ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Χρήση της γλώσσας προγραμματισμού Chuck για την σύνθεση μουσικής με μουσικό όργανο το ηλεκτρικό βιολί

Use of the Chuck programming language to develop an electric violin music synthesis

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΚΟΥΤΖΙΝΗΣ ΝΙΚΟΛΑΟΣ

Επιβλέποντες καθηγητές: **Ακρίτας Αλκιβιάδης**
Καθηγητής Π.Θ.

Σταμούλης Γεώργιος
Καθηγητής Π.Θ.

Βόλος, Ιούλιος 2015

Περιεχόμενα

Περίληψη 3

1. Εισαγωγή 7
1.1 Περιγραφή της γλώσσας προγραμματισμού Chuck.....7
1.2 Υπολογιστική Μουσική και Ήχος.....8

2. Chuck.....10
2.1 Βασικά χαρακτηριστικά της γλώσσας προγραμματισμού chuck 10
2.2 Ο chuck τελεστής (=>) 10
2.3 Η Chuck στην διάσταση του χρόνου 11
2.4 Συγχρονισμός των Shreds.....13
2.5 Προδιαγραφές και χαρακτηριστικά της Chuck.....14
2.5.1 Τύποι τιμών και μεταβλητών.....14
2.5.2 Πίνακες..... 15
2.5.3 Τελεστές.....16
2.5.4 Δομές ελέγχου.....16
2.5.5 Διαχείριση του Χρόνου.....16
2.5.6 Συναρτήσεις 18
2.5.7 Συγχρονισμός ,διεργασίες και shreds.....18
2.5.8 Προγραμματίζοντας με Events.....19
2.5.9 Αντικείμενα.....19
2.5.10 Unit generators.....20
2.5.11 Unit analyzers.....20

3.Βιολί.....21
3.1 Ιστορική εξέλιξη.....21
3.2 Περιγραφή ηλεκτρικού βιολιού.....22
3.3 Τεχνικές του βιολιού και παραγωγή ήχου.....25

4. Σύνθεση Μουσικής.....26
4.1. Βασικά χαρακτηριστικά σύνθεσης.....26
4.2 Σύνθεση μουσικής με την γλώσσα προγραμματισμού Chuck.....27
4.2.1 Βασικά μουσικά, τονικά και ρυθμικά χαρακτηριστικά

της σύνθεσης σε chuck.....28
4.2.2 Εκτέλεση της μουσικής σύνθεσης με την χρήση νημάτων "Shreds".....31
4.2.3 Συγχρονισμός Shreds για την δημιουργία μουσικής ορχήστρας.....45
4.2.4 Αλληλεπίδραση του ηλεκτρικού βιολιού με τον
υπολογιστή για την ολοκλήρωση της μουσικής σύνθεσης.....47
4.2.5 Ηχογράφηση της μουσικής σύνθεσης.....49

Περίληψη

Η μουσική αποτελεί βασικό συστατικό του πολιτισμού και της εξέλιξης του ανθρώπου. Η τεχνολογία και οι ηλεκτρονικοί υπολογιστές έχουν εισχωρήσει σε όλους τους τομείς της καθημερινότητας. Συνθέτες και μουσικοί έχουν αρχίσει εδώ και χρόνια να χρησιμοποιούν του υπολογιστές ως εργαλεία για την σύνθεση μουσικής. Τα μουσικά όργανα όπως το ηλεκτρικό βιολί μπορούν να αλληλεπιδράσουν με τους υπολογιστές και να δημιουργήσουν μια μουσική σύνθεση. Επίσης οι γλώσσες προγραμματισμού έχουν εξελιχθεί και δίνουν τεράστιες δυνατότητες σους χρήστες.

Στην παρούσα διπλωματική εργασία γίνεται εκτενής περιγραφή της γλώσσας προγραμματισμού chuck που χρησιμοποιείται για την σύνθεση μουσικής. Κατόπιν προγραμματίζοντας στην γλώσσα chuck παράγεται μια μουσική σύνθεση που εκτελείται από τέσσερα μουσικά όργανα, δημιουργώντας μια ορχήστρα. Τέλος, παρουσιάζεται η διαδικασία αλληλεπίδρασης της chuck με τον χρήστη μέσω του ηλεκτρικού βιολιού.

Στην πεμπτουσία της μουσικής

και

στον καθηγητή μου κ.Ακρίτα.

Κεφάλαιο 1

Εισαγωγή

1.1 Περιγραφή της γλώσσας προγραμματισμού Chuck.

Η γλώσσα προγραμματισμού Chuck χρησιμοποιείται για την αναπαραγωγή ήχου σε πραγματικό χρόνο και την σύνθεση μουσικής. Διατίθεται δωρεάν στο MacOS X, Windows και Linux. Η chuck παρέχει την δυνατότητα τροποποίησης των προγραμμάτων της on-the-fly ,δηλαδή χωρίς να απαιτείται προηγουμένως ο τερματισμός της εκτέλεσης του προγράμματος και βασίζεται στο ταυτόχρονο μοντέλο προγραμματισμού. Επιπλέον, η chuck υποστηρίζει MIDI και αλληλεπιδρά με ηλεκτρικά μουσικά όργανα όπως το ηλεκτρικό βιολί . Είναι εύκολη στην χρήση της και αποτελεί ένα ισχυρό εργαλείο προγραμματισμού για συνθέτες, ερευνητές και εκτελεστές μουσικών οργάνων ,τόσο για την σύνθεση ,όσο και για τον πειραματισμό τους στην παραγωγή μουσικής σε πραγματικό χρόνο.

Σήμερα τα μηχανήματα και οι ηλεκτρονικοί υπολογιστές έχουν φτάσει σε τέτοιο σημείο εξέλιξης ,όπου ο σχεδιασμός τους δεν στοχεύει μόνο στις εντυπωσιακές επιδόσεις αλλά επικεντρώνεται κυρίως στην ευελιξία και μεγιστοποίηση της αλληλεπίδρασης του χρήστη με το σύστημα. Έτσι η γλώσσα προγραμματισμού chuck είναι έντονα επίκαιρη και σύγχρονη ωθώντας τον χρήστη να παράγει μουσική και να προγραμματίσει με τον δικό του τρόπο. Συνδυάζει την λακωνικότητα των γλωσσών υψηλού επιπέδου που χρησιμοποιούνται για την παραγωγή μουσικής με την ευκολία και την απλότητα των χαμηλού επιπέδου γλωσσών προγραμματισμού.

Η chuck εξακολουθεί να είναι ένα ερευνητικό πείραμα «ανοικτού» κώδικα με σκοπό την δημιουργία μιας γλώσσας προγραμματισμού αποκλειστικά για την παραγωγή μουσικής. Ενώ αντλεί ιδέες από πολλές γλώσσες (C, Java, Max, SuperCollider), δεν βασίζεται όμως σε καμία υπάρχουσα .Ο σχεδιασμός της γλώσσας έγινε με τέτοιον τρόπο ώστε να εξασφαλίζει στον προγραμματιστή τον απόλυτο έλεγχο του προγράμματος και να προσαρμόζεται στον σχεδιασμό και την απόδοση του συστήματος. Ο στόχοι σχεδιασμού της γλώσσας είναι οι εξής: [1]

1. **Ευελιξία:** επιτρέπει στον προγραμματιστή να καθορίσει με απλό τρόπο τόσο τις υψηλού όσο και τις χαμηλού επίπεδου διεργασίες στο πέρασμα του χρόνο.
2. **Συγχρονισμός:** επιτρέπει στον προγραμματιστή να γράψει παράλληλα modules που μοιράζονται δεδομένα και χρόνο, και μπορούν να είναι ακριβώς συγχρονισμένα.
3. **Αναγνωσιμότητα:** παρέχει και διατηρεί μια ισχυρή επικοινωνία μεταξύ της δομής του κώδικα και του χρόνου.
4. Είναι μια γλώσσα **do-it-yourself**: συνδυάζει την εκφραστικότητα γλωσσών χαμηλότερου επιπέδου και την ευκολία των υψηλού επιπέδου γλωσσών προγραμματισμού.

5. Είναι **On-the-fly**: Ένα on-the-fly προγραμματιζόμενο σύστημα παρέχει τη δυνατότητα στον χρήστη να γράψει ή να τροποποιήσει, να μεταγλωττίσει ,να εκτελέσει τον νέο ή υπάρχον κώδικα και στη συνέχεια να τον ενσωματώσει σε αυτόν ενώ βρίσκεται σε λειτουργία. Πάντα με συνέπεια στον χρόνο και συγχρονισμό. Ο στόχος του on-the-fly προγραμματισμού είναι να επιτρέπει στους προγραμματιστές ,ερμηνευτές και συνθέτες να τροποποιήσουν απ' ευθείας τα ενεργά προγράμματά τους .Χωρίς πρώτα να χρειάζεται να σταματήσουν για να τροποποιήσουν τον κώδικα και μετά να κάνουν επανεκκίνηση [2].

Βασικό χαρακτηριστικό της chuck είναι ο τρόπος διαχείρισης της στο πέρασμα του χρόνου. Δηλαδή παρέχει στους προγραμματιστές άμεσο, ακριβή και εύκολο έλεγχο του προγράμματος κατά την διάρκεια εκτέλεσης του και σε οποιαδήποτε άλλη παράμετρο σχετίζεται με αυτό. Αυτό κάνει την chuck ένα διασκεδαστικό και ιδιαίτερα ευέλικτο εργαλείο για την περιγραφή, το σχεδιασμό, την εφαρμογή και την σύνθεση ήχου τόσο σε χαμηλά όσο και υψηλά επίπεδα [3].

1.2 Υπολογιστική Μουσική και Ήχος.

Η Υπολογιστική Μουσική και Ήχος - Sound and Music Computing (SMC) είναι ένα ερευνητικό πεδίο που μελετάει διεπιστημονικά ολόκληρη τη μουσική και ηχητική επικοινωνία. Συνδυάζοντας επιστημονικές, τεχνολογικές και καλλιτεχνικές μεθόδους σκοπεύει στην κατανόηση, μοντελοποίηση και παραγωγή της μουσικής και του ήχου μέσω υπολογιστικών προσεγγίσεων.

Η πρώτη προσπάθεια χρήσης των υπολογιστών στη μουσική και στην παράγωγή ήχου με σκοπό την εγκαθίδρυση ενός επιστημονικό πεδίο βρίσκονται γύρω στο 1950, όταν μερικοί συνθέτες, μαζί με επιστήμονες και μηχανικούς, σε διάφορα μέρη του κόσμου, ξεκίνησαν την εξερεύνηση των μουσικών εφαρμογών με ψηφιακές τεχνολογίες. Από τότε, ως επιστημονικό πεδίο έχει μια πλούσια ιστορία. Η επιστημονική κοινότητα καθιέρωσε το 1974 τη Διεθνή Ένωση Μουσικής με Υπολογιστές (International Computer Music Association) και το Διεθνές Συνέδριο Μουσικής με Υπολογιστές (International Computer Music Conference). Το 1977 καθιερώθηκε το Επιστημονικό Περιοδικό Μουσικής με Υπολογιστές (Computer Music Journal). Το Κέντρο Υπολογιστικής Έρευνας στη Μουσική και Ακουστική στο Πανεπιστήμιο Stanford (The Center for Computer Research in Music and Acoustics (CCRMA)) ,δημιουργήθηκε στις αρχές της δεκαετίας του 70 και στα τέλη το Ινστιτούτο Έρευνας και Συντονισμού Ακουστικής/Μουσικής (Institute for Research and Coordination Acoustic/Music (IRCAM)) στο Παρίσι. Το 2004 ξεκίνησε το Συνέδριο Υπολογιστών στη Μουσική και τον Ήχο, όπως και μία πρώτη πρωτοβουλία από την ευρωπαϊκή ένωση που οδήγησε στο SMC Roadmap [4].

Σήμερα ο υπολογιστής αποτελεί αναπόσπαστο κομμάτι της καθημερινής ζωής. Η χρήση του σε κάθε κλάδο της επιστήμης είναι δεδομένη. Στην μουσική και γενικότερα στην τέχνη ο υπολογιστής έχει ανοίξει νέους ορίζοντες για τους δημιουργούς . Οι νέοι συνθέτες προτιμούν να γράφουν τα κομμάτια τους κατευθείαν με ειδικά προγράμματα στον υπολογιστή παρά στην παρτιτούρα. Με τον τρόπο αυτό τους δίνεται η δυνατότητα να ακούσουν απευθείας την νέα τους σύνθεσή και να την διορθώσουν αμέσως. Σε αντίθεση με τους μεγάλους κλασσικούς συνθέτες του παρελθόντος ,οι οποίοι ορισμένα από τα έργα τους δεν κατάφεραν να τα ακούσουν ποτέ από μια συμφωνική ορχήστρα .Γι' αυτό και η σύνθεση μουσικής αποτελούσε προνόμιο των πλουσίων κάθε εποχής. Στην εποχή μας με την εξέλιξη του υπολογιστή δίνεται η δυνατότητα στον καθέναν να συνθέσει σε υψηλό επίπεδο. Επίσης

χάρη στα ελευθέρα λογισμικά όπως είναι η chuck αλλά και στα ηλεκτρονικά όργανα που συνδέονται με τον υπολογιστή ,η παραγωγή μουσικής γίνεται ακόμα πιο εύκολη ,απλή και προσιτή σε όλους.

Κεφάλαιο 2

Chuck

2.1 Βασικά χαρακτηριστικά της γλώσσας προγραμματισμού chuck .

Ο κύριος στόχος [5] των δημιουργών της chuck ήταν να σχεδιαστεί μια «φυσική» γλώσσα προγραμματισμού η οποία:

1. Θα αναπαριστά με ακρίβεια και συγχρονισμό μια πολύπλοκη σύνθεση ήχου.
2. Θα παρέχει ευελιξία στον έλεγχο κατά την πάροδο του χρόνου.
3. Θα δίνει την δυνατότητα να λειτουργεί σε πολλαπλά, δυναμικά και ταυτόχρονα επίπεδα ελέγχου .
4. Θα υποστηρίζει On-the-fly προγραμματισμό.

Έτσι οι βασικοί άξονες σχεδιασμού της chuck είναι να εμπεριέχει:

- Ένα συνδετικό και σημαντικής βαρύτητας τελεστή.
- Ένα ακριβές μοντέλο χρονισμού που ενοποιεί τον συγχρονισμό χαμηλού και υψηλού επιπέδου και ευνοεί την ανάπτυξη κώδικα.
- Ένα ακριβές μοντέλο ταυτόχρονου προγραμματισμού που να υποστηρίζει πολλαπλά, ταυτόχρονα και δυναμικά επίπεδα ελέγχου.
- Ένα περιβάλλον run-time που επιτρέπει τον on-the-fly προγραμματισμό.

2.2 Ο chuck τελεστής (=>).

Ο βασικός πυλώνας στον οποίο στηρίζεται το συντακτικό της chuck είναι ο «chuck τελεστής» που συμβολίζεται ως "=>" και χρησιμοποιείται για να τοποθετήσει μία οντότητα μέσα σε μία άλλη. Η γλώσσα χρησιμοποιεί αυτή την ιδιότητα για να εκφράσει διαδοχικές λειτουργίες και να κατευθύνει την ροή των δεδομένων. Αυτός ο τελεστής αποτελεί ουσιαστικά την συντακτική «κόλλα» με την οποία ενώνονται τα στοιχεία του κώδικα της chuck μεταξύ τους.

Στο παρακάτω κομμάτι κώδικα διαπιστώνουμε την χρήση του τελεστή "=>".Παραλείπεται η δήλωση της μεταβλητή foo και υποτίθεται ότι η foo δηλώθηκε ως γεννήτρια μονάδα, δηλαδή ως ένα στοιχείο επεξεργασίας ηχητικού σήματος. Η συμπεριφορά του "=>" σε αυτήν την περίπτωση θα είναι να συνδεθεί η έξοδος του foo στην είσοδο της dac (μια άλλης γεννήτριας μονάδας).

```
//συνδέω 'foo' με 'dac'
foo=>dac;
```

Ένα πιο πολύπλοκο παράδειγμα μπορεί να φανεί στον παρακάτω κώδικα. Εδώ κατασκευάζεται ένα απλό όργανο σύνθεσης χρησιμοποιώντας μια σειρά από γεννήτριες μονάδες(Unit Generators), τις οποίες εδώ παραλείπεται η δήλωση τους. Αυτές είναι: μία γεννήτρια λευκού θορύβου, ένα φίλτρο και μία έξοδος ήχου. Αξίζει να σημειωθεί ότι οι

εντολές εκτελούνται από αριστερά προς τα δεξιά και με την ίδια σειρά πρέπει και οι προγραμματιστές που επιλέγουν την chuck να διαβάζουν και να πληκτρολογούν τις εντολές.

```
//συνδέω 'noise' στο 'filter' στο 'dac'

noise=>filter=>dac;
```

Ένας τελεστής Chuck "=>" μπορεί να αποτελείται από οποιονδήποτε τύπο αντικειμένων όπως γεννήτριες μονάδες, τιμές και μεταβλητές. Η σημασιολογική του δήλωση εξαρτάται από τους τύπους των αντικειμένων και την καταχώρηση (φόρτωση) τους σε αυτούς τους τύπους.

Εκτός από την σύνδεση γεννητριών μονάδων ,ο τελεστής chuck μπορεί να χρησιμοποιηθεί και για άλλες χρήσεις όπως για την κλήση συναρτήσεων, ανάθεση τιμών και η χρονική εξέλιξη. Στο παράδειγμα που ακολουθεί παρουσιάζονται δύο διαφορετικές συντάξεις εντολών για την κλήση εμφωλευμένων συναρτήσεων. Και στις δύο περιπτώσεις ο τελεστής "=>" μπορεί να οδηγήσει σε πιο γραμμική και συγχρονισμένη αναπαράσταση των εμφωλεύσεων. Ο προγραμματιστής μπορεί να θεωρήσει ότι οι μεταβλητές περνάνε μέσα στις συναρτήσεις από μια σειρά μετασχηματισμών από αριστερά προς τα δεξιά.

```
//χωρίς =>
Math.fabs(Math.min( a,b ) );

//με χρήση =>
(a,b)=>Math.min=>Math.fabs;
```

2.3 Η Chuck στην διάσταση του χρόνου .

Το κλειδί στην λειτουργικότητα της chuck είναι να κάνει ένα πρόγραμμα να «αυτό-προγραμματίζεται» ως προς τον χρόνο και να έχει επίγνωση της κατάστασης του σε κάθε χρονική στιγμή και του ελέγχου της προόδου του στην πάροδο του χρόνου .Επιπλέον πολλά κομμάτια προγραμμάτων μπορούν να αντιληφθούν τον χρόνο με μία κοινή λογική, καθιστώντας δυνατό τον συγχρονισμό με παράλληλο κώδικα αποκλειστικά βασιζόμενα στην διάσταση του χρόνου. Αυτό είναι που χαρακτηρίζει την chuck ως μία "strongly-time" γλώσσα προγραμματισμού. Με στόχο την σύνθεση και την ανάλυση, ο έλεγχος των γεννητριών μονάδων πρέπει να μπορεί να ανακτηθεί ανά πάσα χρονική στιγμή και σε οποιαδήποτε κατάσταση .Για να συμβεί αυτό πρέπει :

- Η chuck να εμπεριέχει τους τύπους μεταβλητών **time** και **dur** στην σημασιολογία της γλώσσας της για τον προσδιορισμό του χρόνου και της διάρκειας του.
- Η γλώσσα θα πρέπει να επιτρέπει ρητώς καθορισμένες αριθμητικές πράξεις τόσο στον χρόνο όσο και στην διάρκεια του.
- Το μοντέλο της γλώσσας να παρέχει πλήρη αντιστοίχιση του κώδικα ως προς τον χρόνο για την δυνατότητα σύνθεσης μουσικής με ντετερμινιστικό τρόπο. Είναι

φυσικό λοιπόν να καθορίζεται ο χρόνος από οποιοδήποτε σημείο του προγράμματος.

- Η γλώσσα να προσφέρει μια ειδική λέξη κλειδί (χρονικού τύπου) που θα κρατάει τον χρόνο .Δηλαδή έναν τρόπο για να μπορεί ο προγραμματιστής να αναφερθεί στην χρονική στίμη που θέλει να εκτελεστεί μια εντολή με μια άμεση ,ντετερμινιστική και καλώς ορισμένη έννοια.
- Η chuck προσφέρει μία σταθερά με στόχο να ενημερώνει για τον χρόνο οπουδήποτε στο πρόγραμμα. Για την διάρκεια (D+=>now;) ή για τον απόλυτο χρόνο (T=>now;).

Στο παρακάτω παράδειγμα εισάγεται ένα ημιτονοειδές κύμα και αλλάζει η συχνότητα της ταλάντωσης με τυχαίο τρόπο κάθε 100 millisecond.

```
1 SinOsc foo=> dac;
2
3 while(true)
4 {
5     Std.rand2f(30, 1000) =>foo.freq;
6
7     100::ms => now;
8
9 }
```

Εικόνα 1

Στην **εικόνα 1** η εντολή SinOsc ονομάζεται foo και εκχωρείται στην μεταβλητή dac (χρησιμοποιείται για την έξοδο ήχου). Στην γραμμή 5 ορίζεται η συχνότητα μεταξύ 30 και 1000 Hz για το ημίτονο. Το πρόγραμμα προχωράει ανά 100 msec στην γραμμή 7.Είναι αρκετά σημαντικό να γίνει κατανοητή η εντολή σ' αυτήν τη γραμμή. Η ροή του προγράμματος σταματάει και επιστρέφει τον έλεγχο στην εικονική μηχανή της chuck και την μηχανή σύνθεσης. Αυτή παράγει ήχο ανά 100 msec, πριν τον επιστρέψει στο πρόγραμμα μας. Έτσι η έννοια του χρόνου στην chuck είναι παρόμοια με μία κλήση sleep άλλων γλωσσών. Η διαφορά είναι ότι εδώ η γλώσσα εγγυάται συγχρόνως ακρίβεια σε λογικό χρόνο, επιτρέποντας έτσι να εκτελεστεί μια πολύπλοκη διαδικασία χρονισμού σε όλο το σύστημα. Αυτή η μέθοδος ανάγνωσης του κώδικα μπορεί να εφαρμοστεί και σε πιο σύνθετα προγράμματα για την προσέγγιση του χρόνου με απλό και ξεκάθαρο τρόπο.

Η προσέγγιση αυτή του χρόνου ενσωματώνει στον φυσικό προγραμματισμό τον απευθείας έλεγχο χρονισμού. Δίνεται λοιπόν στον προγραμματιστή η δυνατότητα εκτέλεσης υπολογισμών σε αυθαίρετα χρονικά σημεία και η δυνατότητα διαχείρισης του χρόνου με ακριβή τρόπο. Μία άλλη περίπτωση ιδιαιτερότητας της εξέλιξη του χρόνου στην chuck είναι η αναμονή για ένα ή περισσότερα συμβάντα. Τέτοια γεγονότα μπορεί να είναι η επικοινωνία με ασύγχρονα μηνύματα μέσω MIDI και συσκευές εισόδου όπως το ηλεκτρικό βιολί στην συγκεκριμένη εργασία. Ο κώδικας θα συνεχίσει να εκτελείται όταν επιτευχθεί ο συγχρονισμός .Όσο το γεγονός περιμένει , η εικονική μηχανή είναι ελεύθερη να δρομολογήσει την σύνθεση του ήχου και άλλους υπολογισμούς συγχρονισμού.

Η έννοια του χρόνου παρόλα αυτά από μόνη της δεν είναι αρκετή, αφού ο ήχος και η

μουσική είναι μια ταυτόχρονη διεξαγωγή πολλών παράλληλων διεργασιών. Η chuck είναι μία ταυτόχρονη γλώσσα προγραμματισμού και επιτρέπει πολλαπλές και ανεξάρτητες διαδρομές για τον υπολογισμό και την παράλληλη εκτέλεση κώδικα. Η ελαστικότητα και η δύναμη του συγχρονισμού είναι σε μεγάλο βαθμό επέκταση της βασικής ιδέας της chuck, η οποία επιτρέπει την πολλαπλή και με χρονική ακρίβεια εκτέλεσης των διάφορων μονοπατιών υπολογισμών.

2.4 Συγχρονισμός των Shreds.

Η γλώσσα προγραμματισμού Chuck επιτρέπει από την φύση της στους προγραμματιστές να γράφουν κώδικα που ανταποκρίνεται τόσο σε σειριακό όσο και σε ταυτόχρονο μοντέλο συγχρονισμού των διεργασιών. Αυτός ο μηχανισμός παρέχει πολλαπλά και ταυτόχρονα εργαλεία ελέγχου. Για να το πετύχει αυτό όμως οι δημιουργοί της chuck εισήγαγαν στην γλώσσα την έννοια των shreds. Ένα shred μπορεί να προσδιοριστεί σαν ένα thread, δηλαδή είναι μια ανεξάρτητη, ελαφριά διαδικασία ,η οποία λειτουργεί με ταυτόχρονο τρόπο και μπορεί να μοιράζεται δεδομένα με άλλα shreds. Η βασική διαφορά όμως των shreds με τα παραδοσιακά νήματα είναι ο τρόπος που συγχρονίζονται . Τα threads εκτελούνται με μη ντετερμινιστικό τρόπο και η δρομολόγηση τους επιλέγεται από τον προγραμματιστή. Τα shreds είναι ντετερμινιστικού τύπου κομμάτια κώδικα και η δρομολόγηση τους γίνεται με βάση το χρονοδιάγραμμα του ήχου και φυσικά ο μεταξύ τους συγχρονισμός γίνεται υπό τον ίδιο μηχανισμό χρονισμού.

Τα shreds και τα threads σε γενικές γραμμές προγραμματίζονται με τον ίδιο τρόπο. Υπάρχουν όμως μερικές διαφορές :

1. Ένα shred δεν μπορεί να εξαρτάται και να ενεργοποιείται από ένα άλλο. Αυτό όχι μόνο επιτρέπει σε ένα μόνο shred να είναι τοπικά ντετερμινιστικό ,αλλά ένα ολόκληρο σύνολο από shreds να είναι σε όλο το εύρος του προγράμματος ντετερμινιστικό τόσο στον χρόνο όσο και στην σειρά εκτέλεσης.
2. Ένα shred σταματάει οικειοθελώς να απασχολεί τον επεξεργαστή προκειμένου να δώσει την δυνατότητα στα υπόλοιπα shreds να εκτελεστούν. Αυτή η ενέργεια όμως δεν πραγματοποιείται με την εντολή yield() ,που χρησιμοποιείται στα περισσότερα ταυτόχρονα συστήματα. Τα shred έχουν σχεδιαστεί με τέτοιο τρόπο ώστε να χρησιμοποιούν τον μηχανισμό χρονισμού της chuck .Όταν ένα shred εκτελείται ή περιμένει να τελειώσει κάποιο γεγονός ,έχει στην πραγματικότητα δρομολογηθεί από την δρομολογητή και παραιτείται από τον επεξεργαστή όταν πρέπει. Αυτό είναι σημαντικό όσο αφορά τον συγχρονισμό των shreds στο πέρασμα του χρόνου ,χωρίς να χρησιμοποιείται κάποια συνηθισμένη και εξαρτώμενη από τον χρήστη τεχνική συγχρονισμού.
3. Τα shreds εκτελούνται πλήρως σε επίπεδο χρήστη. Η εικονική μηχανή τρέχει και αυτή στο επίπεδο του χρήστη. Ο παραλληλισμός σε αυτό το επίπεδο έχει σημαντικά πλεονεκτήματα επιδόσεων πάνω στα νήματα του πυρήνα , επιτρέποντας ακόμα και σε εξαιρετικά μικρές διεργασίες να επιτύχουν καλή απόδοση .Αν και το κόστος της δημιουργίας και της διαχείρισης του παραλληλισμού είναι χαμηλός. Επίσης το κόστος εναλλαγής των shreds στον πυρήνα είναι πολύ χαμηλό ,αφού η

αλληλεπίδραση με αυτόν δεν είναι απαραίτητη. Επίσης ένας χρονοδρομολογητής που λειτουργεί σε επίπεδο χρήστη είναι πιο εύκολα τροποποιήσιμος.

Ένα βασικό πλεονέκτημα των shreds είναι ότι ο προγραμματιστής έχει τον πλήρη έλεγχο πάνω στην χρονική αλληλεπίδραση την shreds .Από την άλλη μεριά ένα πιθανό μειονέκτημα είναι ότι ένα shred μπορεί να κρεμάσει την εικονική μηχανή του chuck αν για κάποιο λόγο αποτύχει να ελευθερώσει τον επεξεργαστή .

Τα πολύ-shredded προγράμματα δεν είναι υπολογιστικά ισχυρότερα από τα μόνο-shredded .Τα δεύτερα μπορούν να διαχειριστούν τον παραλληλισμό και τον χρόνο ευκολότερα. Όπως ακριβώς τα threads κάνουν τον ταυτόχρονο προγραμματισμό διαχειρίσιμο και ενδεχομένως αυξάνουν την απόδοση τους, έτσι και τα shreds κατασκευαστικά είναι πιο δυναμικά. Η ευελιξία των shreds βοηθάει τον προγραμματιστή να δημιουργήσει πιο ντετερμινιστικά ,χρονικά ακριβέστερα και πολυνηματικά προγράμματα ήχου αντισταθμίζοντας έτσι τα πιθανά μειονεκτήματα.

2.5. Προδιαγραφές και χαρακτηριστικά της Chuck.

Η chuck είναι μια “strongly-typed” και “strongly-timed” γλώσσα προγραμματισμού με σκοπό την παράλληλη αναπαραγωγή ήχου .Η μεταγλώττιση της γίνεται μέσω εικονικών εντολών οι οποίες δίνονται απευθείας στην εικονική της μηχανή. Σε αυτήν την ενότητα θα παρουσιαστούν τα χαρακτηριστικά και οι δυνατότητες που παρέχει η γλώσσα για την σύνθεση της μουσικής.

2.5.1 Τύποι τιμών και μεταβλητών.

Primitive types

Οι θεμελιακοί τύποι είναι ουσιαστικά απλοί τύποι δεδομένων με τους οποίους προσδιορίζουμε την ιδιότητα μιας τιμής. Αυτοί είναι :

int	ακέραιος
float	δεκαδικός
time	χρόνος
dur	διάρκεια
void	κανένας τύπος
complex	μιγαδικός
polar	πολική μορφή μιγαδικού

Πίνακας 1

Μεταβλητές

Οι μεταβλητές χρησιμεύουν ως «αποθήκες» τιμών στην μνήμη. Πολλές φορές χρειάζεται να κρατήσουμε και να επαναφέρουμε αριθμητικές και χρονικές τιμές ή άλλες πληροφορίες. Για να χρησιμοποιηθεί λοιπόν μια μεταβλητή χρειάζεται πρώτα να την δημιουργήσουμε , να της δώσουμε ένα όνομα και τον τύπο της τιμής που θα αποθηκεύσει. Οι τρείς πιο κοινοί τύποι μεταβλητών είναι ο ακέραιος(int) ,ο δεκαδικός (float) και η χρονική διάρκεια(dur).

Reference types

Οι τύποι αναφοράς είναι τύποι που κληρονομούνται από μια κλάση αντικείμενου. Αυτοί είναι :

Object	Ο τύπος της κλάσης από την οποία θα κληρονομήσουν οι άλλες κλασεις.
array	Διατεταγμένο σύνολο δεδομένων ίδιου τύπου.
Event	Θεμελιώδης με δυνατότητα επέκτασης και συγχρονισμού μηχανισμός.
UGen	Επεκτάσιμη μονάδα της γεννήτριας βασικής κλάσης.
string	Συμβολοσειρά.

Πίνακας 2

Complex types

Δύο τύποι μιγαδικών χρησιμοποιούνται οι complex και polar. Συντακτικά πριν από κάθε μιγαδικό πρέπει να υπάρχει το σύμβολο #(…) .Με τις αναφορές .re και .im μετά την μεταβλητή που περιέχει έναν μιγαδικό μπορεί να επιστραφεί το πραγματικό και το φανταστικό του μέρος αντίστοιχα.

2.5.2. Πίνακες

Οι πίνακες είναι δομές N-διαστάσεων που μπορούν να αποθηκεύσουν διατεταγμένα σύνολα δεδομένων ίδιου τύπου. Ένα από τα πλεονεκτήματα των πινάκων είναι ότι μπορεί να αναδιαταχθούν εύκολα. Μπορούν επίσης να χρησιμοποιηθούν ως ευρετήριο και δομή που θα αποθηκεύει δείκτες σε συμβολοσειρές. Επίσης οι πίνακες μπορούν να θεωρηθούν ως αντικείμενα και να συμπεριφερθούν με τον ίδιο τρόπο. Πρέπει πάντα να γίνεται αρχικοποίηση ενός πινάκα .Αξίζει να σημειωθεί ότι ένας πίνακας από αντικείμενα αρχικοποιείται αυτόματα την στιγμή που δηλώνεται. Όπως και σε άλλες γλώσσες προγραμματισμού έτσι και στην chuck υπάρχει η δυνατότητα χρήσης πολυδιάστατων πινάκων.

2.5.3 Τελεστές.

Εκτός από τον τελεστή chuck "=>" που αναλύθηκε σε προηγούμενη ενότητα ,η chuck διαθέτει και άλλου τύπου τελεστές όπως είναι οι αριθμητικοί, οι δυαδικοί και οι λογικοί. Ο τελεστής ρητής ανάθεσης "@=>" χρησιμοποιείται κυρίως για αναθέσεις σε αναφορές αντικειμένων σε αντίθεση με τον τελεστή "=>" ο οποίος αφορά μόνο primitive τύπους δεδομένων. Οι αριθμητικοί τελεστές είναι +=>,-=>,*=>,/=> κλπ.

2.5.4 Δομές ελέγχου.

Όπως σε πολλές γλώσσες προγραμματισμού έτσι και στην chuck υπάρχουν οι συνηθισμένες δομές ελέγχου. Σε αυτές συμπεριλαμβάνονται τα εξής : if, else ,for, while και ορισμένες συμπληρωματικές όπως το until και το repeat(n), το οποίο δημιουργεί μια λούπα που επαναλαμβάνεται n φορές.

2.5.5. Διαχείριση του Χρόνου.

Η έννοια του χρόνου και του συγχρονισμού παίζει σημαντικό ρόλο στην κατανόηση της φιλοσοφίας της chuck. Τόσο ο χρόνος όσο και η διάρκεια είναι βασικά συστατικά της γλώσσας. Η λέξη **"now"** αποτελεί λέξη κλειδί στο λεξιλόγιο της γλώσσας και κρατάει την τρέχουσα χρονική στιγμή.

Στην chuck η λέξη **"time"** αναπαριστά μια απόλυτη στιγμή στον χρόνο(από την έναρξη του προγράμματος) και η λέξη **"dur"** αναπαριστά χρονική διάρκεια .

```
//διάρκεια ενός δευτερολέπτου
1::second => dur foo;

// ένα χρονικό σημείο μετά το now με διάρκεια της foo
 now+ foo =>time later;
```

Μια χρονική διάρκεια μπορεί να χρησιμοποιηθεί στην κατασκευή μιας νέας ,η οποία μπορεί να αποτελέσει βάση για την επαγωγική κατασκευή νέων.

```
// .5 είναι ένα τέταρτο
.5::second => dur quarter;

// 4 τέταρτα κάνουν ένα ολόκληρο
4::quarter => dur whole;
```

Η chuck στα πλαίσια του χρόνου υποστηρίζει τις εξής τιμές για την διάρκεια:

samp	Διάρκεια ενός δείγματος
ms	1 χιλιοστό του δευτερολέπτου
second	1 δευτερόλεπτο
minute	1 λεπτό
hour	1 ώρα
day	1 μέρα
week	1 εβδομάδα

Πίνακας 3

Μερικά παραδείγματα χρήσης των διαφόρων τύπων διάρκειας:

```
// διάρκεια μισού δείγματος
.5::samp => dur foo;

// 20 weeks
20::week => dur waithere;

// συνδυασμός
2::minute + 30::second => dur bar;
```

Εξέλιξη του χρόνου

Η πάροδος του χρόνου επιτρέπει κάθε φορά να εκτελούνται διαφορετικά shreds και με τον τρόπο αυτό να παράγεται ήχος με καθορισμένο τρόπο. Με την χρήση του τελεστή "+=>" σε μία χρονική διάρκεια επιτυγχάνεται εξέλιξη του χρόνου κατά ορισμένες χρονικές μονάδες της διάρκειας .Με την χρήση του "=>" για ένα χρονικό σημείο επιτυγχάνεται χρονική πρόοδος μέχρι εκείνο το χρονικό σημείο .Επίσης μπορεί να χρησιμοποιηθεί για ένα γεγονός με τον ίδιο τρόπο. Τα παρακάτω παραδείγματα εξηγούν τον τρόπο εξέλιξης του χρόνου.

```
// εξέλιξη του χρόνου για 1 δευτερόλεπτο

1::second => now;


//εξέλιξη του χρόνου κατά 100 millisecond

100::ms => now;
```

```
//εξέλιξη του χρόνου κατά 1 sample
1::samp =>now;

//εξέλιξη του χρόνου λιγότερο από ένα sample
.024::samp => now;
```

2.5.6 Συναρτήσεις.

Η έννοια των συναρτήσεων στην chuck είναι παρόμοια με αυτήν στις άλλες γλώσσες προγραμματισμού όπως οι Java,C και C++ .Για ακόμα μια φορά ο τελεστής (=>) μπορεί να χρησιμοποιηθεί για την κλήση συναρτήσεων.

2.5.7 Συγχρονισμός ,διεργασίες και shreds.

Η chuck είναι σε θέση να τρέξει πολλές διεργασίες ταυτόχρονα .Μια διεργασία στην chuck ονομάζεται shred. Όταν γίνεται spork ένα shred τότε δημιουργείται και προστίθεται μια νέα διεργασία στην εικονική μηχανή. Τα shreds μπορούν να εκτελεσθούν από διάφορα σημεία ενώ ακόμα και τα ίδια μπορούν να δημιουργήσουν νέα.

Η chuck υποστηρίζει επίσης συγχρονισμό χωρίς την άμεση παρέμβαση του χρήστη. Οποιοσδήποτε αριθμός shreds μπορεί αυτόματα να δρομολογήσει και να συγχρονιστεί χρησιμοποιώντας συγκεκριμένες χρονικές κατευθύνσεις. Κάθε shred δεν χρειάζεται απαραιτήτως να γνωρίζει τη ύπαρξη των υπόλοιπων shreds. Η εικονική μηχανή εξασφαλίζει με την σειρά της ,ότι οι υπολογισμοί επιτυγχάνονται σωστά σε όλο το σύστημα. Τόσο ο χρόνος όσο και ο συγχρονισμός είναι ντετερμινιστικά ορισμένος στην chuck.

Κλήση των Shreds

Τα νέα shreds μπορούν να καλεστούν είτε από την γραμμή εντολών είτε μέσα από τα ίδια τα προγράμματα. Για να καλεστεί ένα shred μέσα από τον κώδικα χρησιμοποιείται ο τελεστής **spork~** , ο ποίος έχει τις εξής ιδιότητες:

- Η λέξη κλειδί spork καλεί δυναμικά ένα shred μέσω μιας κλήσης συστήματος.
- Η συγκεκριμένη λειτουργία είναι συγχρονισμένη, δηλαδή το νέο shred είναι δρομολογημένο να εκτελεστεί αμέσως σε λογικό χρόνο, ξεκινώντας από την συνάρτηση που έχει γίνει spork.
- Το parent shred εξακολουθεί να εκτελείται μέχρι να γίνει yield.
- Όταν το parent shred υπάρχει, όλα τα παιδία του επίσης υπάρχουν.
- Η καλούμενη συνάρτηση επιστρέφει μια αναφορά στο νέο shred.

Το παρακάτω παράδειγμα ορίζει μία συνάρτηση και μετά καλεί ένα νέο shred για να ξεκινήσει την εκτέλεση του στην καθορισμένη συνάρτηση.

```
//ορίζω συνάρτηση go()
fun void go()
{

}

// ξεκινά ένα νέο shred από την go()
    spork ~go();
//ξεκινάει ένα νέο shred ,με αναφορά σε ένα νέο shred
spork ~go() => Shred @ bach;
```

2.5.8 Προγραμματίζοντας με Events.

Στην chuck τα events προέρχονται από την κλάση "Event" που περιέχεται σε αυτήν. Αφού κάποιος δημιουργήσει ένα event αυτό μπορεί να συγχρονιστεί στην chuck. Το event τοποθετείται στο τρέχον shred στην σχετική λίστα με τα events που βρίσκονται σε αναμονή και το shred αναστέλλεται. Όταν το event ενεργοποιείται , ένα ή περισσότερα shreds της λίστας αναμονής δρομολογούνται για να τρέξουν αμέσως. Αυτή η ενεργοποίηση μπορεί να προέρχεται από ένα άλλο shred ή από δραστηριότητες εκτός της εικονικής μηχανής όπως είναι ένα ηλεκτρικό βιολί που θα παρουσιαστεί σε επόμενη ενότητα ή MIDI.

Τα events μπορούν να ενεργοποιηθούν με δύο τρόπους, εξαρτάται από την συμπεριφορά τους. Η κλήση της signal() απελευθερώνει το πρώτο shred από την λίστα των events και τρέχει εκείνη την χρονική στιγμή. Αντίθετα ,με την κλήση της broadcast() απελευθερώνονται και δρομολογούνται όλα τα shreds από την ουρά με την σειρά που προστέθηκαν.

2.5.9. Αντικείμενα.

Η chuck δανείζεται χαρακτηριστικά για την διαχείριση των αντικειμένων τόσο από την C++ όσο και από την Java. Ορισμένα από αυτά είναι :

1. Μπορούν να οριστούν τυπικές κλάσεις ως νέοι τύποι και να ενσωματωθούν σε αντικείμενα.
2. Η chuck υποστηρίζει την έννοια της κληρονομικότητας , όπως ακριβώς χρησιμοποιείται και την Java.
3. Όλες οι μεταβλητές τύπου αντικειμένου λειτουργούν σαν αναφορές (reference στην java) .
4. Υπάρχει μία προεπιλεγμένη βιβλιοθήκη για την διαχείριση τους.
5. Όλα τα αντικείμενα κληρονομούνται από την κλάση Object όπως στην java.

Η κληρονομικότητα σε αντικειμενοστραφής γλώσσες προγραμματισμού επιτρέπει στον προγραμματιστή να πάρει μια υπάρχουσα κλάση και να την επεκτείνει ή να

τροποποιήσει την λειτουργία της. Με αυτόν τον τρόπο μπορεί να δημιουργηθεί μια ταξινομημένη ομάδα κλάσεων στην οποία όλοι μοιράζονται ένα συγκεκριμένο σύνολο συμπεριφορών .Παράλληλα αυτές μπορούν να εκτελεστούν με διαφορετικούς αλλά καλώς ορισμένους τρόπους. Έχει αναφερθεί ότι μια νέα κλάση κληρονομεί από μία άλλη κλάση δεδομένα χρησιμοποιώντας λέξεις κλειδιά. Η κλάση η οποία κληρονομείται από άλλες ονομάζεται κλάση «πατέρας» και αύτη που κληρονομεί ονομάζεται κλάση «παιδί». Η κλάση «παιδί» λαμβάνει όλα τα δεδομένα και τις συναρτήσεις από την πατρική κλάση ,αν και οι συναρτήσεις από την πατρική κλάση μπορούν να παρακαμφθούν. Επειδή στις κλάσεις «παιδιά» περιλαμβάνονται οι συναρτήσεις της κλάσης του πατέρα ,οι αναφορές σε περιπτώσεις κλάσεων παιδιών μπορούν να ανατεθούν σε αναφορές του τύπου της πατρικής κλάσης.

Ένα άλλο χαρακτηριστικό της κληρονομικότητας είναι ότι παρέχει έναν αποτελεσματικό τρόπο διαμοίρασης κώδικα μεταξύ των κλάσεων που έχουν παρόμοια συμπεριφορά. Καθώς αλλάζει ο τρόπος που μια καθορισμένη συμπεριφορά λειτουργεί, αλλάζουν και τα πρότυπα συμπεριφοράς που μπορούν να οριστούν.

2.5.10. Unit generators

Οι unit generators είναι αντικείμενα τα οποία κληρονομούν τα χαρακτηριστικά τους από την κλάση UGen .Μέσω του τελεστή "=>" μπορεί ένα αντικείμενο μιας κλάσης τύπου UGen να συνδεθεί με ένα άλλο. Πριν χρησιμοποιηθεί ένας unit generator χρειάζεται ενεργοποίηση . Στόχος τους είναι τα σήματα που παράγουν να χρησιμοποιηθούν ως σήματα ήχου ή σήματα ελέγχου. Η εικονική μηχανή βοηθάει στον έλεγχο του χρόνου και των unit generations.

Η αλλαγή και η τροποποίηση των unit generations μπορεί να γίνει οποιαδήποτε χρονική στιγμή και σε οποιοδήποτε σημείο του προγράμματος. Ο προγραμματιστής αρκεί να θέσει διαφορετικές παραμέτρους που θα τον βοηθήσουν να επιβάλλει τον έλεγχο των unit generations κάθε φορά.

Για τα σήματα εξόδου μπορούν να χρησιμοποιηθούν οι μετατροπείς dac και adc. Ο dac μετατρέπει ένα σήμα από ψηφιακό σε αναλογικό και το στέλνει στην συσκευή εξόδου. Ό adc μετατρέπει ένα σήμα από αναλογικό σε ψηφιακό .

2.5.11. Unit analyzers

Οι unit analyzers έχουν ως στόχο να αναλύουν τα σήματα εισόδου του ήχου και να παράγουν ένα επεξεργασμένο σήμα στην έξοδο. Είναι αντικείμενα και κληρονομούνται από την κλάση UAna. Μπορούν να συνδεθούν με τον τελεστή "=^" .Η συνάρτηση upchuck() που διαθέτουν πραγματοποιεί ανάλυση του σήματος όταν καλεστεί. Με την βοήθεια της εικονικής μηχανής παρέχεται ο έλεγχος τους στο πέρασμα του χρόνου.

Για την χρήση των unit analyzers απαιτείται η δήλωση τους όπως και στους unit generators. Αξίζει να σημειωθεί ότι είναι δυνατόν να δημιουργηθούν δίκτυα που περιλαμβάνουν μαζί unit generator και unit analyzers.

Κεφάλαιο 3

Βιολί

3.1.Ιστορική εξέλιξη

Οι πρόγονοι του βιολιού είναι το αναγεννησιακό όργανο Φιντλ, το Ρεμπέκ και η Λίρα ντα μπράτσο. Πήρε την κλασσική του μορφή στα μέσα του 16^{ου} αιώνα από τον διάσημο Ιταλό κατασκευαστή εγχόρδων οργάνων Andrea Amati και τελειοποιήθηκε αργότερα (ως το1750) από τον Niccolo Amati και τους μαθητές του Antonio Stradivari και Giuseppe Guarneri . Έκτοτε δεν έχουν σημειωθεί σημαντικές αλλαγές στο σχήμα στα υλικά κατασκευής και στην ακουστική δομή του βιολιού . Μεγάλη ήταν η συμβολή του Niccolo Paganini στην τεχνική του βιολιού, ο οποίος εισήγαγε νέες τεχνικές και εκμεταλλεύτηκε στο έπακρο τις ήδη υπάρχουσες . Το δοξάρι πήρε τη σημερινή του μορφή στα τέλη του 18^{ου} αιώνα στο Παρίσι από τον Francois Tourte .

Οι επαγγελματίες μουσικοί χρησιμοποιούν το βιολί ήδη από 15^{ο} αιώνα στο θέατρο, στο μπαλέτο και στους χορούς των ευγενών. Οι πρώτες τυπωμένες παρτιτούρες για βιολί προορίζονταν για γαμήλιους εορτασμούς ευγενών (Παρίσι 1581 , Φλορεντία 1582) . Η πρώτη σονάτα για βιολί του Τζίμα δημοσιεύτηκε το 1610. Οι σονάτες του Κορέλι λειτούργησαν ως σύνοψη των εξελίξεων του 17^{ου} αιώνα σ' αυτό τον τομέα και αποτελούσαν πρότυπο για διάφορους Ιταλούς και Γερμανούς συνθέτες , ανάμεσά τους οι Βιβάλντι ,Βιτάλι , Τέλμαν, Χαίντελ και Μπαχ . Πρότυπα για ανάλογες συνθέσεις του Μπαχ υπήρξαν τα έργα για βιολί των Τορέλι ,Κορέλι ,Αλμπινόνι , αλλά κυρίως του Βιβάλντι . Επιπλέον ο Μπαχ δημιούργησε με το έργο του « Έξι σόλο για βιολί χωρίς συνοδεία μπάσου» ένα ορόσημο στη συνθετική τεχνική για βιολί, το οποίο έργο απαιτεί από τον οργανοπαίκτη τεράστια δεξιοτεχνία και από τον ακροατή εξ ίσου σημαντική ικανότητα ηχητικής επεξεργασίας . Στο έργο αυτό εκτελούνται με ένα και μοναδικό βιολί, αξιοποιώντας την τεχνική των παραλείψεων, ακόμα και τετραφωνίες .

Μια άλλη εξέλιξη οδηγεί στην Γαλλία του 17^{ου} αιώνα από τον Λουλύ , στις σονάτες για βιολί με συνεχές βάσιμο, στην τρίο σονάτα και στη «σονάτα για 2 βιολιά» κι από εκεί στις σονάτες για πιάνο με συνοδεία βιολιού και άλλων εγχόρδων των Ραμό και Μότσαρτ. Μετά την ίδρυση του Κονσερβατόριου του Παρισιού παρουσιάστηκαν οι σημαντικές συνθέσεις που κυριαρχούν μέχρι και σήμερα στην εκμάθηση του βιολιού .

Ο Μότσαρτ έχει συμβάλει αποφασιστικά στην ανάπτυξη των μουσικών ειδών όπως το «κουαρτέτο εγχόρδων» και της χρήσης σολιστικού οργάνου με κουαρτέτο και κουιντέτο εγχόρδων, επηρεασμένος αρχικά από διάφορους συνθέτες , αργότερα κυρίως από τον Χάυντν , στον οποίο και αφιέρωσε 6 από τα κουαρτέτα του .Σημαντική είναι η συμβολή του επίσης σε έργα για βιολί και ορχήστρα . Έγραψε 5 κονσέρτα για βιολί σε ηλικία 19 ετών μέσα περίπου σε 10 μήνες , για το δεξιοτέχνη βιολονίστα Μπρουνέτι .Τα πρώτα δύο σε μάλλον παραδοσιακό στυλ , τα υπόλοιπα με διάφορους νεωτερισμούς .

Οι σπουδαιότερες συνθέσεις για σόλο βιολί του 19ου και 20ου αιώνα προέρχονται από συνθέτες που δεν ήταν βιολονίστες όπως οι εξής: Μπετόβεν , Μέντελσον , Σούμαν , Μπραμς ,Τσαϊκόφσκι ,Ντβόρζακ ,Μανσέ ,Σαιν-Σαν ,Μπάρτοκ , Σοστάκοβιτς , Ανδρέας Νεζερίτης κ.α. .Μερικοί βιολονίστες ανάμεσά τους ο κορυφαίος σε δεξιοτεχνία όλων των εποχών , Νικολό Παγκανίνι ,συνήθιζαν να συνθέτουν μουσικά έργα .Ο Παγκανίνι έχει συνθέσει και σονάτες για βιολί και κιθάρα.

Το βιολί μεταφέρθηκε από την κεντρική Ευρώπη στους βαλκανικούς λαούς και στους ελληνικούς πληθυσμούς των Βαλκανίων κυρίως από τους τσιγγάνους .Αποτελεί έκτοτε όργανο της δημοτικής μας μουσικής . Με την έλευση και διάδοσή του εκτόπισε τη λύρα που ήταν διαδεδομένη στα Βαλκάνια και την Μ. Ασία .Η χρήση της οποίας περιορίζεται τώρα στην Κρήτη . Πολύ συχνά χρησιμοποιείται δε το βιολί και σε συνθέσεις της σύγχρονης ελληνικής μουσικής.

Η χρήση ενισχυτών στο βιολί έγινε για πρώτη φορά στην δεκαετία του 1920 από τζαζ εκτελεστές όπως ο Stuff Smith .Τα πρώτα ηλεκτρικά βιολιά άρχισαν να πωλούνται στην δεκαετία του 1930 .Έκτοτε το ηλεκτρικό βιολί καθιερώθηκε ως μουσικό όργανο .Πολλές γνωστές εταιρίες μουσικών οργάνων άρχισαν να το εξελίσσουν ώστε να παράγει ήχο παρόμοιο με αυτόν του ακουστικού βιολιού. Σήμερα πολλοί μουσικοί το χρησιμοποιούν για να παράγουν ηλεκτρονική μουσική τόσο με την βοήθεια ενισχυτών όσο και με ηλεκτρονικούς υπολογιστές.

3.2. Περιγραφή ηλεκτρικού βιολιού.

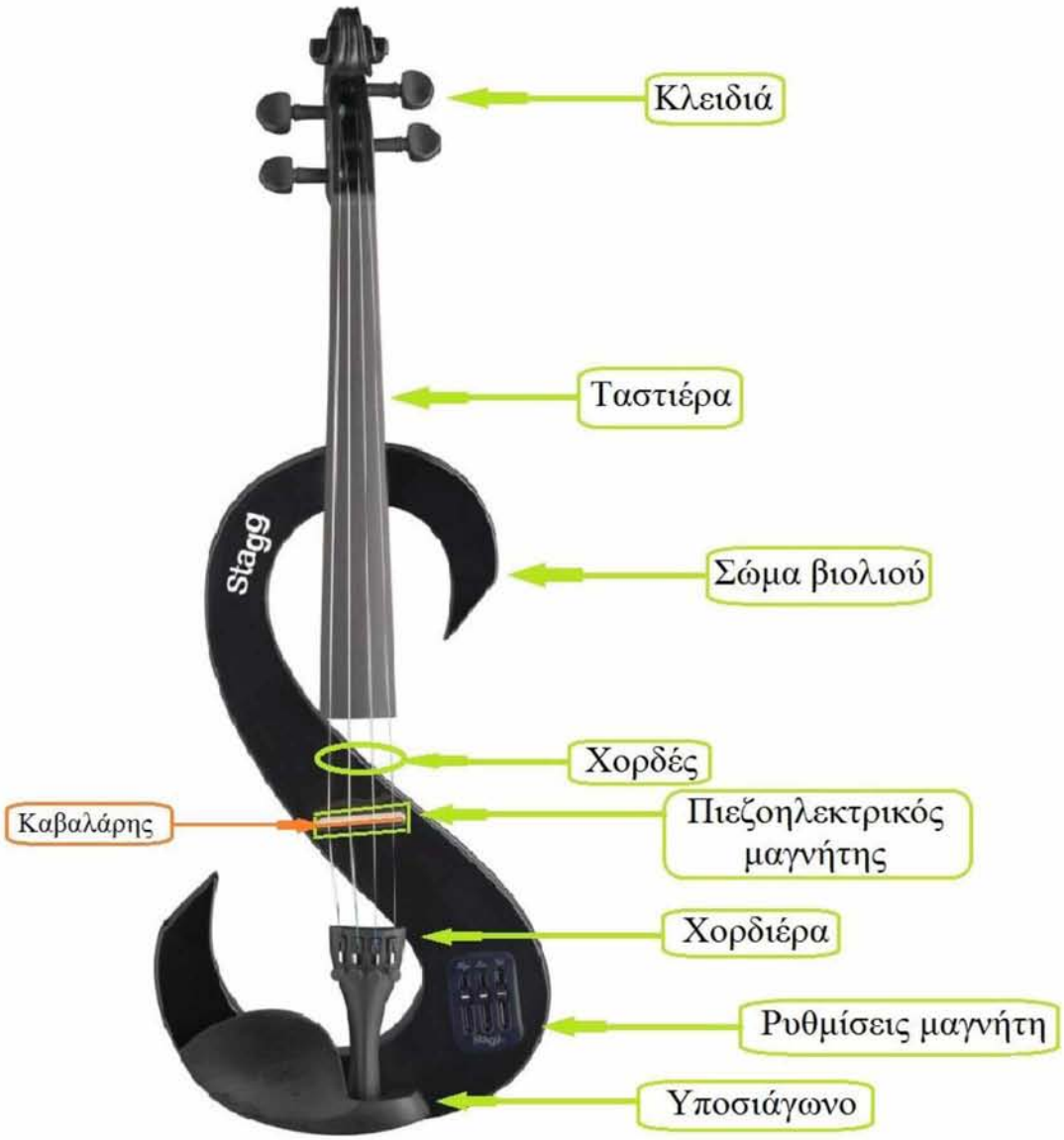
Ένα ηλεκτρικό βιολί [6] είναι ένα βιολί εξοπλισμένο με μία ηλεκτρονική έξοδο ήχου. Συνήθως ο όρος αναφέρεται σε ένα όργανο το οποίο αποτελείται από ένα στερεό σώμα και έναν μαγνήτη που ηλεκτροδοτείται. Μπορεί επίσης σε ένα ακουστικό βιολί να τοποθετηθεί ένας τύπος μαγνήτη δημιουργώντας έτσι ένα ηλεκτρο-ακουστικό βιολί.

Για να αποφευχθεί η ανάδραση του κοίλου σώματος ενός βιολιού ,η οποία οφείλεται στους συντονισμούς που παράγει η υψηλή ενίσχυση, χρησιμοποιείται ένα στερεό και συμπαγές υλικό για το σώμα. Αν και οι ταλαντώσεις καθορίζουν τελικά το ηχόχρωμα ενός μη ηλεκτρικού βιολιού ,στα ηλεκτρικά βιολιά αυτό μπορεί να ρυθμιστεί. Τα ηλεκτρικά βιολιά έχουν συνήθως μινιμαλιστικό σχεδιασμό για να εξασφαλιστεί το μικρό τους βάρος. Κατά την διαδικασία κατασκευής τους χρησιμοποιούνται υλικά όπως το γυαλί, το ξύλο και ο άνθρακας. Το μπράτσο του βιολιού δεν έχει τάστα ,γεγονός που κάνει δύσκολη την εκμάθησή του. Κατασκευάζεται από σφεντάμι και καταλήγει σε έναν κοχλία , στον οποίο ανοίγονται τρύπες για να την τοποθέτηση τα κλειδιών . Επάνω στο μπράτσο κολλιέται η ταστιέρα από έβενο ή ροδόξυλο . Από το ίδιο υλικό είναι και η χορδιέρα , στην οποία στηρίζονται οι χορδές . Ανάμεσα στην ταστιέρα και την χορδιέρα βρίσκεται ο καβαλάρης ο οποίος στηρίζεται σε δύο ποδαράκια και συγκρατείται στη θέση του από την πίεση των χορδών, έχει προορισμό να μεταφέρει τις ταλαντώσεις των χορδών στο καπάκι που βρίσκεται ο μαγνήτης.

Τα ηλεκτρικά σήματα περνάν μέσω ηλεκτρονικής επεξεργασίας με τον ίδιο τρόπο όπως στην ηλεκτρική κιθάρα, προκειμένου να επιτευχθεί ο επιθυμητός ήχος. Αυτή η διαδικασία μπορεί να περιλαμβάνει και την επίδραση παραμορφώσεων που αλλάζουν τον ήχο αναλόγως το είδος μουσικής που θέλει να εκτελέσει ο μουσικός.

Τα ηλεκτρικά βιολιά χρησιμοποιούν μαγνητικούς, πιεζοηλεκτρικούς ή ηλεκτροδυναμικούς δέκτες ήχου. Οι μαγνήτες με πηνίο που χρησιμοποιούνται και στην κιθάρα απαιτούν στο βιολί να υπάρχουν χορδές που να περιέχουν κράματα σιδήρου. Λίγοι μονοί μαγνήτες τύπου κιθάρας είναι διαθέσιμες σε τόσο μικρό μέγεθος ώστε να μπορούν να προσαρμοστούν στο μικρό σώμα του βιολιού και στην τοξοειδή διάταξη των χορδών. Για την χρήση τέτοιου τύπου μαγνητών σε ακουστικό βιολί απαιτείται μόνο να είναι ηλεκτρικά αγώγιμες οι χορδές που χρησιμοποιούνται.

Οι πιεζοηλεκτρικοί είσοδοι ήχου είναι οι πιο συχνές και φθηνές επιλογές στην κατασκευή ενός ηλεκτρικού βιολιού. Έχουν σχήμα κεραμικού δίσκου, κυλίνδρου ή πλαστικής μεμβράνης. Ανιχνεύουν φυσικές δονήσεις άμεσα και τοποθετούνται συνήθως μέσα στο σώμα ή πάνω στο σώμα για να λαμβάνουν τις δονήσεις που παράγει ο καβαλάρης. Στην **εικόνα 2** παρουσιάζονται τα μέρη ενός ηλεκτρικού βιολιού.



Εικόνα 2

Ένα ηλεκτρικό βιολί ενισχύεται ακριβώς όπως μια ηλεκτρική κιθάρα .Η έξοδος του ήχου μεταφέρεται μέσω ενός καλωδίου ήχου σε έναν ενισχυτή ή στην κάρτα ήχου του υπολογιστή. Για τον λόγο αυτό υπάρχουν λίγοι ενισχυτές που είναι διαμορφωμένοι μόνο για βιολί .Οι περισσότεροι βιολιστές χρησιμοποιούν ενισχυτές ηλεκτρικής κιθάρας ,οι οποίοι έχουν χρησιμοποιηθεί χρόνια και έχουν αξιόπιστο ήχο. Η **εικόνα 3** δείχνει το πίσω μέρος του ηλεκτρικού βιολιού και την έξοδο του.

Εικόνα 3

Το δοξάρι του βιολιού είναι ουσιαστικά ένα τόξο από ξύλο με το οποίο τεντώνονται 150-250 τρίχες αλόγου. Με την τριβή του δοξαριού στις χορδές παράγονται ταλαντώσεις που μετατρέπονται σε ήχο. Η **εικόνα 4** δείχνει τα μέρη του δοξαριού.

Εικόνα 4

3.3 Τεχνικές του βιολιού και παραγωγή ήχου.

Το βιολί τοποθετείται κάτω από την αριστερή πλευρά του σαγονιού του εκτελεστή. Με το τράβηγμα του δοξαριού με το δεξί χέρι πάνω στις χορδές παράγεται ο ήχος . Με το πάτημα των δακτύλων του αριστερού χεριού πάνω στη χορδή , στην περιοχή της ταστιέρας , ο εκτελεστής αλλάζει το μήκος τους και συνεπώς τις νότες.

Η έκταση ήχων του βιολιού καλύπτει τέσσερις οκτάβες , από το σολ μέχρι το σολ . Ο χαρακτηριστικός συνεχής ήχος του βιολιού δημιουργείται με γλίστρημα του δοξαριού πάνω στις χορδές . Με έλεγχο στην πίεση και την ταχύτητα του δοξαριού , το οποίο λειτουργεί ως προέκταση του ενός χεριού και με κατάλληλες κινήσεις, και τους δακτύλισμούς του άλλου χεριού στις χορδές ,ο βιολονίστας επηρεάζει:

- Τη δυναμική του κομματιού(από molto pianissimo μέχρι molto fortissimo).
- Τους εκφραστικούς χρωματισμούς του (παθητικά, τραγουδιστά, με κέφι, θριαμβευτικά ,θλιμμένα).
- Και την άρθρωση μεταξύ των φθόγγων , όπως το τονισμένο , το τρέμολο, τις τρίλιες κ.α.)

Με το βιολί είναι δυνατόν να εκτελεστούν επίσης διπλοί φθόγγοι και με κατάλληλες τεχνικές συνθέσεως , ακόμα και τετραφωνικές μελωδίες . Το βιολί μπορεί να υπογραμμίσει την παρουσία του ανάμεσα σε άλλα όργανα , ακόμα κι αν αυτά έχουν εκ κατασκευής ισχυρότερο ήχο .Όσο εντυπωσιακοί κι αν είναι όμως οι ήχοι του μεμονωμένου βιολιού , το σύνολο των βιολιών μιας ορχήστρας δίνει ένα διαφορετικό , πλούσιο και γεμάτο ήχο . Το βιολί έχει επίσης συχνά σημαντικό ρόλο σε κομμάτια για τζαζ, ροκ και ηλεκτρονικής μουσικής. Η χρήση του υπολογιστή μπορεί να προσαρμόσει τον ήχο αλλά και να τον παραμορφώσει. Δίνει επίσης την δυνατότητα στον μουσικό να πειραματιστεί και να ανακαλύψει νέες τεχνικές εξελίσσοντας έτσι την μουσική και τον τρόπο παιξίματος.

Κεφάλαιο 4

Σύνθεση μουσικής.

4.1. Βασικά χαρακτηριστικά σύνθεσης.

Η μουσική [7] είναι η επιστήμη ή η τέχνη που έχει ως σκοπό τον συνδυασμό και την ταξινόμηση τόνων και ήχων προκειμένου να παραχθεί μια σύνθεση .Η σύνθεση αυτή πρέπει να έχει συνοχή και να είναι μοναδική. Παρ' όλα αυτά για να παράγει κανείς μουσική είναι αναγκαίο να υπάρχει α) ένας τρόπος για την δημιουργία ήχων, β) μια διαδικασία ελέγχου του συγχρονισμού και της δημιουργίας ήχων, γ) επαρκής διορατικότητα στην ψυχολογία του ανθρώπου για να γίνουν κατανοητές οι έννοιες «μοναδικότητα και συνοχή» σε μια μουσική σύνθεση.

Στην εξέλιξη της παραγωγής ήχου έχουν συναινέσει πολλά υλικά όπως το ξύλο και τεντωμένες χορδές ,στα οποία μπορούν να προκληθούν ταλαντώσεις με διάφορους τρόπους και να προκαλέσουν ήχο στα πλαίσια του αποδεκτού ηχητικού εύρους. Η ποιότητα του ήχου που παράγεται εξαρτάται από τα παρακάτω συνδεδεμένα μεταξύ τους χαρακτηριστικά :

- Η τονικότητα, η οποία είναι η υποκειμενική εντύπωση της τοποθέτησης ενός μουσικού τόνου στα πλαίσια του εύρους και των συχνοτήτων που αντιλαμβάνεται ο άνθρωπος. Γενικά ,τόνοι οι οποίοι είναι αποτέλεσμα γρήγορων δονήσεων έχουν ψηλότερη τονικότητα από αυτούς που παράγονται από πιο αργές. Η τονικότητα συνήθως είναι ένα υποκειμενικό χαρακτηριστικό μιας περιοδικής δόνησης η οποία αναλογεί στον λογάριθμο της θεμελιώδης αντίληψης της συχνότητας. Αξίζει να σημειωθεί ότι η τονικότητα επηρεάζεται και από άλλα χαρακτηριστικά του ήχου και όχι μόνο από την συχνότητα .Τέτοια είναι η ένταση και η αλλαγή της απόστασης της πηγής με τον ακροατή.
- Η ένταση, η οποία είναι η υποκειμενική εντύπωση της δύναμης και του εύρους ενός ήχου. Η ένταση ξεκάθαρα επηρεάζεται από συγκεκριμένους συντελεστές συχνότητας του ήχου, αφού το αυτί δεν είναι το ίδιο ευαίσθητο σε όλο το εύρος των ηχητικών συχνοτήτων.
- Η ποιότητα του τόνου, κάθε ήχος που συναντάται στην φύση, συμπεριλαμβανομένων και μουσικών ήχων, συνθέτεται από τον συνδυασμό ταλαντώσεων πολλών συχνοτήτων, πλάτους και φάσεων. Το χαρακτηριστικό της χρονικής εξέλιξης ενός ηχητικού φάσματος είναι υπεύθυνο για την συνολική ποιότητα του ήχου.
- Οι χωρικοί παράγοντες, οι οποίοι αναφέρονται σε συγκριμένη κατεύθυνση, απόσταση και τροχιά μιας ηχητικής πηγής μέσα σε μια συγκεκριμένη ακουστική που περιβάλλει τον ακροατή. Ο ανθρώπινος ακουστικός μηχανισμός κατασκευάζεται με τέτοιον τρόπο ώστε να είναι ευαίσθητος σε αυτές τις εκφάνσεις του ήχου.

Έτσι με βάση τους παραπάνω παράγοντες μπορεί να οριστεί η έννοια της μελωδίας ως η διαδοχή των ήχων με διαφορετική οξύτητα, ασχέτως αν αυτή η διαδοχή είναι εύηχη ,εύκολη ή δύσκολη. Σε αντίθεση με την μελωδία που είναι η οριζόντια ανάπτυξη ,δηλαδή το διαδοχικό άκουσμα ήχων ,η αρμονία στην μουσική είναι το κάθετο άκουσμα .Δηλαδή η συνήχηση περισσοτέρων από δύο ήχους. Στην μελωδία κάθε ήχος μπορεί να ακολουθείται από οποιονδήποτε άλλον, έτσι και στην αρμονία κάθε ήχος μπορεί να ακούγεται ταυτόχρονα με κάποιον άλλον. Η έννοια του ρυθμού είναι πιο δύσκολη να προσδιοριστεί. Στην μουσική ,ο ρυθμός είναι η εναλλαγή της διάρκειας του ήχου. Η περιοδικότητα η οποία μπορεί να εμφανίζεται σε κάθε μουσική, τραγούδι ή παίξιμο ενός μουσικού οργάνου εκφράζεται περισσότερο με την έννοια του μέτρου.

Το μοναδικό στοιχείο που δεν μπορεί να απουσιάσει για την σύνθεση μιας μελωδίας ή αρμονίας είναι ο ρυθμός. Ο ρυθμός είναι το απαραίτητο και αυτονόητο στοιχείο το οποίο συντελεί στην ύπαρξη της μουσικής ,όπως το οξυγόνο στην ύπαρξη της ζωής. Αξίζει να αναφερθεί ότι υπάρχουν μουσικά ρυθμικά όργανα τα οποία δεν μπορούν να παίξουν μελωδική ή αρμονική μουσική. Τα μουσικά όργανα που συντελούν στην σύνθεση μιας μελωδίας μπορούν να χωριστούν σε τρείς βασικές κατηγορίες. Τα έγχορδα, τα πνευστά και τα κρουστά.

4.2 Σύνθεση μουσικής με την γλώσσα προγραμματισμού Chuck.

Σύμφωνα με τις βασικές αρχές σύνθεσης μουσικής και τις δυνατότητες της γλώσσας προγραμματισμού chuck που αναπτύχθηκαν θα παρουσιαστεί ένα παράδειγμα εναρμόνισης και σύνθεσης μιας μελωδίας στην chuck. Η μελωδία που προγραμματίστηκε στην chuck είναι η «Ωδή στην Χαρά» [8] (Ode an die Freude) .Είναι η ωδή που γράφτηκε από τον Γερμανό ποιητή και ιστορικό Φρίντριχ Σίλερ. Η ωδή αυτή που περιλαμβάνει 108 στίχους, στη τελική του μορφή, έγινε ευρύτερα γνωστή όταν μελοποιήθηκε από τον Λούντβιχ βαν Μπετόβεν το 1824, ο οποίος την ενέταξε στο τέταρτο και τελευταίο μέρος της Ενάτης συμφωνίας του, ως χορωδιακή συμφωνία, για τέσσερις σόλο φωνές , χορωδία και ορχήστρα σε ρε μείζονα. Λιγότερο γνωστές μελοποιήσεις είναι αυτή του Φραντς Σούμπερτ (για φωνή και πιάνο του 1815) και του Πιοτρ Τσαϊκόφσκι (για σόλο φωνές, χορωδία και ορχήστρα και στίχους στα ρώσικα, του 1865).

Το 1972 υιοθετήθηκε η σύνθεση του Μπετόβεν ως ύμνος του Ευρωπαϊκού Συμβουλίου. Το 1985 οι Ευρωπαίοι ηγέτες υιοθέτησαν ομοίως την σύνθεση ως επίσημο ύμνο της Ευρωπαϊκής Ένωσης ,χωρίς όμως τους στίχους. Η συγκεκριμένη μελωδία έχει χρησιμοποιηθεί σε ταινίες των Beatles και του Στάνλεϋ Κιούμπρικ καθώς σε αθλητικές διοργανώσεις όπως στο διασυλλογικό πρωτάθλημα νοτίου Αμερικής και στους θερινούς και χειμερινούς Ολυμπιακούς αγώνες.

Ο ρυθμός του κομματιού είναι τέσσερα τέταρτα που σημαίνει ότι σε κάθε μουσικό μέτρο αντιστοιχούν τέσσερις χρόνοι. Η τονικότητα που επιλέχθηκε είναι η Ντο μείζονα. Η εναρμόνιση της μελωδίας έγινε για τέσσερα μουσικά όργανα ,βιολί, βιμπράφωνο, σαξόφωνο και κλαρινέτο τα οποία βρίσκονται στην υπέρ-κλάση STK-Instruments της chuck. Για να δημιουργηθεί μια εύηχη και αρμονικά γεμάτη σύνθεση επιλέχθηκαν ως unit generation,

δηλαδή σήματα ήχου, όργανα και από τις τρείς βασικές κατηγορίες μουσικών οργάνων.

Φυσικά ο κώδικας ενός προγράμματος δεν μπορεί να αντικαταστήσει πλήρως μια παρτιτούρα. Παρόλα αυτά η διαδικασία μεταφοράς μιας παρτιτούρας στην chuck δίνει πολλά πλεονεκτήματα στον συνθέτη. Αρχικά το πρόγραμμα είναι εύκολα κατανοητό από κάποιον που είναι εξοικειωμένος με τέτοιου είδους συστήματα μουσικής. Η εκτέλεση του κώδικα θα έχει ως αποτέλεσμα την ακριβή χρονικά αλλά και τονικά αναπαραγωγή της σύνθεσης. Από την άλλη ο ερμηνευτής μιας παρτιτούρας μπορεί να βάλει το προσωπικό του συναίσθημα και να εκφραστεί μέσα από το μουσικό όργανο. Τέλος με την chuck μπορούν να γίνουν κάθε στιγμή αλλαγές όπως στην τονικότητα και στην μετατροπή ολόκληρων μέτρων χωρίς να επηρεαστεί το υπόλοιπο κομμάτι του κώδικα.

4.2.1. Βασικά μουσικά, τονικά και ρυθμικά χαρακτηριστικά της σύνθεσης σε chuck.

Στην chuck οι νότες συμβολίζονται με ακεραίους που έχουν εύρος από 0 έως 127. Στην συγκεκριμένη σύνθεση χρησιμοποιήθηκαν δύο οκτάβες ,ξεκινώντας από το ντο που έχει την τιμή MIDI από 60 μέχρι 83. Η τονικότητα στην οποία βασίστηκε η μελώδια είναι η Ντο μείζονα. Αυτή η σκάλα δεν έχει αλλοιώσεις διέσεις ή υφέσεις και τα διαστήματα μεταξύ των διαδοχικών νοτών σε μια οκτάβα είναι: τόνος, τόνος, ημιτόνιο, τόνος ,τόνος, ημιτόνιο. Ακολουθεί η **εικόνα 5** με την κλίμακα της Ντο μείζονα :

Εικόνα 5

Για τον συμβολισμό της αξίας της κάθε νότας, δηλαδή την διάρκεια την οποία θα έχει, χρησιμοποιήθηκαν πάλι ακέραιοι αριθμοί όπως φαίνεται στον **πίνακα 4**:

Συμβολική αναπαράσταση στην chuck.	**Αξία νότας**
1	Ολόκληρο
2	Μισό
4	Τέταρτο
8	Όγδοο

Πίνακας 4

Για να μπορεί να γίνει ο χειρισμός των διάφορων διακυμάνσεων της ταχύτητας που θα έχει το μουσικό κομμάτι έχει οριστεί μία μεταβλητή **"tempo"**. Αυτή αναπαριστά τον αριθμό των παλμών ανά λεπτό ή τις μονάδες μετρονόμου τύπου Maelzel ,οι οποίες πήραν αυτό το όνομα προς τιμήν του εφευρέτη του μετρονόμου Johann Nepomuk Mälze.

Ακολουθεί ο **πίνακας 5** με τα πιο κοινά είδη ρυθμικών ταχυτήτων και των αντίστοιχων τιμών παλμών ανά λεπτό.

Τέμπο	**Παλμοί ανά λεπτό**
Largo	40
Adagio	60
Moderato	80
Allegretto	100
Allegro	120
Presto	160

Πίνακας 5

Έτσι ο κώδικας κάθε νήματος, που ουσιαστικά αναπαριστά και ένα μουσικό όργανο ,βασίζεται και περιέχει τα παραπάνω χαρακτηριστικά. Προκειμένου να μπορέσει να προσομοιωθεί η παρτιτούρα σε κώδικα της γλώσσας προγραμματισμού chuck για τον ρυθμό, τις αξίες των νοτών και την αναπαράσταση των νοτών της κλίμακας έχουν καθοριστεί τα παρακάτω κομμάτια κώδικα.

Για το τέμπο η μεταβλητή **"tempo"** η οποία λαμβάνει τις παραπάνω τιμές του πίνακα:

```
120 => int tempo;
```

Για την αξία κάθε νότας [9]:

```
dur duration[9];
240000::ms / ( 1 * tempo )  => duration[1]; // ολόκληρο
240000::ms / ( 2 * tempo )  => duration[2]; // μισό
240000::ms / ( 4 * tempo )  => duration[4]; // τέταρτο
240000::ms / ( 8 * tempo )  => duration[8]; // όγδοο
(duration[4] + duration[8]) => duration[5]; // παρεστιγμένο τέταρτο
(duration[2] + duration[4]) => duration[3]; // παρεστιγμένο μισό
```

Η χρήση ακέραιων για την αναπαράσταση των νοτών είναι μια επώδυνη και

χρονοβόρα διαδικασία, γιατί ο συνθέτης καλείται να θυμάται την τιμή κάθε νότας που θα χρησιμοποιήσει και μπορεί να τον οδηγήσει σε πολλά λάθη. Αντικαταστάθηκαν λοιπόν με μεταβλητές. Κάθε νότα θα έχει μια μεταβλητή με το όνομα της στην αγγλική ορολογία .Η πρώτη οκτάβα θα είναι με μικρά γράμματα και η δεύτερη με κεφαλαία. Στις νότες που περιέχουν αλλοιώσεις αντί για " #" θα ακολουθείται μετά από το όνομά της το γράμμα **"s"** .

Ο κώδικας για τον συμβολισμό των νοτών :

```
60 => int c;     72 => int C;
61 => int cs;    73 => int Cs;
62 => int d;     74 => int D;
63 => int ds;    75 => int Ds;
64 => int e;     76 => int E;
65 => int f;     77 => int F;
66 => int fs;    78 => int Fs;
67 => int g;     79 => int G;
68 => int gs;    80 => int Gs;
69 => int a;     81 => int A;
70 => int as;    82 => int As;
71 => int b;     83 => int B;
```

Η αναπαράσταση κάθε μελωδίας στην γλώσσα προγραμματισμού chuck γίνεται με την βοήθεια ενός δισδιάστατου πίνακα ακεραίων. Στην πρώτη διάσταση βρίσκεται μια μεταβλητού τύπου λίστα από νότες. Στην δεύτερη διάσταση ένα ζευγάρι αριθμών όπου ο πρώτος αναπαριστά την νότα και ο δεύτερος την αξία της ,δηλαδή την διάρκειά της.

Ακολουθεί η αναπαράσταση σε κώδικα chuck της κλίμακας Ντο μείζονα της **εικόνας 5**:

```
[ [c,4], [d,4], [e,4], [f,4], [g,4], [a,4], [b,4], [C,4] ]@=> int DoMajore [] [] ;
```

Επίσης η συνάρτηση **"playNote"** λαμβάνει ως ορίσματα την νότα και την αξία της από τον πίνακα που περιέχει ολόκληρη την μελωδία και την μετατρέπει σε ήχο σε πραγματικό χρόνο. Αυτή η μετατροπή γίνεται με την συνάρτηση **"mtof"** της Std βιβλιοθήκης η οποία λαμβάνει μια νότα (midi) και την μετατρέπει σε συχνότητα ήχου.

```
fun void playNote( int n, int d ) {

        std.mtof(n + trans) => bow.freq;
       duration[d] => now;
}
```

Η μεταβλητή **"trans"** χρησιμεύει στην εύκολη και γρήγορη αλλαγή τονικότητας της μελωδίας. Δηλαδή μεταφέρει την νότα που είναι έτοιμη να εκτελεστεί κάποια ημιτόνια πάνω

ή κάτω. Στην συγκεκριμένη σύνθεση έχει τεθεί ίση με το μηδέν.

```
0=>int trans;
```

4.2.2. Εκτέλεση της μουσικής σύνθεσης με την χρήση νημάτων “Shreds”.

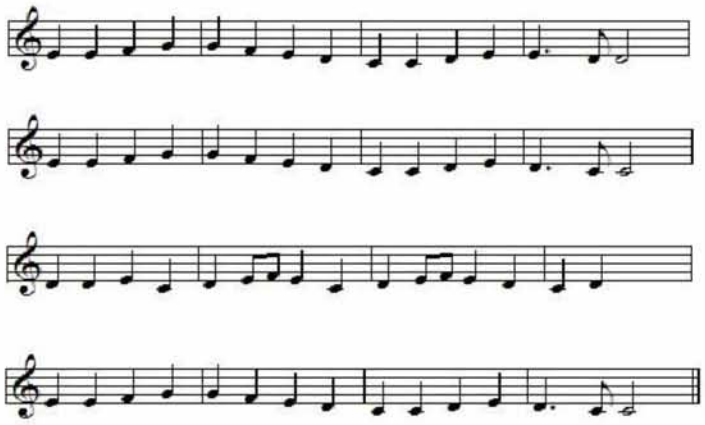
Η γλώσσα προγραμματισμού chuck δίνει την δυνατότητα εκτέλεσης πολλών νημάτων ταυτόχρονα. Έτσι έγινε εναρμόνιση της βασικής μελωδίας του Beethoven για τέσσερα μουσικά όργανα. Την βασική μελωδία την εκτελεί το βιολί, για τον ρυθμό χρησιμοποιήθηκε ένα βιμπράφωνο και την ορχήστρα της chuck συμπληρώνουν δύο πνευστά το σαξόφωνο και το κλαρινέτο.

Εξαιτίας της ταυτόχρονης πολυνηματικής εκτέλεσης και αναπαριστώντας το κάθε μουσικό όργανο σε ένα ξεχωριστό νήμα η μελώδια «ωδή της χαράς» έγινε πολυφωνική. Το κάθε νήμα μπορεί να διαχειρίζεται σαν μια ξεχωριστή οντότητα εκτελώντας την μελωδία του. Επίσης δίνεται στον συνθέτη η δυνατότητα να συγχρονίσει τα νήματα-μουσικά όργανα και να εκτελέσει το καθένα την χρονική στιγμή που ταιριάζει στην μελωδία.

Αρχικά έγινε η μετατροπή της κύριας μελωδίας από την παρτιτούρα σε εντολές γλώσσας προγραμματισμού chuck. Ο κώδικας γράφτηκε σε ξεχωριστό νήμα και ο στόχος είναι το ηχητικό αποτέλεσμα μετά την εκτέλεση του να μοιάζει με τον ήχο εγχόρδου όπως το βιολί. Γι’ αυτόν το λόγο χρησιμοποιήθηκε το κατάλληλο unit generator από την κατηγορία stk-instruments της chuck.

Η μουσική παρτιτούρα για βιολί που μεταφέρθηκε σε κώδικα chuck είναι η εξής (**εικόνα 6**):

ΩΔΗ ΣΤΗΝ ΧΑΡΑ- ΒΙΟΛΙ
Beethoven



Εικόνα 6

Το νήμα-Shred που παράγει μέσα από τον υπολογιστή την εκτέλεση της παρτιτούρας της **εικόνας 6** με τον ήχο του βιολιού και προγραμματίστηκε σε chuck είναι το εξής :

Shred «Ωδή στην χαρά» για βιολί.

```
Bowed bow => dac;
0.6 => bow.gain;
120 => int tempo;

dur duration[9];
240000::ms / ( 1 * tempo )  => duration[1]; // whole
240000::ms / ( 2 * tempo )  => duration[2]; // half
240000::ms / ( 4 * tempo )  => duration[4]; // quarter
240000::ms / ( 8 * tempo )  => duration[8]; // eighth
(duration[4] + duration[8]) => duration[5]; // dotted quarter
(duration[2] + duration[4]) => duration[3]; // dotted half
//notes
60 => int c;     72 => int C;
61 => int cs;    73 => int Cs;
62 => int d;     74 => int D;
63 => int ds;    75 => int Ds;
64 => int e;     76 => int E;
65 => int f;     77 => int F;
66 => int fs;    78 => int Fs;
67 => int g;     79 => int G;
68 => int gs;    80 => int Gs;
69 => int a;     81 => int A;
70 => int as;    82 => int As;
71 => int b;     83 => int B;
//melody in 3 parts

[ [e,4],[e,4],[f,4],[g,4],
 [g,4],[f,4],[e,4],[d,4],
 [c,4],[c,4],[d,4],[e,4],
 [e,5],[d,8],[d,2] ] @=> int mel_1[][];

[ [e,4],[e,4],[f,4],[g,4],
  [g,4],[f,4],[e,4],[d,4],
  [c,4],[c,4],[d,4],[e,4],
  [d,5],[c,8],[c,2] ] @=> int mel_2[][];

[ [d,4],[d,4],[e,4],[c,4],
  [d,4],[e,8],[f,8],[e,4],
  [c,4],[d,4],[e,8],[f,8],
  [e,4],[d,4],[c,4],[d,4],[g-12,2]] @=> int mel_3[][];

//for transporto
0 => int trans;
```

```
//function for playing the notes
fun void playNote( int n, int d ) {

    std.mtof(n + trans) => bow.freq;
    duration[d] => now;
     .6 => bow.noteOn;
}
for( 0 => int j; j<2; j++){

    for( 0 => int i; i < mel_1.cap(); i++) {
        //choice the way that will sound one note
        0.8=> bow.bowPosition;
        0.01=> bow.bowPressure;
        0.01=> bow.vibratoGain;
        10=> bow.vibratoFreq;
        0.5=>bow.startBowing;
        0.7=>bow.volume;
        //play the note
        playNote(mel_1[i][0], mel_1[i][1]);
    }

    for( 0 => int i; i < mel_2.cap(); i++) {
        //choice the way that will sound one note
        0.9=> bow.bowPosition;
        0.01=> bow.bowPressure;
        0.01=> bow.vibratoGain;
        10=> bow.vibratoFreq;
        0.5=>bow.startBowing;
        0.7=>bow.volume;
        //play the note
        playNote(mel_2[i][0], mel_2[i][1]);
    }

    for( 0 => int i; i < mel_3.cap(); i++) {
        //choice the way that will sound one note
        0.4=> bow.bowPosition;
        0.01=> bow.bowPressure;
        0.01=> bow.vibratoGain;
        8=> bow.vibratoFreq;
        0.5=>bow.startBowing;
        0.5=>bow.volume;
        //play the note
        playNote(mel_3[i][0], mel_3[i][1]);
    }

    for( 0 => int i; i < mel_2.cap(); i++) {
        //choice the way that will sound one note
        0.9=> bow.bowPosition;
        0.01=> bow.bowPressure;
```

```
        0.01=> bow.vibratoGain;
        10=> bow.vibratoFreq;
        0.5=>bow.startBowing;
        0.8=>bow.volume;
        //play the note
        playNote(mel_2[i][0], mel_2[i][1]);
    }
}

1.0=>bow.volume;
playNote( c, 1 );
.6 => bow.noteOff;
```

Στον παραπάνω κώδικα αρχικά γίνεται δήλωση του μουσικού οργάνου που θα ακουστεί στην έξοδο της εκτέλεσης. Η υπέρ-κλάση STK instruments που παρέχει η chuck διαθέτει μια μεγάλη γκάμα μουσικών οργάνων. Το μουσικό όργανο που χρησιμοποιήθηκε σε αυτό το κομμάτι κώδικα είναι έγχορδο με δοξάρι (Bow) όπως το βιολί. Στην συνέχεια με την επιλογή **"gain"** ρυθμίζεται το πλάτος του σήματος εξόδου (dac). Μετά από τις αναγκαίες δηλώσεις για την αξία κάθε νότας και των μεταβλητών τους ,όπως παρουσιάστηκε στην προηγούμενη υποενότητα, γίνεται η μετατροπή της μελωδίας. Η παρτιτούρα μπορεί να χωριστεί σε τρία μέρη. Κάθε μέρος περιέχει τέσσερα μουσικά μέτρα. Αυτός ο διαχωρισμός έγινε για την καλύτερη διαχείριση της μελωδίας και για να μπορέσει να αποδοθεί με τον καλύτερο τρόπο η δυναμική του κομματιού. Κάθε μέρος καταχωρείται σε έναν δισδιάστατο πίνακα. Έτσι τα πρώτα τέσσερα μέτρα μετατρέπονται από νότες στο πεντάγραμμο σε νότες στην chuck και καταχωρούνται στον πίνακα "mel_1" ,τα επόμενα τέσσερα στον πίνακα "mel_2" .Άλλα τέσσερα μέτρα στον πίνακα "mel_3". Τα τελευταία τέσσερα μέτρα την παρτιτούρας είναι ίδια με τα πρώτα τέσσερα άρα για την εκτέλεση τους χρησιμοποιείται ο "mel_1" .

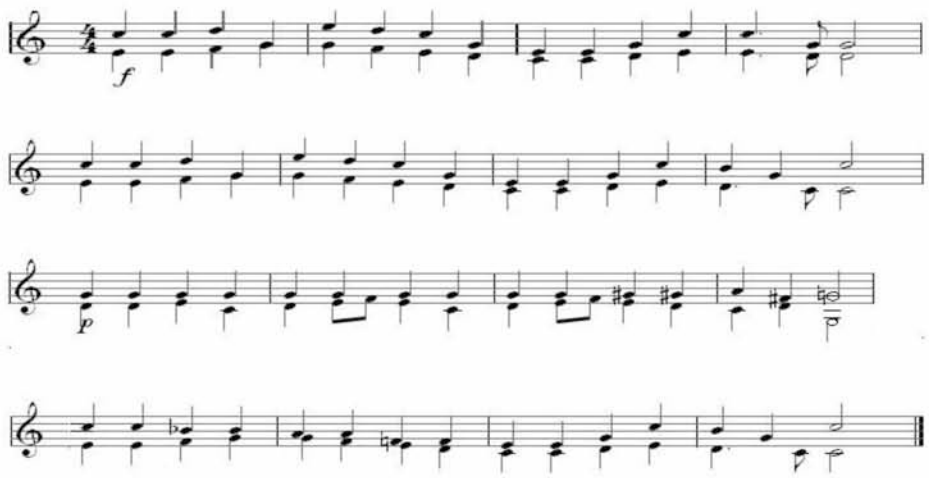
Κατόπιν με την βοήθεια της δομής επανάληψης "for" διατρέχονται όλα τα στοιχεία κάθε πίνακα τα οποία στέλνονται ως ορίσματα στην συνάρτηση "playNote" και παράγουν ήχο στην έξοδο. Αξίζει να σημειωθεί ότι πριν την κλήση της συνάρτησης γίνεται χρήση των εργαλείων που παρέχει η κλάση "Bow" . Έτσι για την βέλτιστη επιθυμητή ποιότητα ήχου δόθηκαν οι επιθυμητές τιμές στην **".vibradoFreq"** και **".vibradoGain"** ώστε να προσδιοριστεί η ταλάντωση που θα προκαλέσει το βιμπράτο. Για την θέση του δοξαριού και την πίεση του προσδιορίστηκαν τιμές στις **".bowPosition"** και **".bowPressure"** . Η ένταση ρυθμίζεται με την επέκταση **".volume"** .Στην συνάρτηση "playNote" για την ενεργοποίηση όλων των παραπάνω προτιμήσεων και την εκτέλεση της νότας αρκεί να καλεστεί η **".noteOn"**.

Προσεγγίζοντας μουσικά το αποτέλεσμα της εκτέλεσης ο ακροατής θα διαπιστώσει ότι υπάρχει μια διακύμανση στην ένταση και την δυναμική του κομματιού. Δηλαδή στα πρώτα οχτώ μέτρα η μελωδία παίζεται δυνατά "*forte*" ενώ μετά πιο σιγά "*piano*". Η chuck έχει «χτιστεί» με τέτοιον τρόπο ώστε το αποτέλεσμα που παράγει ο υπολογιστής να προσεγγίζει τον ήχο του μουσικού οργάνου.

Ο ρυθμός σε μία μουσική σύνθεση εκτελείται συνήθως από ένα κρουστό όργανο. Στην εναρμόνιση που έγινε επιλέχθηκε το βιμπράφωνο . Ακολουθεί η μουσική παρτιτούρα

για βιμπράφωνο που μεταφέρθηκε σε κώδικα chuck **εικόνα** 7:

ΩΔΗ ΣΤΗΝ ΧΑΡΑ- ΒΙΜΠΡΑΦΩΝΟ
Beethoven



Εικόνα 7

Η μελωδία του βιμπράφωνου είναι η πάνω ,δηλαδή η πιο ψηλή τονικά, ενώ από κάτω είναι του βιολιού. Επίσης στην παρτιτούρα διαφαίνεται και η δυναμική που επιθυμεί ο συνθέτης. Το "*f*" σημαίνει δυνατά και το "*p*" σιγά.

Το νήμα-Shred που παράγει μέσα από τον υπολογιστή την εκτέλεση της παραπάνω παρτιτούρας (**εικόνα** 7) με τον ήχο του βιμπράφωνου και προγραμματίστηκε σε chuck είναι το εξής :

Shred «Ωδή στην χαρά» για βιμπράφωνο.

```
ModalBar bar => dac;

120 => int tempo;

dur duration[9];
240000::ms / ( 1 * tempo )  => duration[1]; // whole
240000::ms / ( 2 * tempo )  => duration[2]; // half
240000::ms / ( 4 * tempo )  => duration[4]; // quarter (crotchet)
240000::ms / ( 8 * tempo )  => duration[8]; // eighth (quaver)
(duration[4] + duration[8]) => duration[5]; // dotted quarter
(duration[2] + duration[4]) => duration[3]; // dotted half

//notes
60 => int c;     72 => int C;
61 => int cs;    73 => int Cs;
```

```
62 => int d;     74 => int D;
63 => int ds;    75 => int Ds;
64 => int e;     76 => int E;
65 => int f;     77 => int F;
66 => int fs;    78 => int Fs;
67 => int g;     79 => int G;
68 => int gs;    80 => int Gs;
69 => int a;     81 => int A;
70 => int as;    82 => int As;
71 => int b;     83 => int B;

//melody in four parts
[[C,4], [C,4], [D,4], [g,4],
 [E,4], [D,4], [C,4], [g,4],
 [e,4], [e,4], [g,4], [C,4],
 [C,5], [g,8], [g,2]] @=> int mel_1[][];

[[C,4], [C,4], [D,4], [g,4],
 [E,4], [D,4], [C,4], [g,4],
 [e,4], [e,4], [g,4], [C,4],
 [b,5], [g,8], [C,2]] @=> int mel_2[][];

[[g,4], [g,4], [g,4], [g,4],
 [g,4], [g,4], [g,4], [g,4],
 [g,4], [g,4], [gs,4], [gs,4],
 [a,4], [fs,4], [g,2]] @=> int mel_3[][];

[[C,4], [C,4], [as,4], [as,4],
 [a,4], [a,4], [f,4], [f,4],
 [e,4], [e,4], [g,4], [C,4],
 [b,4], [g,4], [C,2]] @=> int mel_4[][];

//for transporto
0 => int trans;

//function for playing the notes
fun void playNote( int n, int d ) {

    std.mtof(n + trans) => bar.freq;
    duration[d] => now;
     .6 => bar.noteOn;
}
for( 0 => int j; j<2; j++){

    for( 0 => int i; i < mel_1.cap(); i++) {
        //choice the way that will sound one note
        1 => bar.preset;
        0.01=> bar.stickHardness;
        0.02=> bar.strikePosition;
        0.5 => bar.vibratoGain;
```

```
        12 => bar.vibratoFreq;
        0.7=> bar.volume;
        0.09 => bar.directGain;
        0.9 => bar.masterGain;
        //play the note
        playNote(mel_1[i][0], mel_1[i][1]);
    }

    for( 0 => int i; i < mel_2.cap(); i++) {
        //choice the way that will sound one note
        1 => bar.preset;
        0.01=> bar.stickHardness;
        0.02=> bar.strikePosition;
        0.5 => bar.vibratoGain;
        12 => bar.vibratoFreq;
        0.7=> bar.volume;
        0.09 => bar.directGain;
        0.9=> bar.masterGain;
        //play the note
        playNote(mel_2[i][0], mel_2[i][1]);
    }

    for( 0 => int i; i < mel_3.cap(); i++) {
        //choice the way that will sound one note
        1 => bar.preset;
        0.3=> bar.stickHardness;
        0.4=> bar.strikePosition;
        0.5 => bar.vibratoGain;
        15 => bar.vibratoFreq;
        0.5=> bar.volume;
        0.7 => bar.directGain;
        0.7 => bar.masterGain;
        //play the note
        playNote(mel_3[i][0], mel_3[i][1]);
    }

    for( 0 => int i; i < mel_2.cap(); i++) {
        //choice the way that will sound one note
        1 => bar.preset;
        0.1=> bar.stickHardness;
        0.2=> bar.strikePosition;
        0.5 => bar.vibratoGain;
        12 => bar.vibratoFreq;
        0.7=> bar.volume;
        0.7 => bar.directGain;
        0.9 => bar.masterGain;
        //play the note
        playNote(mel_4[i][0], mel_4[i][1]);
    }
}
```

Η υπερκλάση STK instrument της chuck με την κλάση ModalBar προσφέρει στον χρήστη ένα μεγάλο αριθμό κρουστών οργάνων. Μεταξύ αυτών και το βιμπράφωνο. Η επιλογή αυτού του κρουστού οργάνου έγινε γιατί εκτός από τον ρυθμό που μπορεί να κρατάει ,εκτελεί με εύηχο τρόπο και μια μελωδία.

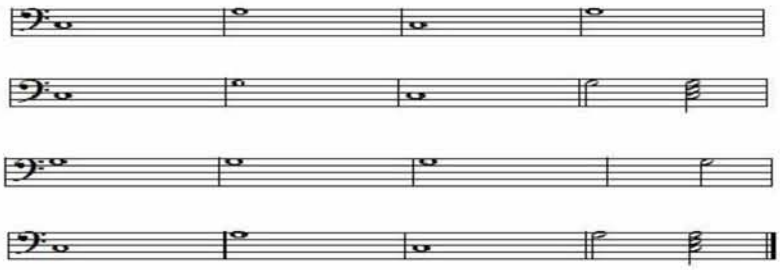
Και σε αυτό το νήμα αρχικά δηλώνονται οι αξίες και οι μεταβλητές των νοτών και κατόπιν ξεκινάει η μετατροπή την παρτιτούρας σε κώδικα. Στην συγκεκριμένη παρτιτούρα δεν επαναλαμβάνεται κάποιο μουσικό μοτίβο γι' αυτό χωρίστηκε σε τέσσερα μέρη. Κάθε μέρος εμπεριέχει τέσσερα μουσικά μέτρα. Αυτά τα μουσικά μέρη καταχωρούνται ξανά σε τέσσερις δισδιάστατους πίνακες τους mel_1,mel_2,mel_3 και mel_4. Κατά την διάρκεια της ενορχήστρωσης της μελωδίας για το βιμπράφωνο χρησιμοποιήθηκαν και ορισμένες αλλοιώσεις στις νότες (υφέσεις και διέσεις) όπως μπορεί να προσέξει κανείς στην παρτιτούρα. Για την μετατροπή αυτών των αλλοιωμένων νοτών στην chuck χρησιμοποιήθηκαν οι μεταβλητές που μετά το όνομα τους ακολουθεί το γράμμα **"s"**.

Η κλάση ModalBar της chuck παρέχει και αυτή κάποιες προεκτάσεις για την προσαρμογή του ήχου του κρουστού οργάνου. Αρχικά η **".preset"** καθορίζει το κρουστό όργανο που θα ακουστεί στην έξοδο του προγράμματος. Η επιλογή γίνεται ανάμεσα από οχτώ όργανα. Το βιμπράφωνο έχει την τιμή ένα. Κατά πόσο σκληρός ή μαλακός θα είναι ο ήχος του βιμπράφωνου το καθορίζουν οι επιλογές **".stickHardness"** και **".strikePosition"**. Η πρώτη εκφράζει την σκληρότητα της μπακέτας και η δεύτερη το σημείο που θα γίνει το χτύπημα στο κρουστό. Το βιμπράτο ρυθμίζεται από τις **".vibratoGain"** και **".vibratoFreq"**. Το πλάτος του σήματος στην έξοδο και τον θόρυβο που προκαλεί το ρυθμίζουν οι **".directGain"** και **".masterGain"** .Τέλος, η ένταση της παραγόμενης μουσικής μπορεί να ελεγχθεί από την **".volume"**. Αξίζει να σημειωθεί ότι η τιμή της **".volume"** στα σημεία που η παρτιτούρα έχει "*f*" παίρνει τιμή 0.7 και στα σημεία που έχει "*p*" παίρνει 0.5.

Από πνευστά όργανα αν και η chuck παρέχει αρκετές επιλογές επιλέχθηκαν το σαξόφωνο και το κλαρινέτο. Το σαξόφωνο επιλέχθηκε γιατί μπορεί να παίξει χαμηλές νότες στο κλειδί του φα και καλύψει τις πιο μπάσες νότες της ενορχήστρωσης. Ακολουθεί η μουσική παρτιτούρα για σαξόφωνο **εικόνα 8** :

ΩΔΗ ΣΤΗΝ ΧΑΡΑ- ΣΑΞΟΦΩΝΟ

Beethoven



Εικόνα 8

Το νήμα-Shred που παράγει μέσα από τον υπολογιστή την εκτέλεση της παραπάνω παρτιτούρας (**εικόνα 8**) με τον ήχο του σαξόφωνου και προγραμματίστηκε σε chuck είναι το εξής :

Shred «Ωδή στην χαρά» για σαξόφωνο.

```
Saxofony sax => JCRev r => dac;
.2 => r.gain;
.05 => r.mix;

120 => int tempo;

dur duration[9];
240000::ms / ( 1 * tempo )  => duration[1]; // whole
240000::ms / ( 2 * tempo )  => duration[2]; // half
240000::ms / ( 4 * tempo )  => duration[4]; // quarter (crotchet)
240000::ms / ( 8 * tempo )  => duration[8]; // eighth (quaver)
(duration[4] + duration[8]) => duration[5]; // dotted quarter
(duration[2] + duration[4]) => duration[3]; // dotted half


//notes
60 => int c;     72 => int C;
61 => int cs;    73 => int Cs;
62 => int d;     74 => int D;
63 => int ds;    75 => int Ds;
64 => int e;     76 => int E;
65 => int f;     77 => int F;
66 => int fs;    78 => int Fs;
67 => int g;     79 => int G;
68 => int gs;    80 => int Gs;
69 => int a;     81 => int A;
70 => int as;    82 => int As;
71 => int b;     83 => int B;

//melody in 3 parts
[ [c-12,1],[g-12,1],[c-12,1],[g-12,1]] @=> int mel_1[][];

[ [c-12,1],[g-12,1],[c-12,1],[g-12,2],[g-12,2]] @=> int mel_2[][];

[ [g-12,1],[g-12,1],[g-12,1],[g-12,1]] @=> int mel_3[][];

//for transporto
0 => int trans;

//function for playing the notes
fun void playNote( int n, int d ) {

    std.mtof(n + trans) => sax.freq;
```

```
    duration[d] => now;
     .6 => sax.noteOn;
}
for( 0 => int j; j<2; j++){

    for( 0 => int i; i < mel_1.cap(); i++) {

        //choice the way that will sound one note
        0.8 => sax.stiffness;
        0.65 => sax.aperture;
        0.01 => sax.noiseGain;
        0.7 => sax.blowPosition;
        6 => sax.vibratoFreq;
        0.04 => sax.vibratoGain;
        0.4 => sax.pressure;

        //play the note
        playNote(mel_1[i][0], mel_1[i][1]);
    }

    for( 0 => int i; i < mel_2.cap(); i++) {

        //choice the way that will sound one note
        0.8 => sax.stiffness;
        0.65 => sax.aperture;
        0.01 => sax.noiseGain;
        0.7 => sax.blowPosition;
        6 => sax.vibratoFreq;
        0.04 => sax.vibratoGain;
        0.4 => sax.pressure;

        //play the note
        playNote(mel_2[i][0], mel_2[i][1]);
    }

    for( 0 => int i; i < mel_3.cap(); i++) {

        //choice the way that will sound one note
        .1 => r.gain;
        0.8 => sax.stiffness;
        0.65 => sax.aperture;
        0.01 => sax.noiseGain;
        0.7 => sax.blowPosition;
        6 => sax.vibratoFreq;
        0.04 => sax.vibratoGain;
        0.4 => sax.pressure;

        //play the note
        playNote(mel_3[i][0], mel_3[i][1]);
    }
```

```
    for( 0 => int i; i < mel_2.cap(); i++) {

        //choice the way that will sound one note
        .2 => r.gain;
        0.8 => sax.stiffness;
        0.65 => sax.aperture;
        0.01 => sax.noiseGain;
        0.7 => sax.blowPosition;
        6 => sax.vibratoFreq;
        0.04 => sax.vibratoGain;
        0.4 => sax.pressure;

        //play the note
        playNote(mel_2[i][0], mel_2[i][1]);
    }
}

playNote( g-12, 1 );
.6 => sax.noteOff;
```

Η κλάση "Saxofony" εκφράζει ένα ψηφιακό όργανο με κυματοειδή μορφή εξόδου που μπορεί να παράγει μια πληθώρα πνευστών ήχων. Ας υποτεθεί ότι ο ήχος παράγεται από μια χορδή που ταλαντώνεται αναλόγως το φύσημα που δέχεται. Το φύσημα μπορεί να γίνει σε οποιοδήποτε σημείο ανάμεσα στα δύο άκρα της χορδής. Αν η δύναμη ασκηθεί στο μέσον της χορδής τότε ο ήχος μοιάζει πιο πολύ σε κλαρινέτο. Αν όμως ασκηθεί κοντά στο ένα άκρο ο ήχος μοιάζει με σαξόφωνο.

Αρχικά στον παραπάνω κώδικα ρυθμίζεται το πλάτος του κύματος εξόδου. Η μελωδία αυτή την φορά με βάση την παρτιτούρα είναι πιο απλή . Χωρίζεται σε τρία μέρη .Το καθένα έχει τέσσερα μουσικά μέτρα και τα τελευταία τέσσερα είναι όμοια με τα τέσσερα πρώτα. Άρα αρκούν τρείς δισδιάστατοι πίνακες για να καταχωρηθεί η μελωδία. Αυτοί είναι όπως διαφαίνεται και στον κώδικα οι mel_1, mel_2 και mel_3 .

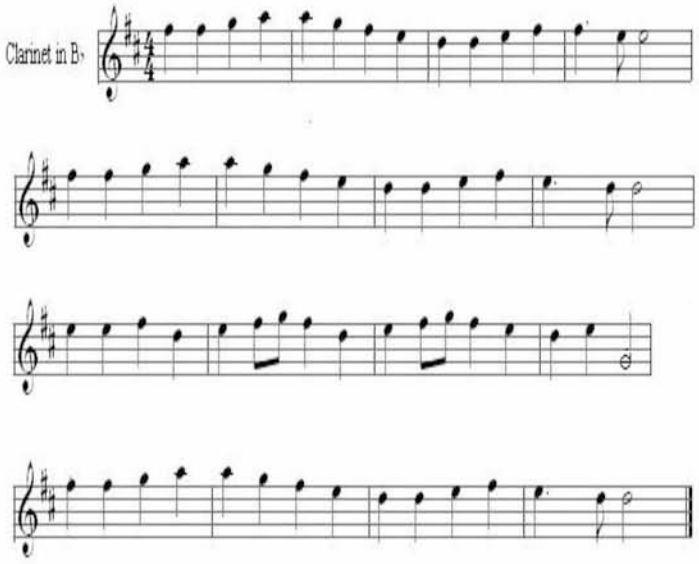
Τα εργαλεία που παρέχει η "Saxofony" βοηθούν ώστε να διαμορφωθεί ένας ήχος παρόμοιος με αυτόν του σαξοφώνου. Οι δύο πιο καθοριστικές προεκτάσεις είναι οι **".stiffness"** και **".aperture"**. Η πρώτη ρυθμίζει πόσο άκαμπτη είναι η παλλόμενη χορδή και η δεύτερη πόσος αέρας μπορεί να περάσει σε κάθε φύσημα. Γι' αυτό στην πρώτη η τιμή είναι κοντά στο άνω όριο που μπορεί να δεχθεί ώστε να μοιάζει με σαξόφωνο. Ειδικά αν συνδυαστεί με την μεγάλη τιμή της **".blowPosition"** .Με τις **".vibratoGain"** και **".vibratoFreq"** εκφράζεται το βιμπράτο του πνευστού. Τέλος η επέκταση **".pressure"** αυξομειώνει την ένταση.

Το δεύτερο πνευστό όργανο που χρησιμοποιήθηκε και με αυτό ολοκληρώνεται η ενορχήστρωση της συγκεκριμένης μουσικής σύνθεσης είναι το κλαρινέτο. Αυτό το πνευστό μπορεί να παίξει αρκετά ψηλές νότες. Η παρακάτω παρτιτούρα που εμπεριέχει την μελωδία του κλαρινέτου στην «ωδή στην χαρά» είναι γραμμένη σε άλλη τονικότητα από την προηγούμενη. Το κλαρινέτο είναι σε σι ύφεση. Σε σχέση με τα υπόλοιπα όργανα ,το

κλαρινέτο όταν παίξει μία νότα θα ακουστεί έναν τόνο κάτω . Δηλαδή η βασική μελωδία ξεκινάει από την νότα μι. Το κλαρινέτο για να παίξει αυτή τη νότα θα πρέπει στην παρτιτούρα η πρώτη νότα να είναι φα δίεση. Άρα από την Ντο μείζονα που είναι η τονικότητα της μελωδίας για το κλαρινέτο η τονικότητα θα γίνει Ρε μείζονα.

Ακολουθεί η μουσική παρτιτούρα για κλαρινέτο **εικόνα 9** που μεταφέρθηκε σε κώδικα chuck :

ΩΔΗ ΣΤΗΝ ΧΑΡΑ- ΚΛΑΡΙΝΕΤΟ
Beethoven



Εικόνα 9

Το νήμα-Shred που παράγει μέσα από τον υπολογιστή την εκτέλεση της παραπάνω παρτιτούρας (**εικόνα 9**) με τον ήχο του κλαρινέτου και προγραμματίστηκε σε chuck είναι το εξής:

Shred «Ωδή στην χαρά» για κλαρινέτο.

```
Clarinet clair => JCRev r => dac;
.75 => r.gain;
.1 => r.mix;

120 => int tempo;

dur duration[9];
240000::ms / ( 1 * tempo )  => duration[1]; // whole
```

```
240000::ms / ( 2 * tempo )  => duration[2]; // half
240000::ms / ( 4 * tempo )  => duration[4]; // quarter (crotchet)
240000::ms / ( 8 * tempo )  => duration[8]; // eighth (quaver)
(duration[4] + duration[8]) => duration[5]; // dotted quarter
(duration[2] + duration[4]) => duration[3]; // dotted half

//notes
60 => int c;     72 => int C;
61 => int cs;    73 => int Cs;
62 => int d;     74 => int D;
63 => int ds;    75 => int Ds;
64 => int e;     76 => int E;
65 => int f;     77 => int F;
66 => int fs;    78 => int Fs;
67 => int g;     79 => int G;
68 => int gs;    80 => int Gs;
69 => int a;     81 => int A;
70 => int as;    82 => int As;
71 => int b;     83 => int B;

//melody in 3 parts
[ [E,4],[E,4],[F,4],[G,4],
 [G,4],[F,4],[E,4],[D,4],
 [C,4],[C,4],[D,4],[E,4],
 [E,5],[D,8],[D,2] ] @=> int mel_1[][];

[ [E,4],[E,4],[F,4],[G,4],
  [G,4],[F,4],[E,4],[D,4],
  [C,4],[C,4],[D,4],[E,4],
  [D,5],[C,8],[C,2] ] @=> int mel_2[][];

[ [D,4],[D,4],[E,4],[C,4],
  [D,4],[E,8],[F,8],[E,4],
  [C,4],[D,4],[E,8],[F,8],
  [E,4],[D,4],[C,4],[D,4],[G-12,2]] @=> int mel_3[][];

//for transporto
0 => int trans;

//function for playing the notes
fun void playNote( int n, int d ) {

    std.mtof(n + trans) => clair.freq;
    duration[d] => now;
     .6 => clair.noteOn;
}
for( 0 => int j; j<2; j++){

    for( 0 => int i; i < mel_1.cap(); i++) {
        //choice the way that will sound one note
```

```
        0.04 => clair.reed;
        0.1 => clair.noiseGain;
        6 => clair.vibratoFreq;
        0.01 => clair.vibratoGain;
        0.4 => clair.pressure;
        //play the note
        playNote(mel_1[i][0], mel_1[i][1]);
    }

    for( 0 => int i; i < mel_2.cap(); i++) {
        //choice the way that will sound one note
        0.04 => clair.reed;
        0.1 => clair.noiseGain;
        5 => clair.vibratoFreq;
        0.01 => clair.vibratoGain;
        0.4 => clair.pressure;
        //play the note
        playNote(mel_2[i][0], mel_2[i][1]);
    }

    for( 0 => int i; i < mel_3.cap(); i++) {
        //choice the way that will sound one note
        0.04 => clair.reed;
        0.1 => clair.noiseGain;
        5 => clair.vibratoFreq;
        0.1 => clair.vibratoGain;
        0.395 => clair.pressure;
        //play the note
        playNote(mel_3[i][0], mel_3[i][1]);
    }



    for( 0 => int i; i < mel_2.cap(); i++) {
        //choice the way that will sound one note
        0.04 => clair.reed;
        0.1 => clair.noiseGain;
        5 => clair.vibratoFreq;
        0.01 => clair.vibratoGain;
        0.4 => clair.pressure;
        //play the note
        playNote(mel_2[i][0], mel_2[i][1]);
    }
}

0.43 => clair.pressure;
playNote( C,1);
.6 => clair.noteOff;
```

Η κλάση “Clarinet” ανήκει και αυτή στην υπέρ-κλάση STK instruments της chuck.

Προσπαθεί να προσομοιώσει μέσα από τον υπολογιστή τον ήχο του κλαρινέτου. Η μελωδία έχει τα ίδια χαρακτηριστικά με του βιολιού και χωρίζεται με τον ίδιο τρόπο σε τέσσερα μέρη. Για την μεταφορά της στην chuck χρησιμοποιήθηκαν τρείς δισδιάστατοι πίνακες οι mel_1, mel_2 και mel_3.

Ο ήχος του κλαρινέτου βασίζεται στο τρόπο που φυσάει ο μουσικός τον αέρα στο στόμιο του οργάνου. Γι' αυτό το λόγο η κλάση παρέχει την επέκταση **".reed"** η οποία προσομοιώνει αυτό το μέλος του πνευστού. Όσο μεγαλύτερη είναι η τιμή που παίρνει ,τόσο πιο ασθενές είναι ο ήχος που παράγεται. Στον παραπάνω κώδικα προτιμήθηκαν χαμηλές τιμές για βέλτιστο ηχητικό αποτέλεσμα. Για την ρύθμιση του πλάτους και του θορύβου υπάρχει η **".noiseGain"**. Το βιμπράτο καθορίζεται πάλι από τις **".vibratoFreq"** και **".vibratoGain"** . Τέλος η ένταση του παραγόμενου ήχου οφείλεται στην προέκταση **".pressure"** . Στα σημεία που η δυναμική του κομματιού απαιτεί *forte* η τιμή της pressure είναι 0.4 , ενώ όταν απαιτείται *piano* 0.395.Αριθμητικά η διαφορά είναι πολύ μικρή αλλά η ένταση του παραγόμενου ήχου διαφέρει αρκετά.

4.2.3 Συγχρονισμός Shreds για την δημιουργία μουσικής ορχήστρας .

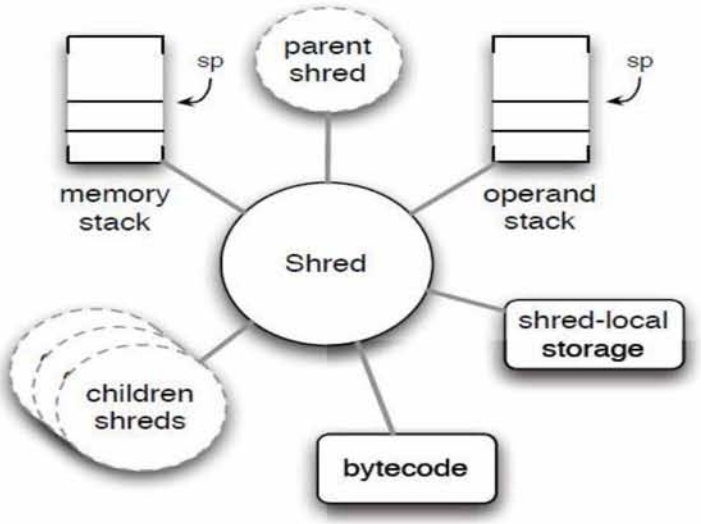
Η μεταγλώττιση[1] ενός προγράμματος chuck ακολουθεί τις καθιερωμένες φάσεις λεκτικής και συντακτικής ανάλυσης, έλεγχος των τύπων κτλ. Τα προγράμματα τρέχουν μέσα από τις εντολές μιας εικονικής μηχανής ,είτε ως ένα shred ,είτε ως αντικείμενα ή ρουτίνες. Ο μεταγλωττιστής τρέχει στην ίδια διαδικασία με την εικονική μηχανή και μπορεί να μεταγλωττίσει απ' ευθείας νέα προγράμματα.

Μετά την μεταγλώττιση, ένα shred περνάει απ' ευθείας στην εικονική μηχανή. Εκεί δρομολογείται ώστε να ξεκινήσει αμέσως την εκτέλεση. Κάθε shred περιέχει αρκετές πληροφορίες όπως (**εικόνα 10**):

1. Οδηγίες που προέρχονται από τον bytecode.
2. Μια στοίβα για τοπικούς και προσωρινούς υπολογισμούς.
3. Μια στοίβα για την αποθήκευση τοπικών μεταβλητών για τις κλήσεις συστήματος.
4. Αναφορές σε προγόνους και απογόνους του shred.
5. Πληροφορία για τον χρόνο και το «τώρα».

Η κατάσταση ενός shred καθορίζεται αποκλειστικά από το περιεχόμενο των στοιβών και των δεικτών τους. Ένα shred μπορεί να τερματιστεί με εντολές κατά την πάροδο του χρόνου. Επίσης τα ίδια τα shreds μπορούν να δημιουργήσουν και να απομακρύνουν άλλα shreds.

Ο δρομολογητής σειριοποιεί την εκτέλεση των shreds με την μηχανή ήχου, σεβόμενος την έννοια και την αξία που έχει για το σύστημα η λέξη κλειδί **"now"**. Η τιμή της **"now"** θα αντιστοιχηθεί στα δείγματα της τελικής σύνθεσης που έχουν εκτελεστεί από την αρχή του προγράμματος.



Εικόνα 10

Οι εντολές του bytecode στην εικονική μηχανή χειρίζονται την στοίβα των τελεστών και τη στοίβα της μνήμης του shred ,την αναφορά στο ίδιο το shred και την εικονική μηχανή που τρέχει το shred. Κάθε εντολή έχει ξεκάθαρη συμπεριφορά ως προς τις στοίβες, τα shreds και την εικονική μηχανή. Οι εντολές μπορεί να είναι απλά αριθμητικές ή και πιο σύνθετες. Εξαιτίας των ιδιοτήτων της chuck , οι εντολές μπορούν να λειτουργήσουν και χωρίς έλεγχο τύπου run-time.

Οι unit generations ,οι οποίοι χρησιμοποιήθηκαν στα shreds της μουσικής σύνθεσης για τον ήχο των μουσικών οργάνων, δημιουργούνται δυναμικά, συνδέονται ή αποσυνδέονται και ελέγχονται από τα shreds. Παρ' όλα αυτά ο πραγματικός υπολογισμός του ήχου γίνεται ξεχωριστά. Όταν ο δρομολογητής την shreds αποφασίζει ότι είναι σκόπιμο να υπολογιστεί το επόμενο δείγμα στην πάροδο του χρόνου, η μηχανή παραγωγής ήχου καλείται.

Αρχικά εκτελούνται οι unit generators που συνδέονται με τις πιο συνηθισμένες λειτουργίες όπως είναι η dac. Κάθε unit generator που συνδέεται άμεσα ή έμμεσα καλείται να υπολογίσει και να επιστρέψει το επόμενο δείγμα του. Το σύστημα μαρκάρει τους κόμβους που έχει επισκεφτεί ώστε κάθε unit generator να καλείται ακριβώς μια φορά για κάθε δείγμα. Η τιμή της εξόδου κάθε unit generator αποθηκεύεται και μπορεί να επανακτηθεί .

Μετά την αναλυτική περιγραφή της μεταγλώττισης και εκτέλεσης ενός shred, ο συγχρονισμός και η εκτέλεση των shreds της μουσικής σύνθεσης του Beethoven που παρουσιάστηκε στην προηγούμενη ενότητα, μπορεί να γίνει με τον εξής τρόπο. Τα shreds που κατασκευάστηκαν στην chuck «αναπαριστούν» τα όργανα που εκτελούν την μουσική σύνθεση. Για την εκτέλεση των shreds θα χρειαστούν οι ονομασίες των αρχείων τύπου chuck που αποθηκεύτηκαν. Το shred με unit generation το βιολί έχει την ονομασία Beethoven_violin.ck ,με το βιμπράφωνο Beethoven_vib.ck , με το σαξόφωνο Beethoven_sax.ck και με το κλαρινέτο Beethoven_clar.ck .Η απλότητα της chuck δίνει την δυνατότητα στον χρήστη να εκτελέσει ταυτόχρονα και τα τέσσερα αυτά νήματα και ο συγχρονισμός τους να γίνει με τον τρόπο που περιγράφτηκε από την ίδια την εικονική

μηχανή. Αρκεί λοιπόν σε ένα τερματικό ο χρήστης να δώσει την εντολή:

```
%> chuck  Beethoven_violin.ck  Beethoven_vib.ck  Beethoven_sax.ck  Beethoven_clar.ck
```

Με αυτή την διαδικασία ο υπολογιστής παράγει τελικά μία πολυφωνική μελωδία. Αν επιθυμεί ο χρήστης έχει την δυνατότητα να μειώσει τα shreds ή να προσθέσει περισσότερα. Πληκτρολογώντας την εντολή **"%> chuck +..."** μπορεί να προσθέσει ακολούθως τα shreds που θέλει να συγχρονιστούν μαζί και με την "**"%> chuck -**" να αφαιρέσει κάποιο. Με την εντολή **"-kill"** μπορεί να τερματίσει τα νήματα που τρέχουν. Ένας άλλος τρόπος εκτέλεσης των shreds είναι με την **Machine.add("μονοπάτι shred")**. Αυτή η μέθοδος είναι χρήσιμη για την εκτέλεση προγραμμάτων με δυναμικό τρόπο. Ένα παράδειγμα προσθήκης και αφαίρεσης shreds:

```
//add Beethoven_violin.ck
   Machine.add("Beethoven_violin.ck") => int id;
//remove shred id
  Machine.remove(id);
//replace shreds
  Machine.add("Beethoven_sax.ck") => int id;
  Machine.replace(id, "Beethoven_vib.ck") ;
```

4.2.4. Αλληλεπίδραση του ηλεκτρικού βιολιού με τον υπολογιστή για την ολοκλήρωση της μουσικής σύνθεσης.

Για την πραγματοποίηση της μουσικής σύνθεσης σε γλώσσα προγραμματισμού chuck, επιλέχθηκε η «ωδή στην χαρά» .Το κομμάτι αυτό ανήκει στο ρεπερτόριο της κλασικής μουσικής. Το βιολί είναι ένα μουσικό όργανο το οποίο χρησιμοποιείται κατά κόρων σε αυτό το είδος μουσικής. Με την βοήθεια της τεχνολογίας έχει πλέον βελτιωθεί ο ήχος του ηλεκτρικού βιολιού και προσεγγίζει αυτόν του φυσικού οργάνου. Το πλεονέκτημα του ηλεκτρικού βιολιού είναι ότι μπορεί να συνδεθεί και με τον ηλεκτρονικό υπολογιστή.

Ο συγχρονισμός και η εκτέλεση των shreds που παρουσιάστηκε στην προηγούμενη ενότητα θα δημιουργήσουν μια πολυφωνική μελωδία στην έξοδο ήχου του υπολογιστή. Αυτή θα αποτελέσει την ορχήστρα με την οποία μπορεί να αλληλεπιδράσει ο χρήστης χρησιμοποιώντας το ηλεκτρικό βιολί. Ακολουθεί η περιγραφή της διαδικασίας για την σύνδεση του ηλεκτρικού βιολιού στον υπολογιστή.

Το ηλεκτρικό βιολί μπορεί με την χρήση των unit generators να εισάγει το σήμα του στο πρόγραμμα chuck. Ο ugen adc λαμβάνει την είσοδο του αναλογικού αυτού σήματος .Αντίστοιχα ο dac εξάγει το σήμα στην έξοδο. Η εντολή για την σύνδεση βιολιού με το πρόγραμμα είναι :

```
adc=>dac
```

Ωστόσο για την βελτίωση της ποιότητας του ήχου του ηλεκτρικού βιολιού χρησιμοποιήθηκε επιπλέον εξωτερική κάρτα ήχου usb (delock usb sound adapter 7.1) . Στις **εικόνες 11 και 12** φαίνονται η είσοδος και η έξοδος της κάρτας καθώς και τα υπόλοιπα χαρακτηριστικά της.

Εικόνα 11

Εικόνα 12

Για να λειτουργήσει η κάρτα ήχου στο miniAudicle ,το οποίο είναι το περιβάλλον που χρησιμοποιείται για την δημιουργία προγραμμάτων σε chuck, πρέπει να γίνουν ορισμένες ρυθμίσεις. Πηγαίνοντας στο μενού edit και μετά preferences εμφανίζεται ένα νέο παράθυρο. Εκεί στα πεδία audio input και audio output επιλέγεται το όνομα της usb κάρτας ήχου.

Στην συνέχεια μπορεί γίνεται η σύνδεση του ηλεκτρικού βιολιού με την κάρτα ήχου. Το βιολί θα συνδεθεί στην είσοδο μικροφώνου της κάρτας ήχου. Ωστόσο χρειάζεται ένας μετατροπέας. Η είσοδος της κάρτας ήχου είναι 3.5mm και το βύσμα του καλωδίου που μεταφέρει το σήμα του ηλεκτρικού βιολιού 6.3mm. Στην **εικόνα 13** φαίνεται αυτή η σύνδεση.

Εικόνα 13

Μετά την σύνδεση του ηλεκτρικού βιολιού στον υπολογιστή μπορεί να εκτελεστεί η μουσική σύνθεση. Ο βιολιστής-χρήστης εκτελώντας την παρτιτούρα για βιολί του κομματιού «ωδή στην χαρά» **(εικόνα 6)** έχει την δυνατότητα να παίξει μαζί με μια ορχήστρα αυτή την μουσική σύνθεση. Για να ολοκληρωθεί αυτή η διαδικασία χρησιμοποιείται ακόμα ένα shred που περιέχει μόνο την εντολή "**adc=>dac**" και έχει το όνομα **"electric_violin.ck"**. Εκτελώντας την παρακάτω εντολή, η οποία εμπεριέχει τα shreds με τα μουσικά όργανα, η μουσική σύνθεση σε chuck με την αλληλεπίδραση του ηλεκτρικού βιολιού αρχίζει να αναπαράγεται :

```
%> chuck Beethoven_violin.ck Beethoven_vib.ck Beethoven_sax.ck Beethoven_clar.ck
electic_violin.ck
```

Ο χρήστης καλείται να ερμηνεύσει το κομμάτι «ωδή στην χαρά» του Beethoven με το βιολί και να γευτεί την πεμπτουσία της κλασσικής μουσικής και την ικανοποίηση της μουσικής σύνθεσης που προσφέρει η γλώσσα προγραμματισμού chuck.

4.2.5. Ηχογράφηση της μουσικής σύνθεσης.

Η chuck παρέχει την δυνατότητα ηχογράφησης της μουσικής σύνθεσης. Μετά την εγκατάσταση της chuck στον υπολογιστή ,δημιουργείται ένας φάκελος με ονομασία

“Chuck” .Στο μονοπάτι **examples/basic/** του φακέλου βρίσκονται τα αρχεία **“rec.ck”** και **“rec-auto.ck”** .Η διαδικασία μπορεί να ξεκινήσει, αφού γίνει η μεταφορά των αρχείων στον φάκελο με τα shreds προς ηχογράφηση. Ακολουθεί ο κώδικάς που έχουν κατασκευάσει οι δημιουργοί της chuck για την ηχογράφηση.

Shred rec.ck για ηχογράφηση.

```
me.arg(0) => string filename;
if( filename.length() == 0 ) "foo.wav" => filename;

// pull samples from the dac
dac => Gain g => WvOut w => blackhole;
// this is the output file name
filename => w.wavFilename;
<<<"writing to file:", "'" + w.filename() + "'">>>;
// any gain you want for the output
.5 => g.gain;

// temporary workaround to automatically close file on remove-shred
null @=> w;

// infinite time loop...
// ctrl-c will stop it, or modify to desired duration
while( true ) 1::second => now;
```

Για την πραγματοποίηση της ηχογράφησης δεν απαιτούνται αλλαγές στον παραπάνω κώδικα. Με την παρακάτω εντολή η μουσική σύνθεση που κατασκευάστηκε σε αυτή την εργασία μπορεί να αποθηκευτεί στον σκληρό δίσκο του υπολογιστή με την μορφή **“.wav”** .

```
%> chuck Beethoven_violin.ck Beethoven_vib.ck Beethoven_sax.ck Beethoven_clar.ck
electic_violin.ck rec.ck
```

Μετά την εκτέλεση της εντολής δημιουργείται ένα αρχείο με την ονομασία **“foo.wav”** το οποίο μετά μπορεί να μετονομαστεί. Με το shred **rec-auto.ck** αποφεύγονται εγγραφές με νέες ηχογραφήσεις στο ίδιο αρχείο. Αυτό το shred αποθηκεύει το όνομα του αρχείου με βάση την ώρα.

Τέλος ,το **rec.ck** μπορεί να εκτελεστεί και “on-the-fly” .

Τερματικό 1

```
%>chuck -loop
```

Τερματικό 2

```
%>chuck + rec.ck
```

Βιβλιογραφία.

[1] Wang, G., P. R. Cook, and A. Misra. 2005. "*Designing and Implementing the ChucK Programming Language*" *In Proceedings of the International Computer Music Conference (ICMC)* Barcelona.

[2] Wang G., and P. R. Cook. 2004. "*On-the-fly Programming: Using Code as an Expressive Musical Instrument.*" *In Proceedings of the 2004 International Conference on New Interfaces for Musical Expression (NIME)* , Hamamatsu, Japan, June 2004.

[3] Wang, G., R. Fiebrink, and P. Cook. 2007. "*Combining Analysis and Synthesis in the ChucK Programming Language*" *In Proceedings of the International Computer Music Conference (ICMC)*, Copenhagen.

[4] https://en.wikipedia.org/wiki/Sound_and_music_computing

[5] Ge Wang. "*The ChucK Audio Programming Language: A Strongly-timed and On-the-fly Environ/mentality.*"
PhD Thesis, Princeton University, 2008.

[6] https://en.wikipedia.org/wiki/Electric_violin

[7] Richard Moore ."*Real time interactive computer music synthesis"*. September 1977

[8] https://en.wikipedia.org/wiki/Ode_to_Joy

[9] http://wiki.cs.princeton.edu/index.php/ChucKing_Scores