνολογίες. Επίσης υποστηρίζει TCP.
- Πλήρης υποστήριξη από την Processing. Συγκεκριμένα η βιβλιοθήκη η οποία θα χρησιμοποιείτο ήταν η **oscP5** .
- Πλήρης υποστήριξη από πολλές εφαρμογές ήχου. Συγκεκριμένα για τη πλατφόρμα εξεταζόταν διασύνδεση με τις εφαρμογές Max/MSP, Traktor και Reaktor, επίσης με το project IanniX και με τη γλώσσα προγραμματισμού Csound.
- Πιθανότητες διασύνδεσης με συσκευές που δεν έχουν απαραίτητα μόνο ηχητικές λειτουργίες. Συγκεκριμένα εξεταζόταν η σύνδεση με τον μικροεπεξεργαστή Arduino.
- Παραμετροποιήσιμα μηνύματα.

Επειδή όμως δε έγινε κάποια έμπρακτη υλοποίηση, η λειτουργία αυτή κατατάσσεται (και θα αναφερθεί) στο κεφάλαιο “Μελλοντική Εργασία & Επεκτάσεις”.

4.8 Σχεδιασμός Μηχανής Αντιστοίχησης

Όπως αναφέρθηκε παραπάνω, σαν input χρησιμοποιούνται gestures από τη κάμερα βάθους του Kinect. Συγκεκριμένα για να παράγουμε τα απαιτούμενα gestures αντλούμε δεδομένα από τη λειτουργία Skeleton Tracking. Η διαδικασία έχει ως εξής:

(1) Αρχικά δημιουργείται ένα αντικείμενο τύπου SimpleOpenNI, το οποίο ουσιαστικά αντιπροσωπεύει τη κάμερα.

SimpleOpenNI kinect; *kinect = new SimpleOpenNI(this);*

(2) Στη συνέχεια ενεργοποιούμε τον αισθητήρα βάθους ο οποίος θα μας προσφέρει τα δεδομένα τα οποία χρειαζόμαστε.

(3) Κατόπιν, ενεργοποιούμε το Skeleton Tracking το οποίο μας επιτρέπει να αντλήσουμε συντεταγμένες μόνο για τα σημεία του χρήστη τα οποία μας ενδιαφέρουν (χρησιμοποιώντας το user ID του). Εδώ αξίζει να σημειωθεί ότι είναι αρκετά ωφέλιμο να έχουμε ενεργοποιημένη τη λειτουργία για αυτόματο Calibration του χρήστη.

```
void GetInputs(int userID) {
  kinect.getJointPositionSkeleton(userID, SimpleOpenNI.SKEL_RIGHT_HAND, leftHand);
  kinect.getJointPositionSkeleton(userID, SimpleOpenNI.SKEL_LEFT_HAND, rightHand);
  kinect.getJointPositionSkeleton(userID, SimpleOpenNI.SKEL_RIGHT_ELBOW, leftElbow);
  kinect.getJointPositionSkeleton(userID, SimpleOpenNI.SKEL_LEFT_ELBOW, rightElbow);
  kinect.getJointPositionSkeleton(userID, SimpleOpenNI.SKEL_RIGHT_SHOULDER, leftShoulder);
  kinect.getJointPositionSkeleton(userID, SimpleOpenNI.SKEL_LEFT_SHOULDER, rightShoulder);
  kinect.getJointPositionSkeleton(userID, SimpleOpenNI.SKEL_LEFT_HIP, rightHip);
  kinect.getJointPositionSkeleton(userID, SimpleOpenNI.SKEL_RIGHT_HIP, leftHip);
  kinect.getJointPositionSkeleton(userID, SimpleOpenNI.SKEL_HEAD, head);
}
```

30: Joint Coordinates Used

Στο instance παραπάνω φαίνονται οι συντεταγμένες οι οποίες θα χρησιμοποιηθούν αργότερα. Η αναστροφή στον οριζόντιο άξονα (left-right) οφείλεται στο ότι το Kinect στέλνει δεδομένα από την οπτική γωνία της κάμερα, ενώ εμείς τα χειριζόμαστε από την οπτική γωνία του χρήστη.

(4) Σε αυτό το σημείο, μόλις εντοπίσει κάποιον χρήστη η κάμερα, πραγματοποιείται μία προσαρμογή στα φυσιομετρικά δεδομένα του. Αυτό είναι σημαντικό στοιχείο της μηχανής, καθώς γίνεται ουσιαστικά ορισμός του πεδίου δράσης που χρησιμοποιείται.

```
maxHeight = head.y*1.10f;

minHeight = ((leftHip.y-(leftHip.y*0.2))+(rightHip.y-(rightHip.y*0.2)))/2;

diff = dist(0,minHeight,0,maxHeight);

minWidth = leftShoulder.x-2.0*dist(leftShoulder.x, leftShoulder.y,
  leftShoulder.z, leftHand.x, leftHand.y, leftHand.z);

maxWidth = rightShoulder.x+2.0*dist(rightShoulder.x, rightShoulder.y,
  rightShoulder.z, rightHand.x, rightHand.y, rightHand.z);;

println("Initialized. Left: "+minWidth+"  Right: "+maxWidth+"  Top: "
  +maxHeight+"  Bottom: "+minHeight);
```

31: Custom Mapping Initialization

Στην παραπάνω εικόνα, διαφαίνεται ο τρόπος αρχικοποίησης της μηχανής αντιστοίχησης. Πρέπει να σημειωθεί ότι εσκεμμένα χρησιμοποιούνται συντελεστές πολλαπλασιασμού ως προς αποστάσεις συντεταγμένων του χρήστη , και όχι πράξεις πρόσθεσης/αφαίρεσης με απόλυτα νούμερα, ούτως ώστε η αρχικοποίηση να είναι πιο ακριβής για κάθε χρήστη (αντί για να είναι μετρημένη για συγκεκριμένους χρήστες).

(5) Τέλος, γνωρίζοντας τις συντεταγμένες που μας ενδιαφέρουν και έχοντας κάνει το Initialization που θέλουμε, μπορούμε να δημιουργήσουμε τα gestures που θέλουμε, και να παράγουμε μια μηχανή αντιστοίχησης ανάλογα με την λειτουργία που επιθυμούμε.

Να σημειωθεί ότι το OpenNI μας προσφέρει και δικιά του μηχανή για gestures η οποία έχει δυνατότητες αυτόματου εντοπισμού, ανάκλησης και αποθήκευσης αυτών. Παραταύτα στη πτυχιακή δε χρησιμοποιήθηκε.

Ο λόγος για τον οποίο έγινε αυτό είναι ότι ορίζοντας manually τα gestures που επιθυμούμε, είναι πιο εύκολο να φανεί ο τρόπος διασύνδεσης των τεχνολογιών τεχνητής όρασης και επεξεργασίας ήχου. Προφανώς, σε περίπτωση ανάπτυξης μίας πλατφόρμας ευρείας χρήσης, μια πιο εξελιγμένη προσέγγιση στον τρόπο αποθήκευσης/αναγνώρισης gestures, δε θα είναι απλά επιθυμητή, αλλά

απαραίτητη.

Στην επόμενη παράγραφο γίνεται ξεκάθαρο, πως δημιουργήθηκαν τα gestures και ενώθηκαν με τη μηχανή ήχου (που επεξηγήσαμε πρωτύτερα), πίσω από ένα κοινό interface, σχηματίζοντας τη τελική μορφή της πλατφόρμας.

4.9 Ένωση Επιμέρους Μηχανών (Ολοκλήρωση Πλατφόρμας)

Πρώτο βήμα για την ένωση των επιμέρους στοιχείων της εφαρμογής, είναι η δημιουργία ενός κοινού interface. Μέχρι τώρα δεν έχει γίνει αναφορά σε στοιχεία γραφικού περιβάλλοντος, καθώς δεν υπήρξε τέτοιο ζήτημα.

Η φύση της πλατφόρμας είναι τέτοια η οποία οδηγεί τον χρήστη να δέχεται ηχητικά ερεθίσματα, αντί για οπτικά, έτσι λοιπόν οι το γραφικό περιβάλλον εξ ορισμού επιβάλλεται να είναι απλό.

Η λιτότητα του περιβάλλοντος εργασίας όμως εξυπηρετεί και έναν δεύτερο σκοπό. Έχοντας ήδη τις ομολογουμένως βαριές μηχανές παραγωγής/επεξεργασίας ήχου, τεχνητής όρασης και αντιστοίχησης, η επιπλέον χρήση επεξεργαστικής ισχύς για κατασκευή σύνθετου περιβάλλοντος είναι το λιγότερο αφελής.

Τέλος, η ευκολία χρήσης, είναι μία από τις απαιτήσεις της εφαρμογής, η οποία μπορεί να ικανοποιηθεί δημιουργώντας ένα απέριττο περιβάλλον.

32: Το γραφικό περιβάλλον

Τα βασικά στοιχεία είναι τα εξής:

a) **Ένδειξη Λειτουργίας (Mode Indicator)**. Βρίσκεται στο πάνω μέρος του παραθύρου και δείχνει εάν ο χρήστης είναι σε Synthesis Mode (δημιουργίας ήχου) ή Filter Mode (επεξεργασίας ήχου).

b) **Κύρια Οθόνη (Monitor)**. Μιας και η εφαρμογή χρησιμοποιεί δεδομένα βάθους, στον κύριο χώρο της οθόνης, στο κέντρο, βρίσκεται μία ασπρόμαυρη αναπαράσταση (φωτεινότητα

αντιστρόφως ανάλογη της απόστασης από τον αισθητήρα) των τρισδιάστατων δεδομένων του αισθητήρα.

c) **Παράθυρο Πληροφοριών & Παράθυρο Μηνυμάτων (Info Screen & Message Screen)**. Σε αυτόν τον χώρο, ο οποίος βρίσκεται στο κάτω μέρος της εφαρμογής, εμφανίζονται πληροφορίες και μηνύματα σχετικά με τη λειτουργία της πλατφόρμας σε πραγματικό χρόνο.

Παραπάνω αναφέρθηκε ο Mode Indicator, μιας και πρόκειται για μία διαδραστική εφαρμογή που βασίζεται στην τεχνητή όραση, όταν ο χρήστης θέλει να αλλάξει mode, θα ήταν λίγο ανούσιο να γίνει αυτό από το πληκτρολόγιο.

Όπως προείπαμε, όταν γίνεται το Initialization της περιοχής του mapping, δημιουργείται ένα πλαίσιο (βασισμένο στη σωματική διάπλαση και τη τοποθέτηση του χρήστη) μέσα στο οποίο γίνεται η αναγνώριση των gestures. Χρησιμοποιώντας λοιπόν το πλαίσιο σαν μέσο και την αλλαγή mode σαν σκοπό, δημιουργείται το πρώτο gesture.

Πρόκειται για μία πολύ απλή, αλλά εξαιρετικά ουσιώδες λειτουργία. Η πλατφόρμα ξεκινάει από default σε Synthesis Mode. Όταν ο χρήστης θελήσει να αλλάξει και να περάσει σε Filter Mode, το μόνο που έχει να κάνει είναι να περάσει και τα δύο χέρια του πάνω από ύψος του κεφαλιού του (το οποίο είναι το άνω όριο του πλαισίου). Όταν θελήσει να ξαναγυρίσει στο Synethesis Mode πρέπει να χαμηλώσει τα χέρια του κάτω από το ύψος των γοφών του (κάτω όριο του πλαισίου). Το τελευταίο είναι και το gesture για το αρχικό Initialization.

Συνοψίζοντας, αυτές είναι οι δράσεις στις οποίες πρέπει να προβεί ο χρήστης μόλις τρέξει την εφαρμογή:

1. Να μπει στο πεδίο της κάμερας.
2. Να περιμένει μερικά δευτερόλεπτα για να ολοκληρωθεί το Calibration.
3. Να χαμηλώσει τα χέρια του κάτω από το ύψος των γοφών του για να γίνει το Initialization.
4. Όταν θελήσει να μπει στο Filter Mode, να σηκώσει τα χέρια του πάνω από το ύψος του κεφαλιού του.
5. Όταν θελήσει να επιστρέψει στο Synthesis Mode, να χαμηλώσει τα χέρια του κάτω από το ύψος των γοφών του.

Στο στιγμιότυπο μέσα από το πρόγραμμα της προηγούμενης σελίδας *(εικ. 1.)* μόλις έχει γίνει το Initialization – όπως μπορούμε να πληροφορηθούμε από το πεδίο εμφάνισης μηνυμάτων. Τα χέρια του βρίσκονται ακόμη στη θέση για switch σε λειτουργία σύνθεσης.

Όταν ο χρήστης βρίσκεται στο Synthesis Mode, αυτόματα ξεκινάει η παραγωγή ήχου. Κατά τη διάρκεια του testing, είχαμε το ποντίκι του υπολογιστή και ανάλογα με τη θέση του παραγωγή ήχου (X axis: pitch – Y axis: volume). Ακριβώς με τον ίδιο τρόπο λειτουργεί το mapping και τώρα. Η διαφορά είναι ότι αντί για μία κυματομορφή, έχουμε δύο (μία για κάθε χέρι – το οποίο λειτουργεί σαν δείκτης).

```
void UpdateWaveRight() {
  rightSine.setFreq(map(rightHand.x, minWidth, maxWidth, 0, 20000));
  rightSine.setAmp(map((maxHeight-rightHand.y), minHeight, maxHeight, 0, 1));
}
```

33: Right Hand SineWave Mapping

Παραπάνω βλέπουμε τον κώδικα για την αντιστοίχηση της θέσης του δεξιού χεριού με την κυματομορφή της, όμοια είναι η λειτουργία και για το αριστερό χέρι.

Η φιλοσοφία πίσω από αυτή την επιλογή (των δύο κυματομορφών) προέρχεται από διάφορα μουσικά όργανα. Όπως για παράδειγμα στο πιάνο ή στην άρπα, ο χρήστης χρησιμοποιεί το ένα χέρι για να κρατάει τον ρυθμό (συνήθως στο μπάσο εύρος), και με το άλλο να παράγει τη μελωδία, κάτι αντίστοιχο μπορεί να γίνει και εδώ.

Αν από την άλλη χρειάζεται να παράγει μόνο ένα κύμα, αρκεί να βγάλει το ένα του χέρι εκτός των ορίων του πλαισίου για να γίνει mute, και να συνεχίσει με το άλλο. Τότε έχουμε έναν ήχο ο οποίος είναι παρόμοιος με αυτόν που παράγει ένα Theremin.

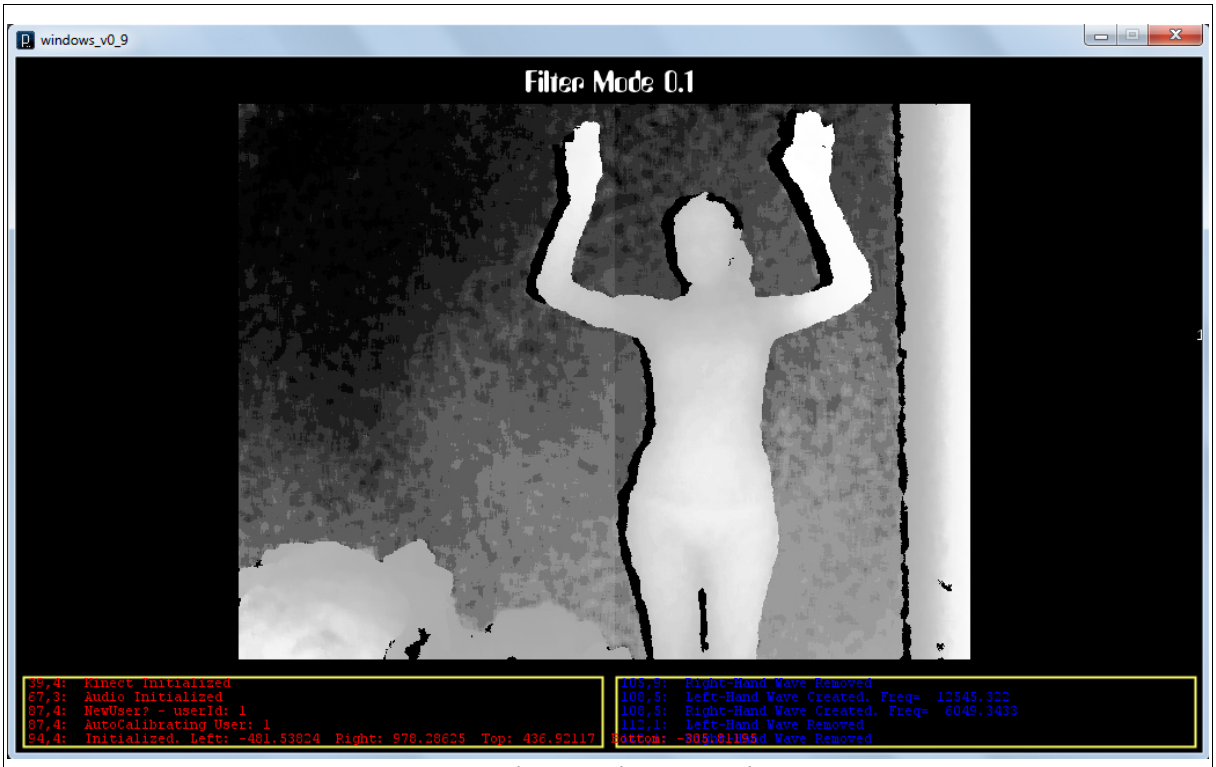
Τέλος, εαν ο χρήστης θελήσει να σταματήσει ο ήχος, χωρίς να αλλάξει mode, μπορεί να κάνει mute και τα δύο ηχητικά σήματα, αρκεί προφανώς να μην γίνουν και τα δύο από το άνω όριο του πλαισίου (gesture για αλλαγή σε Filter Mode).

```
if (leftHand.x<minWidth||leftHand.y<minHeight) {
  if (leftOn==true)  DeleteWaveLeft();
}

if (rightHand.x>maxWidth||rightHand.y<minHeight) {
  if (rightOn==true)  DeleteWaveRight();
}

if (leftOn==false) {
  if (leftHand.x>minWidth&&leftHand.y>minHeight)  CreateWaveLeft();
}
else  UpdateWaveLeft();

if (rightOn==false) {
  if (rightHand.x<maxWidth&&rightHand.y>minHeight)  CreateWaveRight();
}
else UpdateWaveRight();

//switch to Filter Mode
if (rightHand.y>maxHeight&&leftHand.y>maxHeight)  CreateBandPassFilter();
```

34: Synthesis Mode Functions

Με αντίστοιχο τρόπο έγινε και η σύνδεση για το Filter Mode. Όπως βλέπουμε στο στιγμιότυπο κώδικα παραπάνω, όταν εντοπιστεί το gesture για λειτουργία φίλτρου, δημιουργείται ένα φίλτρο ζώνης, αφού πρώτα έχουν διαγραφεί τα δύο κύματα από το Synthesis Mode.

Αυτό μπορούμε να το δούμε στη πράξη και στο στιγμιότυπο που ακολουθεί. Ο χρήστης είναι σε θέση για αλλαγή σε Filter Mode. Μπορούμε να δούμε ότι η αλλαγή έχει ήδη γίνει (από τον τίτλο-ένδειξη). Κοιτάζοντας το panel με τις πληροφορίες κάτω-δεξιά βλέπουμε ότι και τα δύο ημιτονοειδή κύματα του Synethsis Mode έχουν απενεργοποιηθεί.



35: Filter Mode Screenshot

Όπως βλέπουμε και στη συνάρτηση δημιουργίας του φίλτρου ζώνης ξεκινάει να παίζει το "PWSteal.Ldpinch.D" όπως το είχαμε ορίσει νωρίτερα. Στη συνέχεια δημιουργείται το φίλτρο ζώνης, παίρνοντας σαν παραμέτρους τη κεντρική συχνότητα, το εύρος συχνότητας του φίλτρου και τον ρυθμό δειγματοληψίας. Τέλος, το ενεργοποιούμε στο ίδιο αντικείμενο με το οποίο γίνεται η αναπαραγωγή.

```
//FILTER MODE
void CreateBandPassFilter() {
  player.play();
  bpFilter = new BandPass(bandFrequency, bandWidth, player.sampleRate());
  player.addEffect(bpFilter);
  kinectSynth=false;
  enqueueInfo("Synth Mode Terminated");
  enqueueMsg("Filter Mode Initiated");
}
```

36: Create BandPass Filter Function

Από τη στιγμή λοιπόν που θα δημιουργηθεί το φίλτρο, ανανεώνονται συνεχώς οι παράμετροι του (από τη μηχανή αντιστοίχισης), μέχρις ότου εντοπιστεί το Gesture για επιστροφή στο Synthesis Mode.

```
}
else if(!kinectSynth) {//case: filter mode
  if (leftHand.y<minHeight&&rightHand.y<minHeight) DeleteBandPassFilter();
  else UpdateBandPassFilter();
}
```

37: Filter Mode Functions

Ο τρόπος που γίνεται το mapping είναι αρκετά απλός. Ο χρήστης γνωρίζοντας ότι η ενεργός περιοχή είναι από το ύψος του κεφαλιού του, μέχρι το ύψος των γοφών του μπορεί να υποθέσει ότι εκεί βρίσκονται όλες οι συχνότητες. Στη συνέχεια "δείχνοντας" με τα χέρια του ποιες συχνότητες θέλουν να περάσουν αυτές περνάνε. Για παράδειγμα έαν θελήσει να περνάνε οι μέσες συχνότητες μπορεί να έχει τα χέρια του ανοικτά σε μία περιοχή κάτω από το στέρνο κ.ο.κ. Αν θελήσει να περνάνε όλες οι συχνότητες μπορεί να ανοίξει τα χέρια του ούτως ώστε να περικλείουν όλη τη περιοχή, αν θέλει να κάνει mute μπορεί πολύ απλά να κλείσει τα χέρια του.

Προγραμματιστικά για να επιτευχθεί αυτό, παίρνουμε τη σχετική θέση κάθε χεριού (ως προς την ενεργό περιοχή) και της κάνουμε μία αντιστοίχηση με βάση τις ακουστικές συχνότητες. Στη συνέχεια μετράμε την απόσταση τους, και έτσι καθορίζουμε το εύρος του φίλτρου, ενώ ανάλογα με τη θέση τους καθορίζεται η κεντρική συχνότητα του φίλτρου.

4.10 Ανάλυση και Βελτιστοποίηση Αλγορίθμων

- Παρατηρήθηκε ότι με δυναμικό (real-time) ορισμό "πλαισίου", υπήρχε πιο ακριβές mapping, αλλά ξοδευόταν πολύ επεξεργαστική ισχύ, προκαλώντας καθυστερήσεις, γι αυτό και το Initialization, γίνεται μία φορά και οι συντεταγμένες παραμένουν σταθερές.
  Αν όμως ο χρήστης θελήσει να ξανακάνει Initialize (είτε επειδή δεν έγινε σωστά, είτε επειδή μετακινήθηκε) έχει αυτή τη δυνατότητα – πατώντας το πλήκτρο "I".

- Κατά τη διάρκεια δημιουργίας διάφορων effect, δοκιμάστηκαν διάφοροι τρόποι για να αυξηθεί η ταχύτητα υπολογισμού, ούτως ώστε να μη προκύπτει clipping, παραταύτα κανείς δεν ήταν αρκετά αποδοτικός (ακόμη και η χρήση thread).

- Επίσης στα effects δοκιμάστηκαν samples διαφορετικού μεγέθους χωρίς όμως εμφανή διαφορά.

- Όταν ο χρήστης βρισκόταν αρκετά μακρυά από την κάμερα (κοντά στο όριο) υπήρχαν κάποια εσφαλμένα δεδομένα. Αυτό δεν επηρέαζε τόσο στον ήχο καθώς ήταν πολύ αραιά, όσο στην αλλαγή modes. Το πρόβλημα αυτό λύθηκε ενσωματώνοντας ένα buffer μεγέθους 2 frames το οποίο χρησιμοποιήθηκε για διασταύρωση συντεταγμένων.

- Δοκιμάστηκαν και άλλοι ρυθμοί ανανέωσης της συνάρτησης draw(). Όμως ο πιο αποδοτικός ήταν ο default ρυθμός ανανέωσης των 30Hz.

- Επίσης δοκιμάστηκε event-driven μοντέλο προγραμματισμού, στο οποίο οι λειτουργίες του Kinect τρέχανε σε ξεχωριστό thread. Δεν παρατηρήθηκε σημαντική διαφορά στην απόδοση του προγράμματος.

- Τεχνικές mapping οι οποίες αξιοποιούσαν όλα τα δεδομένα θέσης (όχι μόνο αυτά από τη μέση και πάνω) σχεδιάστηκαν, αλλά δεν υλοποιήθηκαν σε αυτό το στάδιο.

- Στον αρχικό σχεδιασμό του interface υπήρχαν περισσότερα δυναμικά γραφικά. Για παράδειγμα ένα "κουτί" ακολουθούσε τη θέση κάθε χεριού και εμφάνιζε σχετικές πληροφορίες. Το αποτέλεσμα παρόλο που αισθητικά ήταν πιο εντυπωσιακό, πρακτικά κούραζε τον χρήστη μετά από λίγη ώρα χρήσης της πλατφόρμας και προκαλούσε περιττές επιβαρύνσεις στο σύστημα.

Ένα βίντεο επίδειξης της πλατφόρμας στη τελική μορφή της είναι διαθέσιμο στον εξής σύνδεσμο: http://vimeo.com/convoitise/thesis

Σύγκριση μεταξύ C# - Kinect SDK – XNA και Processing – SimpleOpenNI - P3D

Κατά τη διαδικασία αξιολόγησης υλικού προς επιλογή για προγραμματισμού της πλατφόρμας έγινε μία δίμηνη μελέτη για επικείμενη ανάπτυξη με διαφορετική προσέγγιση (η οποία πραγματοποιήθηκε στα εργαστήρια D.I.R.O. του Πανεπιστημίου του Μόντρεαλ). Συγκεκριμένα σχεδιάστηκε και υλοποιήθηκε πανομοιότυπη μηχανή με τα εξής εργαλεία:

- **C#** (Programming Language) – αντί Processing.
- **Microsoft Kinect SDK** (Kinect Development Libraries) – αντί SimpleOpneNI
- **XNA** (Graphics Renderer) – αντί P3D
- **DirectSound** (Audio Library) – αντί Minim

Κατά τη διάρκεια της ανάπτυξης παρατηρήθηκαν σημαντικές διαφορές στα εξής σημεία:

- Γλώσσα Προγραμματισμού: Η C# έχει ακόμη μία συνάρτηση (πέραν των draw(), setup() της Processing), την update().
- Βιβλιοθήκες Kinect: Το SDK της Microsoft έχει προσανατολιστεί στο event-driven μοντέλο. Έτσι, λειτουργεί καλύτερα όταν στέλνει το Kinect δεδομένα στην εφαρμογή (αντί για να αντλεί η εφαρμογή από το Kinect, όπως είναι το SimpleOpenNI).
- Graphics Renderer: Το XNA χρησιμοποιεί τρεις πίνακες για απεικόνιση και δουλεύει περισσότερο με μοντέλα. Στο P3D η μηχανή δουλεύει κρυφά από τον χρήστη, και είναι επικεντρωμένη στα Vector Graphics
- Audio Library: Η DirectSound είναι εκτός από βιβλιοθήκη ολοκληρωμένο Studio διαχείρισης ήχου.

Τελικά η πλατφόρμα δεν προχώρησε παραπέρα καθώς το DirectSound ήταν δύσχρηστο, χρονοβόρο και ελαφρώς παλαιάς τεχνολογίας. Παραταύτα γίνανε κάποιες δοκιμές χρησιμοποιώντας για ήχο την SoundEffect class του XNA. Τα συμπεράσματα που προκύπτουν (σε σύγκριση με την παρούσα υλοποίηση πάντα) είναι εξαιρετικά ενδιαφέροντα. Παρακάτω παρατίθενται κάποια σχόλια και παρατηρήσεις - αποτελέσματα μίας σύγκρισης των δύο υλοποιήσεων.

Προτού καν ξεκινήσει το προγραμματιστικό μέρος της ανάπτυξης, εντοπίστηκαν σημαντικές διαφορές. Η ετοιμασία του workstation χρησιμοποιώντας το SimpleOpenNI ήταν πολύ πιο χρονοβόρα και δύσκολη από ότι με τους Drivers/SDK της Mircrosoft. Αυτή η δυσχέρεια γινόταν ακόμη πιο έντονη (σε πολύ άσχημο βαθμό) όταν το setup έγινε σε Linux. Ακόμη και μετά τη σωστή εγκατάσταση/ρύθμιση υπήρχαν κάποιες (ασήμαντες ευτυχώς) ασυμβατότητες με το λειτουργικό σύστημα και άλλα προγράμματα που χρησιμοποιούσαν τη κάμερα.

Αυτό ήταν αναμενόμενο μίας και τα Windows, Kinect, Kinect SDK, Kinect Drivers, C#, XNA είναι όλα προϊόντα της Microsoft. Όταν όμως έγινε το setup η ευκολία με την οποία μπορεί να ξεκινήσει το development είναι ίδιου επιπέδου.

Στη συνέχεια έγιναν οι προκαταρκτικές μετρήσεις και αναλύσεις (χρησιμοποιώντας μόνο το Kinect – χωρίς παραγωγή ήχου). Προέκυψαν τα εξής:

- **Η Processing επιτρέπει εξαιρετικά ταχεία ανάπτυξη εφαρμογών, ενώ η C# εξαιρετική σταθερότητα.** Η C# διαχειρίζεται με εξαιρετική ταχύτητα και ευκολία -για τα δεδομένα αντικειμενοστραφούς γλώσσας- σύνθετες μαθηματικές πράξεις, στις οποίες η Processing χωλαίνει. Για πιο απλές εφαρμογές όμως μπορεί να γίνει ανάπτυξη σε κλάσμα του χρόνου από ότι θα απαιτούνταν σε C#. Επιπρόσθετα, ένας άπειρος χρήστης μπορεί να ξεκινήσει την ανάπτυξη σχεδόν αμέσως.
- **Το Processing IDE προσφέρει μηδαμινές λειτουργίες μπροστά στο Microsoft Visual C# 2010 Express IDE.** Αυτό ήταν αναμενόμενο γνωρίζοντας τα παραπάνω. Δεν είναι τυχαίο που τα projects στη Processing ονομάζονται Sketches, ενώ στο Visual C#, solutions. Προς αποφυγή παρεξηγήσεων όμως, το IDE της Processing είναι εξαιρετικά απέριττο και αυτό

μπορεί να αποτελέσει πολύ σημαντικό πλεονέκτημα για μικρά projects ή για άπειρους developers. Παραταύτα όταν ολοκληρωθεί η ενσωμάτωση της Processing στο IDE της Eclipse θα υπάρχει η πολύ χρήσιμη εναλλακτική του ολοκληρωμένου IDE αντί του Sketchpad.

- **Το Kinect SDK είναι πολύ πιο γρήγορο από το SimpleOpenNI και προσφέρει ποικίλες επιπρόσθετες λειτουργίες.** Αυτό δε σημαίνει απαραίτητα ότι το SimpleOpenNI είναι "φτωχό" ή αργό. Σχετικά με τη ταχύτητα, το event-driven μοντέλο του Kinect SDK επιτρέπει πιο άμεση απόκριση και διαχείριση στα εισερχόμενα δεδομένα από το Kinect. Από την άλλη το περιοδικό σύστημα συλλογής του SimpleOpenNI καθιστά ποιο εύκολο τον προγραμματισμό και την a-priori ανάλυση του προγράμματος.
  Όσο αφορά τις λειτουργίες, το SimpleOpenNI έχει εξελιχθεί πάρα πολύ στις τελευταίες εκδόσεις του και έχει φτάσει το επίπεδο του Kinect SDK, σε θέμα διαχείρισης RGB και Depth cameras. Όμως το Kinect SDK προσφέρει πολλές επιπρόσθετες περιφερειακές λειτουργίες όπως για παράδειγμα την ηχητική ανάλυση του σήματος από το microphone array, αναγνώριση φωνητικών εντολών σε διάφορες γλώσσες (σε συνδυασμό με το Microsoft Speech) κ.α.
- **Το XNA προσφέρεται για βαριά γραφικά, ενώ το P3D για δυναμικά – γρήγορα.** Ακολουθώντας τη φιλοσοφία της αντίστοιχης γλώσσας προγραμματισμού οι μηχανές γραφικών τους έχουν αντίστοιχες προτεραιότητες. Έτσι, το XNA είναι μονόδρομος για χρήση πολλών/σύνθετων μοντέλων, ενώ το P3D λύνει τα χέρια σε περίπτωση που ο χρήστης θέλει να έχει γρήγορα/δυναμικά γραφικά. Μία εναλλακτική, ενδιάμεση λύση για Processing είναι η χρήση της βιβλιοθήκης OpenGL.

Το συμπέρασμα που προκύπτει, σε σχέση πάντα με το αντικείμενο της πτυχιακής, είναι ότι ο συνδυασμός που χρησιμοποιείται είναι ο βέλτιστος όταν αφορά εφαρμογές για προτυποποίηση ή proof-of-conceprs όπου δίνεται έμφαση στην ιδέα που παρουσιάζεται. Αντίστοιχα, είναι πολύ χρήσιμος για πολυμεσικές διαδραστικές εφαρμογές. Επίσης, είναι μονόδρομος όταν θέλουμε να έχουμε platform-indepented εφαρμογές ή εφαρμογές που να μπορούν να τρέξουν σε λειτουργικό Linux. Τέλος, αποτελεί μία παρα πολύ αξιόπιστη λύση για καλλιτεχνικές εφαρμογές όπου απαιτείται ευελιξία και τις περισσότερες φορές ο προγραμματιστής έχει μικρή εμπειρία σαν developer.

Από την άλλη, το πακέτο της Microsoft είναι εξαιρετικό σε περιπτώσεις όπου έχει σημασία η ακρίβεια και η ισχύς. Χαρακτηριστικά, δεν υπάρχει άλλη λύση όταν μιλάμε για ανάπτυξη παιχνιδιών με αξιοπρεπή τρισδιάστατα γραφικά. Μία άλλη κατηγορία εφαρμογών στην οποία συγκεντρώνει περισσότερα θετικά χαρακτηριστικά από ότι το πακέτο που χρησιμοποιείται, είναι όταν έχουμε επιστημονικά προγράμματα. Έτσι, όταν απαιτείται ανάλυση-μοντελοποίηση μαζικών δεδομένων, ανάπτυξη ευφυών συστημάτων κλπ, σε συνδυασμό με mapping από το Kinect και απεικόνιση τρισδιάστατων γραφικών , η λύση C#/XNA/KinectSDK είναι η προτεινόμενη.

9 Αποτελέσματα (Test Results Analysis)

Στο τελευταίο μέρος θα καταγραφούν παρατηρήσεις σχετικές με τη πλατφόρμα και θα διατυπωθούν προοπτικές σχετικά με μελλοντική εξέλιξη της και τυχών προεκτάσεις της.

9.1 Συμπεράσματα

Από τη στιγμή που ανέλαβα την διεκπεραίωση της πτυχιακής αυτής εργασίας, μέχρι ότου να φτάσω σε αυτό το σημείο ολοκλήρωσης της, η μορφή της άλλαξε δραστικά, αρκετές φορές. Από το αρχικό θέμα το οποίο ήταν η μετατροπή στατικής δισδιάστατης εικόνας σε ήχο - χρησιμοποιώντας το MatLab, εξελίχθηκε σε ανάπτυξη πλατφόρμας για δυναμική μετατροπή τρισδιάστατων δεδομένων με σκοπό τη σύνθεση και την επεξεργασία ήχου. Πιστεύω ότι όλο αυτό οδήγησε στην δημιουργία μίας εφαρμογής η οποία δεν είναι απλά άλλη μία πτυχιακή εργασία, αλλά η αρχή για κάτι καινούργιο.

Αποδείχθηκε ότι η αντιστοίχιση δεδομένων βάθους και η μεταποίηση τους σε ηχητικά δεδομένα είναι δυνατή, και μάλιστα με όσους τρόπους, όσους μπορεί να φανταστεί ο προγραμματιστής. Πέραν τούτου, ξεδιπλώθηκαν και άλλες προοπτικές, οι οποίες δεν ενσωματώθηκαν σε αυτή τη πλατφόρμα, αλλά είναι εξίσου πρωτοποριακές και υλοποιήσιμες, όσο και αυτές που αναπτύχθηκαν στα πλαίσια της πτυχιακής.

Η χρήση της εξαιρετικά διαδεδομένης κάμερας Kinect έδειξε ότι αντίστοιχες εφαρμογές, όσο φουτουριστικές και αν φαντάζουν στον απλό χρήστη, μπορούν να βρεθούν σπίτι του από τη μία μέρα στην άλλη. Επιπρόσθετα η ανάπτυξη στη γλώσσα προγραμματισμού Processing, είναι ενδεικτική για την απουσία περιορισμών από μεριάς ανάπτυξης εφαρμογών.

Βλέπουμε λοιπόν ότι εφαρμογές τέτοιου τύπου, μπορούν να βρεθούν από εργαστήρια ανάπτυξης τεχνολογιών, μέχρι το το σαλόνι μας και αίθουσες συναυλιών.

9.2 Μελλοντική Εργασία & Επεκτάσεις

Η πλατφόρμα αυτή είναι από τη φύση της ένα project το οποίο αποσκοπεί στο να θέσει βάσεις για ανάπτυξη νέων λειτουργιών στον τομέα. Συγκεκριμένα η ίδια η πλατφόρμα μπορεί να δειχτεί τις ακόλουθες προσθήκες:

- **Audio Interface Mode:** Λεπτομερέστερη περιγραφή βρίσκεται παραπάνω (στη παράγραφο όπου αναφέρεται η λειτουργία των Modes της εφαρμογής), καθώς σχεδιάστηκε, έγινε μελέτη και δοκιμές, αλλά δεν υλοποιήθηκε.

- **Τεχνικές Βελτίωσης Αντιστοίχησης (Mapping Improvement Technicques):** Θα μπορούσαν να γίνουν προσθήκες στις τεχνικές αντιστοίχησης. Ακολουθούν μερικές προτάσεις:
  - Θα μπορούσε να γίνεται mapping μέσω του Skeleton Tracking (όπως τώρα), και συμπληρωματικό mapping (ή Initialization/Calibration) μέσω ανάλυσης του περιβάλλοντος χώρου, χρησιμοποιώντας τα Point Clouds.
  - Αναγνώριση σφαλμάτων κατά τη διάρκεια του Skeleton Tracking χρησιμοποιώντας τεχνικές τεχνητής όρασης με δεδομένα από την RGB camera του Kinect.
  - Βελτιστοποίηση των μηχανών Gesture Recognition / Αντιστοίχησης με ενσωμάτωση νευρωνικών δικτύων.
  - Αποθήκευση φυσιομετρικών δεδομένων χρήστη και αυτόματη ανάκληση τους όταν χρησιμοποιείται η πλατφόρμα από τον ίδιο χρήστη. Κατα αυτόν τον τρόπο θα μπορούσε να γίνεται ένα ακριβέστατο Calibration/Initialization την πρώτη φορά που χρησιμοποιεί ο χρήστης τη πλατφόρμα (και ας είναι πιο χρονοβόρο) και να χρησιμοποιεί αυτά τα δεδομένα κάθε φορά. Ειδικά σε συνδυασμό με νευρωνικό δίκτυο αυτό θα προσέδιδε τρομακτική ακρίβεια και ευκολία χρήσης στη πλατφόρμα.

- **Alternative Mapping:** Πέρα της προκαθορισμένης τεχνικής αντιστοίχησης θα μπορούσε να υπάρχει και κάποια εναλλακτική. Αυτό συμπεριλαμβάνει από κάτι απλό, όπως αντί για γραμμική αντιστοίχηση θέσης χεριού – έντασης ήχου, να έχουμε λογαριθμική, μέχρι κάτι πιο εξεζητημένο, όπως παραγωγή ήχου με συγχρονισμό από beat detection στη θέση του ποδιού.
  Μια εναλλακτική έκφραση του Alternative Mapping θα ήταν, σε μεταγενέστερο στάδιο, ο χρήστης να ορίζει δικά του Gestures πάνω στα οποία να γίνεται η αντιστοίχηση.

- **Υποστήριξη Πολλαπλών Χρηστών:** Σε κάποια μεταγενέστερη έκδοση της πλατφόρμας θα μπορούσε να υπάρξει λειτουργία που να υποστηρίζει περισσότερους του ένα χρήστη. Το πόσους χρήστες θα μπορεί να υποστηρίξει η πλατφόρμα είναι βέβαια θέμα φαντασίας του προγραμματιστή, αλλά και οπτικού πεδίου της συσκευής, επεξεργαστικής ισχύς καθώς και βιβλιοθήκης που χρησιμοποιείται.

- **Mixed Mode:** Μία προσθήκη (ειδικά αργότερα όπου θα υπάρχουν περισσότερα modes, με περισσότερες λειτουργίες) θα ήταν ένα συνδυαστικό mode στο οποίο ο χρήστης θα μπορεί να πραγματοποιεί διαφορετικού τύπου αντιστοιχήσεις ταυτόχρονα. Για παράδειγμα θα μπορούσε να είναι στο Effects mode και με το ένα χέρι να χειρίζεται παραμετροποίηση του τρέχοντος effect, με το άλλο χέρι να δημιουργεί δικό του ήχο, ενώ με τη θέση του σώματος του να αλλάζει ένταση στο κομμάτι.

- **Interface:** Παρόλο που το τρέχον περιβάλλον εργασίας της εφαρμογής είναι επαρκές για τις λειτουργίες που προσφέρονται, είναι αυτονόητο ότι όσο αυξάνονται οι λειτουργίες θα χρειάζονται και προσθήκες στο γραφικό περιβάλλον. Για παράδειγμα, θα μπορούσαν να εμφανίζονται οι τρέχουσες επιλογές οι οποίες έχει ο χρήστης ή δεδομένα από άλλες πηγές .
  Πέραν τούτου όμως, θα μπορούσαν να γίνουν διάφορες αισθητικές προσθήκες. Μία από τις οποίες είχε μελετηθεί να υλοποιηθεί στην υπάρχουσα πλατφόρμα είναι στο Filter Mode να εμφανίζεται η κυματομορφή του ηχητικού δείγματος που παίζει και πάνω σε αυτή η περιοχή πάνω στην οποία λειτουργεί το φίλτρο, ενώ για Synthesis Mode ένα instance του ημιτονοειδές κύματος που δημιουργείται.

- **Effects Mode:** Ενώ στη παρούσα πλατφόρμα το effects mode δεν υλοποιήθηκε για λόγους που αναφέραμε σε προηγούμενη παράγραφο, μετά από όλες αυτές τις δοκιμές, είμαι πεπεισμένος ότι μπορεί να σχεδιαστεί και να ενσωματωθεί στη πλατφόρμα. Δυστυχώς απαιτείται εκτενέστερη ανάλυση και διασύνδεση με λειτουργίες προγραμματισμένες σε χαμηλότερο επίπεδο, για τα οποία δεν υπήρξε χρόνος να γίνουν.

Πέραν όμως τις προαναφερθείσες -και άλλες- προσθήκες που θα μπορούσε να έχει η πλατφόρμα, θα μπορούσαν να αναπτυχθούν τελείως ανεξάρτητες εφαρμογές με βάση τη μελέτη που έχει γίνει. Παρακάτω αναφέρονται κάποια παραδείγμα, ταξινομημένα ως προς τις κατηγορίες/εφαρμογές που προτάθηκαν στην παράγραφο με τις απαιτήσεις συστήματος.

Καλλιτεχνικής Φύσης

- Εκμεταλλευόμενοι τις λειτουργίες του Minim και ανάλογα με τις απαιτήσεις μπορούν να δημιουργηθούν διάφορες εφαρμογές καθαρά για μουσικούς σκοπούς. Χρησιμοποιώντας το Beat Detection θα μπορούσε να υπάρξει ένας εικονικός μίκτης με μηχανή effects δημιουργώντας διαδραστικές εφαρμογές για DJing. Με τις ικανότητες αναπαραγωγής grooves, ένα sampler θα μπορούσε να χρησιμεύσει σε ζωντανές εμφανίσεις παραγωγών ηλεκτρονικής μουσικής. Γενικεύοντας, σχεδόν κάθε μουσικό όργανο θα μπορούσε να έχει μία εικονική ψηφιακή αναπαράσταση η οποία θα χειρίζεται από το Kinect.

- Έχοντας οποιαδήποτε διαδραστική πλατφόρμα αντιστοίχησης μουσικής φύσης, κάνοντας

παράλληλο mapping - χρησιμοποιώντας τα ίδια δεδομένα και εκμεταλλευόμενοι τις ποικίλες βιλιοθήκες γραφικών που προσφέρονται στη Processing, θα μπορούσε ο χρήστης να έχει live visualisations του έργου του. Οι πιθανές υλοποιήσεις αυτού του πλάνου είναι πραγματικά αναρίθμητες.

- Έχοντας καλλιτέχνες οι οποίοι δίνουν οποιουδήποτε τύπου παράσταση (μουσική, θεατρική κλπ), μια εφαρμογή θα μπορούσε να διαβάζει διαφορετικά οπτικά ερεθίσματα στον χώρο και να δημιουργεί ένα ηχητικό τοπίο ούτως ώστε να την εμπλουτίζει.

- Χρησιμοποιώντας το πρωτόκολλο OSC και τις ικανότητες διασύνδεσης που αυτό προσφέρει, θα μπορούσε να γίνει mapping ούτως ώστε να υπάρξει εξομοίωση διαφορετικών ηχητικών controllers και MIDI Interfaces.

Τηλεπικοινωνιακές

- Άτομα τα οποία έχουν προβλήματα ομιλίας θα μπορούσαν σε μία ειδικά διαμορφωμένη πλατφόρμα, να έχουν ηχητική αναπαραγωγή λέξεων, χρησιμοποιώντας νοηματική η οποία θα "διαβάζεται" από τα τρισδιάστατα δεδομένα.

- Χρησιμοποιώντας πρωτόκολλα και τεχνικές για συμπίεση/μεταφορά ήχου θα μπορούσε να υπάρξει αντίστοιχη συμπεριφορά σε gestures για διασύνδεση τελείως διαφορετικών εφαρμογών.

Επεξεργασίας Εικόνας-Ήχου

- Χρησιμοποιώντας μια κάμερα βάθους σε έναν χώρο όπου χορεύει πολύς κόσμος, θα μπορούσε να εκπαιδευτεί ένα νευρωνικό από τις κινήσεις τους, ούτως ώστε να αναγνωρίζει στοιχεία στη μουσική τα οποία αρέσουν περισσότερο στο κοινό και σε δεύτερο στάδιο να βρει εφαρμογή σε live επεξεργασία ήχου κατά τέτοιο τρόπο ώστε να εφαρμόζονται αυτά τα στοιχεία, όταν παρουσιάζονται τα κατάλληλα οπτικά ερεθίσματα.

- Έχοντας σε ένα χώρο μια ηχητική εγκατάσταση (ομιλία, συνέδριο, συναυλία κλπ) με ανίχνευση του κόσμου (πλήθος και τοποθεσία) αυτόματη διαμόρφωση του ήχου. Για παράδειγμα εαν μαζεύεται περισσότερος κόσμος ή είναι απλωμένος σε μεγαλύτερη έκταση αύξηση της έντασης ή αν είναι κοντά στα ηχεία αύξηση στις υψηλές συχνότητες (οι οποίες είναι πιο ευπαθής σε αλλοιώσεις).

Βιβλιογραφία

[1] Jo Kazuhiro, Nagaro Norihisa, “Monalisa: see the sound, hear the image”
[2] Zune Lee, Jonathan Berger, Woon Seung Yeo, “Mapping Sound to Image in Interactive Multimedia Art”
[3] Jonathan Berger, Woon Seung Yeo, “Application of Raster Scanning Method To Image Sonification, Sound Visualization, Sound Analysis and Synthesis”. Center For Computer Research in Music and Acoustics, Stanford University CA
[4] Greg Borenstein, “Making Things See: 3D Vision with Kinect, Processing, Arduino and Makerbot”. O'REILLY

Λογισμικό

Development:
Processing: www.processing.org
Minim Library: http://code.compartmental.net/tools/minim/
OpenGL: http://www.opengl.org/
SimpleOpenNI: http://code.google.com/p/simple-openni/

Image-Sound Mapping:
RGB MusicLab: www.kenjikojima.com
vOICe: www.seeingwithsound.com
MetaSynth: http://www.uisoftware.com/MetaSynth/

Sound Edit/Analysis:
Audacity: http://audacity.sourceforge.org
Adobe Audition: www.adobe.com/products/audition.html

Image Edit/Analysis:
GIMP: www.gimp.org
Adobe Photoshop: www.adobe.com/products/photoshop.html

Other
House of Cards: http://code.google.com/p/radiohead/

Παράρτημα

Tables

i. Processing Sketch Sections

ii. Custom Effects Tested

Code Snippets

a. Global Variables Used

b. Imports

c. Custom initialization function used in setup()

d. SimpleOpenNI events (modified)

e. Display Functions

f. Print on Screen Functions

g. Keyboard Inputs Handling

h. BandPass Filter Functions

I. Sinewave Audio Functions

j. Audio Objects Created

Presentation

TITLE	ROLE
Main	Imports & Declarations
Global	Global Variables Decleration
Draw()	
Setup()	
Audio	Audio Functions
Audio Saw	SawWave Generator
Inputs	Input Handler
Kinect	Kinect Handler & Functions
Screen	Display Management
Support	Support Functions

i.

a. Global Variables Used

```
//---GENERAL
//--Mode
boolean isSynth=false;          //Current Mode
boolean kinectSynth=true;      //Kinect Mode
//--Text
String[] infoText = new String[6];
String[] msgText = new String[6];
String supportText, tempText;

//---AUDIO
//--Synth
boolean isSine = false;
boolean isSaw = false;
boolean isPulse = false;
boolean waveOn = false;        //Wave Active

//--Player
boolean isPlaying = false;     //Currently Playing Audio

//--Effects
boolean isLowPass = false;

//---KINECT
//--Setups
float zoom = 0f;
float rotationX = radians(180);   //OpenNI sends data rotated by 180deg
around X
float rotationY = radians(0);    //OpenNI send data normal on Y
boolean autoCalibration = true;
//--PlayerData
PVector bodyCenter = new PVector();
PVector bodyDirection = new PVector();
PVector rightHand = new PVector();
PVector leftHand = new PVector();
PVector rightElbow = new PVector();
PVector leftElbow = new PVector();
PVector rightShoulder = new PVector();
PVector leftShoulder = new PVector();
PVector rightHip = new PVector();
PVector leftHip = new PVector();
PVector head = new PVector();
//--Initialization
boolean isInitialized = false;
float handRadius = 0;
float xAxis = 0;
float yAxis = 0;
float zAxis = 0;
float maxHeight, minHeight;
float maxWidth, minWidth;
float diff;

//Support Band-Synth
boolean switched=false;

//Tests
int testCount = 0;
boolean oscillatorOn=false;
boolean oscillatorEffectOn=false;
```

b. Imports

```
//import processing.opengl.*;

import net.nexttext.behaviour.dform.*;
import net.nexttext.behaviour.*;
import net.nexttext.behaviour.control.*;
import net.nexttext.behaviour.physics.*;
import net.nexttext.renderer.*;
import net.nexttext.*;
import net.nexttext.property.*;
import net.nexttext.behaviour.standard.*;
import net.nexttext.input.*;

import ddf.minim.*;
import ddf.minim.signals.*;
import ddf.minim.analysis.*;
import ddf.minim.effects.*;

import SimpleOpenNI.*;




//Create OpenNI  (object used for kinect)
SimpleOpenNI kinect;

//Create Minim  (object for sound)
Minim minim;
```

ii. Custom Effects Tested

Oscillator	Διαφοροποίηση έντασης ανάλογα με το πλάτος ενός δεύτερου ημιτονοειδές σήματος (ορισμένο κατόπιν mapping)
Delay	Επανάληψη ενός sample μετά από κάποιο διάστημα. Η παράμετροι που ορίζοταν από το mapping είναι το μέγεθος του sample και ο χρόνος επανάληψης
Echo	Πανομοιότυπο με το Delay. Τα samples επαναλαμβάνονται πάνω από 1 φορά (με fade out). Επιπλέον παραμετροποίηση γίνεται στον αριθμό των επαναλήψεων.
Overdrive	Αλλοίωση γειτονικών samples ανάλογη με την ενέργεια του τρέχοντος sample. Παραμετροποίηση στην ένταση του effect.
Live Sampling	Ο χρήστης επιλέγει ένα τμήμα ήχου (την ώρα που αναπαράγεται) και αυτό επαναλαμβάνεται, μέχρι να αποφασίσει ο χρήστης να επιστρέψει στην κανονική ροή.
Reverse	Αυτό το effect αντλήθηκε από τα παραδείγματα της βιβλιοθήκης minim και έγινε mapping ώστε να παραμετροποιείται το μέγεθος του samples από τα δεδομένα του Kinect. Αναπαράγει samples ανάστροφα (ΠΡΟΣΟΧΗ: Δεν αναπαράγει τα samples με ανάστροφη σειρά)

c. Custom initialization function used in setup()

```
void setup() {

  //Initialize Screen
  InitScreen();

  //SETUP KINECT
  InitKinect();

  //--Initializing Texts
  InitFonts();

  //Custom
  //initialize audio
  InitAudio();

  println(PFont.list());
}
```

d. SimpleOpenNI events (modified)

```
void onNewUser(int userId)
{
  println("NewUser? - userId: " + userId);
  enqueueMsg("NewUser? - userId: " + userId);

  if (autoCalibration) {
    enqueueMsg("AutoCalibrating User: " + userId);
    isInitialized=false;
    kinect.requestCalibrationSkeleton(userId, true);
  }
  else
    kinect.startPoseDetection("Psi", userId);
}

void onLostUser(int userId)
{
  println("Lost User: " + userId);
  enqueueMsg("Lost User: " + userId);
}

void onExitUser(int userId)
{
  println("Exit User: " + userId);
  enqueueMsg("Exit User: " + userId);
}

void onReEnterUser(int userId)
{
  println("ReEnterUser: " + userId);
  enqueueMsg("ReEnterUser: " + userId);
```

```
}

void onStartCalibration(int userId)
{
  println("StartCalibration - userId: " + userId);
}

void onEndCalibration(int userId, boolean successfull)
{
   println("EndCalibration - userId: " + userId + ", successfull: " +
successfull);

  if (successfull)
  {
    println("  User calibrated !!!");
    kinect.startTrackingSkeleton(userId);
  }
  else
  {
    println("  Failed to calibrate user !!!");
    println("  Start pose detection");
    kinect.startPoseDetection("Psi", userId);
  }
}

void onStartPose(String pose, int userId)
{
  println("onStartPose - userId: " + userId + ", pose: " + pose);
  println(" stop pose detection");

  kinect.stopPoseDetection(userId);
  kinect.requestCalibrationSkeleton(userId, true);
}

void onEndPose(String pose, int userId)
{
  println("onEndPose - userId: " + userId + ", pose: " + pose);
}
```

e. Display Functions

```
PFont titleFont;
PFont mouseFont;
PFont infoFont;

void InitScreen(){
  //set window background colour
  background(0, 0, 0);
  //set window size
  size(1024, 600);
  //set draw() rate
  frameRate(30);
```

```
  //smooth enabled (optional)
  smooth();

  //image mode
  imageMode(CENTER);

      println("Screen Initialized");
    enqueueMsg("Screen Initialized");

}


void InitFonts() {
  titleFont=createFont("Asimov", 24, true);
  mouseFont=createFont("Moire Light", 12, true);
  infoFont=createFont("Courier New", 12, true);
}

void DrawElements() {

  //TITLE text(mode)
  noStroke();
  textFont(titleFont);
  textAlign(CENTER);
  fill(255, 255, 255);
  if(kinectSynth==true){  text("Synth Mode 0.1", width/2, 30);}
  else{ text("Filter Mode 0.1", width/2, 30);}



  //MOUSE text(coordinates)
  noStroke();
  textFont(mouseFont);
  textAlign(LEFT);
  fill(255, 255, 255);
  text(mouseX + " : "+mouseY, mouseX, mouseY);


  //INFO AREA
  rectMode(RADIUS);
  strokeWeight(2);
  stroke(250, 250, 100);
  noFill();
  rect(3*(width/4), height-35, 250, 30);

  //MESAGE AREA
  rectMode(RADIUS);
  strokeWeight(2);
  stroke(250, 250, 100);
  noFill();
  rect(width/4, height-35, 250, 30);
}

void DrawText() {

  //INFO TEXT
  if (infoText[0]!=null) {
    noStroke();
    textFont(infoFont);
    textAlign(LEFT);
    fill(0, 0, 255);
```

```
    for (int i =0; i<5;i++) {
      text(infoText[i], width/2+10, (height-68)+((i+1)*12));
      if (i==4)break;
      if(infoText[i+1]==null)break;
    }
  }

    //MSG TEXT
  if (msgText[0]!=null) {
    noStroke();
    textFont(infoFont);
    textAlign(LEFT);
    fill(255, 0, 0);
    for (int i =0; i<5;i++) {
      text(msgText[i], 10, (height-68)+((i+1)*12));
      if (i==4)break;
      if(msgText[i+1]==null)break;
    }
  }


}
```

f. Print on Screen Functions

```
void enqueueInfo(String value) {

  if (infoText[5]!=null) {
    for (int i=0;i<5;i++) {
      infoText[i]=infoText[i+1];
    }
    infoText[5]=nfc(float(millis())/100,1)+":  "+value;
  }
  else {
    int i=0;
    while (infoText[i]!=null)i++;
    infoText[i]=nfc(float(millis())/100,1)+":  "+value;
  }
}

void enqueueMsg(String value) {
  if (msgText[5]!=null) {
    for (int i=0;i<5;i++) {
      msgText[i]=msgText[i+1];
    }
    msgText[5]=nfc(float(millis())/100,1)+":  "+value;
  }
  else {
    int i=0;
    while (msgText[i]!=null)i++;
    msgText[i]=nfc(float(millis())/100,1)+":  "+value;
  }
}
```

g. Keyboard Inputs Handling

```
void keyPressed() {
  if (key=='P'||key=='p') {
    if (!player.isPlaying())   player.play();
    else player.pause();
    //play-pause
  }
  else if (key=='S'||key=='s') {
    player.cue(0);
    player.pause();
    //stop
  }
  else if (key=='O'||key=='o') {
    if (oscillatorOn=false)CreateOscillator(1, 1, 44100);
    else RemoveOscillator();
  }
  else if (key=='E'||key=='e') {
    if (oscillatorEffectOn)RemoveOscillatorEffect();
    else AddOscillatorEffect();
  }
  else if (key=='L'||key=='l') {
    if (!isLowPass)
      AddLowPassFilter();
    else if (isLowPass)
      RemoveLowPassFilter();
  }
  else if (key=='Q'||key=='q') {
    if (isSaw) {
      DeleteSawWave();
    }
    else {
      CreateSawWave();
    }
  }
  else if (key=='I'||key=='i') {
    isInitialized=false;
  }
}
```

h. BandPass Filter Functions

```
//FILTER MODE
void CreateBandPassFilter() {
  DeleteWaveLeft();
  DeleteWaveRight();
  player.play();
  bpFilter = new BandPass(600, 30, player.sampleRate());
  player.addEffect(bpFilter);
  kinectSynth=false;
  enqueueInfo("Synth Mode Terminated");
  enqueueMsg("Filter Mode Initiated");
}

void UpdateBandPassFilter() {

  switched=false;

  //calibrating
  if (rightHand.y<minHeight)rightHand.y=minHeight-1;
  if (leftHand.y<minHeight)leftHand.y=minHeight-1;
```

```
  if (rightHand.y>maxHeight)rightHand.y=maxHeight+1;
  if (leftHand.y>maxHeight)leftHand.y=maxHeight+1;

float passBand;

float distance = dist(0,rightHand.y,0,leftHand.y);
if(rightHand.y>leftHand.y)    passBand  =  dist(0,rightHand.y,0,maxHeight)+
(distance/2);
else passBand = dist(0,leftHand.y,0,maxHeight)+(distance/2);

passBand=map(passBand,0,diff,10,10000);

float bandWidth=map(distance,0,diff,0,20000);

//       float   passBand   =   map((abs((dist(0,minHeight,0,rightHand.y)-
dist(0,minHeight,0,leftHand.y)))/2),  0,  dist(0,maxHeight,0,minHeight),  1f,
10000);
//      float   bandWidth   =   map((dist(0,rightHand.y,0,leftHand.y)),   0,
dist(0,maxHeight,0,minHeight), 0f, 24000);
  bpFilter.setFreq(passBand);
  bpFilter.setBandWidth(bandWidth);

  println("distance: "+distance+"band: "+passBand+"  width: "+bandWidth);
}

void DeleteBandPassFilter() {
  println(leftHand.y+"   "+rightHand.y);
  if(switched){
  player.removeEffect(bpFilter);
  player.cue(0);
  player.pause();
  kinectSynth=true;}
  else switched=true;
}
```

i. Sinewave Audio Functions

```
//SYNTH MODE
void CreateWaveLeft() {
  leftSine = new SineWave(map(leftHand.x, minWidth, maxWidth, 0, 20000),
map((maxHeight-leftHand.y), minHeight, maxHeight, 0, 1), 44100f);
  leftSine.portamento(200);
  output.addSignal(leftSine);
  enqueueInfo("Left-Hand Wave Created. Freq=  "+map(leftHand.x, minWidth,
maxWidth, 0, 20000));
  leftOn=true;
  kinectSynth=true;
}

void CreateWaveRight() {
  rightSine = new SineWave(map(rightHand.x, minWidth, maxWidth, 0, 20000),
map((maxHeight-rightHand.y), minHeight, maxHeight, 0, 1), 44100f);
  rightSine.portamento(200);
  output.addSignal(rightSine);
  enqueueInfo("Right-Hand Wave Created. Freq=  "+map(rightHand.x, minWidth,
maxWidth, 0, 20000));
  rightOn=true;
  kinectSynth=true;
}
```

```
void UpdateWaveLeft() {
  leftSine.setFreq(map(leftHand.x, minWidth, maxWidth, 0, 20000));
//---------  println(leftHand.x);
//---------println(  map(leftHand.x, minWidth, maxWidth, 0, 20000));
  leftSine.setAmp(map((maxHeight-leftHand.y), minHeight, maxHeight, 0, 1));
}

void UpdateWaveRight() {
  rightSine.setFreq(map(rightHand.x, minWidth, maxWidth, 0, 20000));
  rightSine.setAmp(map((maxHeight-rightHand.y), minHeight, maxHeight, 0,
1));
}

void DeleteWaveLeft() {
  output.removeSignal(leftSine);
  leftOn=false;
  enqueueInfo("Left-Hand Wave Removed");
}

void DeleteWaveRight() {
  output.removeSignal(rightSine);
  rightOn=false;
  enqueueInfo("Right-Hand Wave Removed");
}
```

j. Audio Objects Created

```
AudioPlayer player;
AudioInput input;
AudioOutput output;
AudioMetaData meta;

boolean leftOn=false, rightOn=false;
SineWave sine, leftSine, rightSine;
SineWave osci;

SawWave saw;

OscillatorEffect oeffect;

LowPassFS lpFilter;
BandPass bpFilter;
```

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή Εργασία

Ανάπτυξη Πλατφόρμας Μετατροπής Τρισδιάστατης Εικόνας σε Ήχο

(3D Image-to-Sound Mapping)

Ποτετσιανάκης Εμμανουήλ (Α.Μ. 1698)

Δρ. Τριανταφυλλίδης Γεώργιος
(επιβλέπων)

Βασικές Έννοιες

- Αντιστοίχιση (Mapping)
- Τρισδιάστατη Εικόνα (3D Image)
- Ψηφιακός Ήχος (Digital Audio)
- Αισθητήρας Βάθους (Depth Sensor)
- Skeleton Tracking
- Σύνθεση Ήχου

Εργαλεία

Κάμερα

Microsoft's Kinect

Γλώσσα Προγραμματισμού

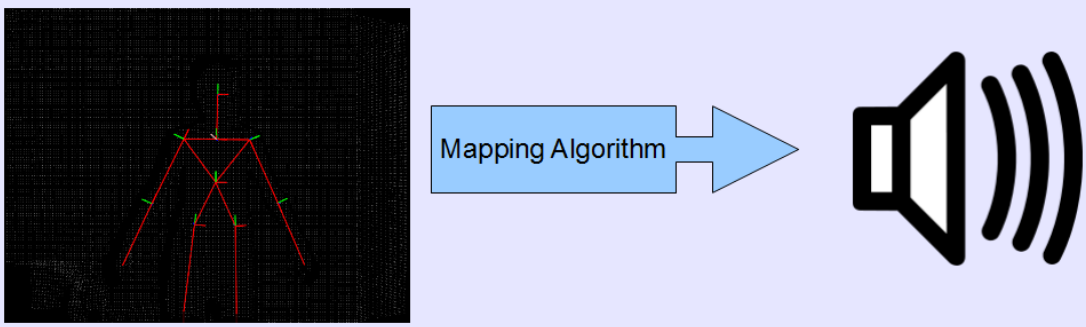
Processing

Βιβλιοθήκες:

Kinect:	**SimpleOpenNI**	http://http://code.google.com/p/simple-openni/
Audio:	**minim**	http://code.compartmental.net/tools/minim/
Graphics:	**P3D**	http://processing.org
Text:	**NextText**	http://www.nexttext.net
OSC:	**oscP5**	http://www.sojamo.de/libraries/oscP5/

Τρόπος Λειτουργίας

1. Αναγνώριση Χρήστη – Calibration
2. Συλλογή Δεδομένων Τρισδιάτατης Εικόνας – Gesture Recognition
3. Εφαρμογή Τεχνικής/Αλγορίθμων Αντιστοίχισης (Mapping)
4. Παρουσίαση Πληροφορίας (Παραγωγή Ήχου)



Τρόποι Λειτουργίας (Modes)

- Σύνθεσης Ήχου (Synthesis)

  Gestures → Synthesizer → Audio Out.

- Επεξεργασίας Ήχου (Effects)

  Audio In. + Gestures → Effect → Audio Out.

- Διεπαφής Χρήστη – Μηχανής Ήχου (Interface)

  Gestures → → Transmit → [redacted] → Audio Out.

Synthesis Mode

Είσοδος: Συντεταγμένες Χρήστη

Έξοδος: Ηχητική Κυματομορφή

Τεχνική Αντιστοίχισης:

Συντεταγμένες στον χώρο → Χαρακτηριστικά Κύματος

Θέση στον Οριζόντιο Άξονα:::Συχνότητα

Θέση στον Κάθετο Άξονα:::Πλάτος

Trigger1:::Σίγαση

Trigger2:::Αλλαγή Λειτουργίας

Λειτουργία Φίλτρου Ζώνης

Είσοδος: Ηχητικό Δείγμα, Συντεταγμένες Χρήστη

Έξοδος: Συγκεκριμένες Συχνότητες Ηχητικού Δείγματος

Τεχνική Αντιστοίχισης:

Συντεταγμένες στον χώρο → Χαρακτηριστικά Φίλτρου

Θέση στον Κάθετο Άξονα:::Κεντρική Συχνότητα

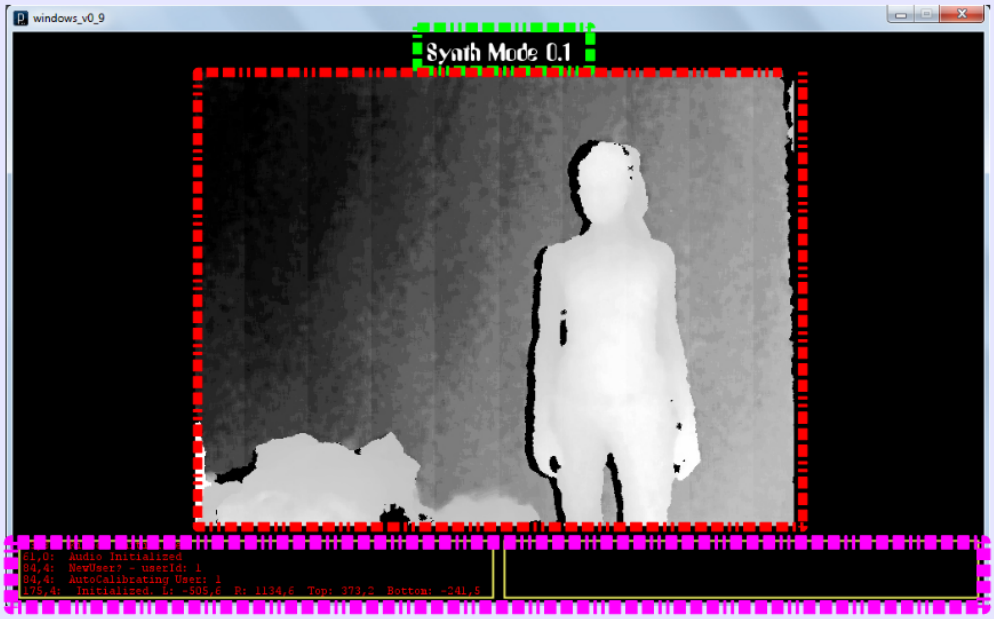
Απόσταση στον Κάθετο Άξονα:::Πλάτος Ζώνης

Trigger1:::Σίγαση

Trigger2:::Απενεργοποίηση

Trigger3:::Αλλαγή Λειτουργίας

Interface



Elements: i. Mode Indicator ii. Monitor iii. Message/Info Display

Showcase Video

- Μέρος 1: Εισαγωγή – Παρουσίαση Βασικών Συναρτήσεων
    - Μέρος 2α: Επίδειξη Πλατφόρμας – Synthesis Mode
- Μέρος 2β: Επίδειξη Πλατφόρμας – BandPass Filter Mode

Συμπεράσματα

Αποδείχτηκε ότι είναι υλοποιήσιμο concept...

...το οποίο απαιτεί εύκολα προσβάσιμη τεχνολογία (Kinect – Java)

...και έχει πολλαπλές πιθανές εφαρμογές (τέχνες, διασκέδαση, επιστήμες)

...χωρίς να χρειάζεται εμπειρία από τον χρήστη.

<u>Processing:</u>

- Πολυμορφία
- Ease of Development
- Ταχεία Εξέλιξη

Εναλλακτικές Προσέγγισης / Υλοποίησης

C#

Microsoft Kinect SDK

XNA

Για Processing:

SimpleOpenNI: OpenKinect

P3D: OpenGL

minim: Sonia | Beads

Προσθήκες/Επεκτάσεις Πλατφόρμας

- Audio Interface Mode
- Effects Mode
- Τεχνικές Βελτίωσης Αντιστοίχισης
- Υποστήριξη Πολλαπλών Χρηστών
- Εναλλακτική Αντιστοίχιση
- Εμπλουτισμός Περιβάλλοντος Εργασία

+

Προτάσεις για μελέτη

Καλλιτεχνικής Φύσης:

- Εικονικά Μουσικά Όργανα / Συσκευές

  + Visualization

- Δημιουργία ηχητικού τοπίου

Τηλεπικοινωνιακού Προσανατολισμού

- Πλατφόρμα σύνθεσης ομιλίας
- Διασύνδεση εφαρμογών ήχου διαφορετικής φύσης

Επεξεργασίας Ήχου

- Αναγνώριση ηχητικών προτύπων χρησιμοποιώντας οπτικά ερεθίσματα
- Αυτόματη διαμόρφωση/επεξεργασία ήχου