ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«ΠΛΗΡΟΦΟΡΙΚΗ»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**ΤΟ ΚΑΝΟΝΑΚΙ ΜΕΣΑ ΑΠΟ ΤΗΝ ΤΕΧΝΟΛΟΓΙΑ VST** **Από το φυσικό στον ψηφιακό ήχο** **THE KANUN THROUGH VST TECHNOLOGY** **From Acoustic to Digital Sound**
Ονοματεπώνυμο Φοιτητή	**ΜΠΙΝΙΑΡΗ -ΠΑΝΤΕΛΕΩΝ ΔΑΦΝΗ**
Πατρώνυμο	**ΓΕΩΡΓΙΟΣ**
Αριθμός Μητρώου	**ΜΠΠΛ21052**
Επιβλέπων	**Τσιχριντζής Γεώργιος, Καθηγητής**

Ημερομηνία Παράδοσης **Ιούνιος, 2024**

Τριμελής Εξεταστική Επιτροπή

Τσιχριντζής Γεώργιος Καθηγητής	Ευάγγελος Σακκόπουλος Αναπληρωτής Καθηγητής	Διονύσιος Σωτηρόπουλος, Επίκουρος Καθηγητής

ΠΕΡΙΛΗΨΗ

Η παρούσα εργασία ασχολείται με τη δημιουργία ενός VST οργάνου χρησιμοποιώντας φυσικό ήχο από κανονάκι, το οποίο ηχογραφήθηκε σε στούντιο. Το όργανο ηχογραφήθηκε σε όλη του την έκταση (όλες οι οκτάβες) ανά ημιτόνιο. Στην συνέχεια επιλέχθηκαν και χρησιμοποιήθηκαν τα καλύτερα ηχητικά δείγματα. Για τις ανάγκες της υλοποίησης του συγκεκριμένου VST χρησιμοποιήθηκε το framework JUCE, μέσω του Visual Studio. Η γλώσσα προγραμματισμού που επιλέχθηκε είναι η C++. Σκοπός της παρούσας εργασίας είναι αφενός, να γίνει μια εισαγωγή στην μουσική πληροφορική καθώς και μια προσπάθεια "'ένωσης" της επιστήμης της μουσικής με την επιστήμη του προγραμματισμού.

ABSTRACT

This project focuses on the creation of a VST instrument using natural sound from a kanun, which was recorded in a studio. The instrument was recorded in its entirety (all octaves) at every semitone. Subsequently, the best sound samples were selected and used. For the implementation of this specific VST, the JUCE framework was utilized via Visual Studio. The programming language chosen was C++. The aim of this project is, on the one hand, to provide an introduction to music informatics and, on the other hand, to attempt a "union" of the science of music with the science of programming.

Περιεχόμενα

ΕΙΣΑΓΩΓΗ

Η μελέτη του ήχου και της μουσικής αποτελεί ένα πολύπλευρο πεδίο που συνδυάζει τη φυσική, την τεχνολογία, και την μουσική. Στο παρόν κείμενο, και με αφορμή το μουσικό όργανο κανονάκι, εξετάζονται τα βασικά στοιχεία του ήχου, η φύση του και τα χαρακτηριστικά του, καθώς και η διαμόρφωση των μουσικών διαστημάτων. Επιπλέον, αναλύεται η εξέλιξη της μουσικής τεχνολογίας, από τον αναλογικό ήχο στον ψηφιακό, και πώς οι σύγχρονες τεχνολογίες βρίσκουν εφαρμογή σε διάφορους επιστημονικούς κλάδους.

Ξεκινώντας από τον ορισμό και τα χαρακτηριστικά του ήχου, γίνεται μια συσχέτιση μεταξύ συχνότητας, έντασης, χροιάς και καθαρότητας, και πώς αυτές οι παράμετροι συμβάλλουν στη μουσική αντίληψη. Στη συνέχεια, επιχειρείται μια μετάβαση στους μουσικούς τόνους και στη συγκερασμένη κλίμακα της Δυτικής μουσικής, αναλύοντας τους φθόγγους και τον τρόπο με τον οποίο δημιουργούνται και χρησιμοποιούνται στην μουσική.

Έπειτα, περιγράφεται η μουσική τεχνολογία και η εξέλιξή της, από τις πρώτες προσπάθειες καταγραφής του ήχου με τον φωνόγραφο του Thomas Edison, μέχρι τις σύγχρονες μεθόδους ψηφιακής επεξεργασίας και παραγωγής μουσικής. Εξετάζεται η διαδικασία της δειγματοληψίας, η κβαντοποίηση, και η μετατροπή του αναλογικού σήματος σε ψηφιακό, καθώς και η σημασία του βάθους δειγματοληψίας και των σφαλμάτων κβαντοποίησης.

Επιπλέον, παρουσιάζονται, οι εφαρμογές της μουσικής πληροφορικής, οι οποίες καλύπτουν ευρύ φάσμα δραστηριοτήτων, από τη σύνθεση και την παραγωγή μουσικής έως την ανάλυση και την εκπαίδευση. Αναλύεται ο τρόπος με τον οποίο, η τεχνολογία VST (Virtual Studio Technology) επιτρέπει στους μουσικούς να επεκτείνουν τις δυνατότητες των ψηφιακών στούντιο, προσφέροντας εικονικά όργανα και εφέ υψηλής ποιότητας.

Τέλος, περιγράφεται η υλοποίηση ενός VST μέσω του προγραμματιστικού περιβάλλοντος Visual Studio και της γλώσσας C++, αναλύοντας τα βασικά αρχεία και τις λειτουργίες που απαιτούνται για τη δημιουργία ενός τέτοιου συστήματος. Με αυτόν τον τρόπο, αποκτάται μια ολοκληρωμένη εικόνα του πώς η τεχνολογία και η μουσική μπορούν να συνδυαστούν για να δημιουργήσουν νέες και καινοτόμες εμπειρίες.

Η παρούσα μελέτη αποσκοπεί στο να παρέχει μια ολοκληρωμένη κατανόηση των θεμελιωδών εννοιών του ήχου και της μουσικής, καθώς και της τεχνολογίας που υποστηρίζει την παραγωγή και την ανάλυσή τους, προσφέροντας παράλληλα πρακτικά παραδείγματα και εφαρμογές για την κατανόηση της σύγχρονης μουσικής τεχνολογίας.

ΚΕΦΑΛΑΙΟ 1

1.1 Ο ΗΧΟΣ

Ήχος είναι οτιδήποτε μπορούμε να ακούσουμε. Παράγεται από μια πηγή και συλλαμβάνεται από το αυτί μας[1]. Ο ήχος διαδίδεται είτε μέσω του αέρα με την μορφή ηχητικών κυμάτων, είτε μέσω των στερεών σωμάτων, είτε μέσω του νερού.

Η εξίσωση του ηχητικού κύματος για έναν απλό ήχο που αντιλαμβανόμαστε από κάποια απόσταση από την πηγή εκφράζεται ως εξής:

$P = p0 \sin 2\pi (ft - x/\lambda)$

Όπου:

- p0 = *το πλάτος της πίεσης στην πηγή*
- x = *η απόσταση από την πηγή*
- λ = *το μήκος κύματος*
- *f = η συχνότητα ταλάντωσης της πηγής*
- *t = ο χρόνος παρατήρησης*

Οι ήχοι χωρίζονται σε δύο μεγάλες κατηγορίες: Σε αυτούς που αντιστοιχούν σε ένα ορισμένο τονικό ύψος και ονομάζονται φθόγγοι και αυτοί που δεν έχουν συγκεκριμένο τονικό ύψος και ονομάζονται κρότοι ή θόρυβοι.

Όταν ένας ήχος έχει κίνηση περιοδική τότε έχει ορισμένο τονικό ύψος και ημιτονικό σήμα. Ανάλογα με την συχνότητά του μπορούμε να τον κατατάξουμε σε οξύ ή μπάσο ήχο. Όσο μεγαλύτερη η συχνότητα τόσο πιο οξύς είναι ο ήχος. Επομένως, η οξύτητα ενός φθόγγου είναι ανάλογη με την συχνότητα. Την συχνότητα την μετράμε σε Hertz (Hz) και στο κούρδισμα των μουσικών οργάνων συνήθως χρησιμοποιούμε τα 440 Hz.

Κάθε όργανο είναι ένα σύστημα, το οποίο παράγει ήχους[2]. Τα μουσικά όργανα βέβαια διαφέρουν μεταξύ τους, λόγω των διαφορετικών αρμονικών τις οποίες παράγουν. Επίσης, βασικό στοιχείο, το οποίο επηρεάζει την κύμανση του ήχου είναι η ένταση. Μονάδα μέτρησης της έντασης είναι τα decibel (dB). Ο ασθενέστερος ήχος που μπορεί να συλλάβει το αυτί του ανθρώπου είναι 1dB.

Ένα τρίτο χαρακτηριστικό του ήχου είναι η χροιά του ή αλλιώς το ηχόχρωμά του, το οποίο επίσης καθορίζεται από τους αρμονικούς φθόγγους τους οποίους παράγει. Μας βοηθάει να ξεχωρίσουμε τα όργανα μεταξύ τους. Το ηχόχρωμα κυρίως εξαρτάται από την σύνθεση των παλμικών κινήσεων[3].

Τέλος, ένα άλλο χαρακτηριστικό του ήχου είναι η καθαρότητα, η οποία εξαρτάται από την περιοδικότητα των παλμικών κινήσεων. Κανονικές περιοδικές παλμικές κινήσεις, παράγουν ήχους συγκεκριμένου τονικού ύψους, δηλαδή φθόγγους (βιολί, κιθάρα κτλ.). Ακανόνιστες παλμικές κινήσεις παράγουν θορύβους καθώς δεν έχουν συγκεκριμένο τονικό ύψος (π.χ. ταμπούρλο, κύμβαλα κτλ.), τα οποία αποδίδουν ρυθμούς.

Επομένως, τα χαρακτηριστικά του ήχου είναι η συχνότητα, η ένταση, η χροιά και η καθαρότητα. Το ανθρώπινο αυτί αντιλαμβάνεται ήχους από 20 Hz έως 20 KHz. Βέβαια διαφέρει η ακοή από άνθρωπο σε άνθρωπο, ανάλογα με την ηλικία του καθώς και τους θορύβους, οι οποίοι βομβαρδίζουν καθημερινά τα αυτιά του. Οι πιο χρήσιμες συχνότητες βρίσκονται κάτω από τα 10 kHz.

[1] Βλ. Διονύσιος Πολίτης, Μουσική Πληροφορική, εκδόσεις Κλειδάριθμος, Αθήνα 2007, σελ. 22.
[2] Βλ. Χαράλαμπος Χ. Σπυρίδης, *Η Φυσική των Μουσικών Οργάνων*, Καθηγητής Πανεπιστημίου Αθηνών, εκδόσεις Grapholine, Θεσσαλονίκη, σελ. 1.
[3] Βλ. Έφη Αβέρωφ, Εισαγωγή στην Οργανογνωσία, εκδόσεις Μουσικός Οίκος Φίλιππος Νάκας, Πέμπτη έκδοση, Αθήνα 1992, σελ. 15.

1.2 ΑΠΟ ΤΟΥΣ ΦΘΟΓΓΟΥΣ ΣΤΟΥΣ ΜΟΥΣΙΚΟΥΣ ΤΟΝΟΥΣ

Όπως προαναφέρθηκε, η συχνότητα ή αλλιώς το τονικό ύψος είναι ένα από τα χαρακτηριστικά του ήχου. Μουσικό τόνο ονομάζουμε την συχνότητα του ήχου. Στην συγκερασμένη κλίμακα της Δυτικής μουσικής υπάρχουν 120 διακριτοί τόνοι, από 16,351 Hz (νότα C0) έως 16,744 Hz (νότα C10) που μπορεί να αντιληφθεί το ανθρώπινο αυτί[4].

Οι φθόγγοι, οι οποίοι χρησιμοποιούνται στο Δυτικό σύστημα σημειογραφίας είναι επτά : *Ντο, Ρε, Μι, Φα, Σολ, Λα, Σι* ή αλλιώς *C, D, E, F, G, A, B*. Σε κάθε νότα αν διπλασιάσουμε την συχνότητά της βρίσκουμε την ίδια νότα μία οκτάβα υψηλότερα. Στα 440 Hz που αναφέρθηκε παραπάνω ότι κουρδίζουν τα μουσικά όργανα, βρίσκεται η νότα Λα4. Ένας τόνος με συχνότητα 220 Hz θα βρίσκεται μια οκτάβα κάτω (Λα3), ενώ ένας τόνος με συχνότητα 880 Hz θα βρίσκεται μια οκτάβα υψηλότερα (Λα5).

1.2.1 ΤΟ ΚΑΝΟΝΑΚΙ

Όπως συμβαίνει σχεδόν με όλα τα μουσικά όργανα, έτσι και το κανονάκι δεν έχει αυστηρά γεωγραφικά όρια. Απαντάται στην Εγγύς και την Μέση Ανατολή και στη βόρεια Αφρική αρκετούς αιώνες πριν. Στους βυζαντινούς και μεταβυζαντινούς χρόνους, οι τοιχογραφίες των εκκλησιών της Πόλης[5] είναι πλούσιες σε αναπαραστάσεις του «ψαλτηρίου». Όμως για μεγάλο διάστημα χάνονται τα ίχνη του, και εμφανίζεται ξανά στα τέλη του 19ου αιώνα στην Κωνσταντινούπολη. Είναι ένα όργανο το οποίο χρησιμοποιήθηκε και στην γραπτή και στην προφορική παράδοση[6]. Στην ευρύτερη περιοχή της Μεσογείου, η οποία μέσα στους αιώνες λειτουργεί ως πολιτισμικό χωνευτήρι και σε συνδυασμό με την έλευση των προσφύγων το 1922 σηματοδοτείται το ξεκίνημα μιας νέας εποχής, αφού οι άνθρωποι αυτοί μετέφεραν έναν ανεκτίμητο μουσικό πολιτισμό, μέρος του οποίου είναι και το κανονάκι[7].

Είναι ένα όργανο έγχορδο, νυκτό, ξύλινο που αποτελείται από χορδές, οι οποίες είναι φτιαγμένες από νάυλον. Ο αριθμός των χορδών φτάνει τις 78 και η έκτασή του φτάνει τις τρεισήμισι οκτάβες, καθώς κάθε νότα αποτελείται από τριπλές χορδές. Μπορεί να παίξει και μονοφωνικά (μία φωνή) και πολυφωνικά (συγχορδίες).

[4] Βλ. Διονύσιος Πολίτης, Μουσική Πληροφορική, ο.π. σ. 33.
[5] Βλ. Φοίβος Ανωγειανάκης, Ελληνικά λαϊκά μουσικά όργανα, Β΄ έκδοση, εκδόσεις Μέλισσα, Αθήνα 1991, σ.294.
[6] 6 Βλ. The new Grove , Dictionary of Music and Musicians, second edition, edited by Stanley Sadie, executive Editor John Tyrrell, volume 20, Macmillan Publishers Limited, 2001, σ. 647- 648.
[7] Βλ. Απόστολος Τσαρδάκας, Το κανονάκι στις 78 στροφές, Fagottobooks - Τμήμα Λαϊκής και Παραδοσιακής Μουσικής ΤΕΙ Ηπείρου, Αθήνα 2008, σ.8.

ΚΕΦΑΛΑΙΟ 2

2.1 ΜΟΥΣΙΚΗ ΤΕΧΝΟΛΟΓΙΑ

Η έννοια της "Μουσικής Τεχνολογίας" προκύπτει από την ένωση της μουσικής με τις επιστήμες της Πληροφορικής και της Ηλεκτρονικής. Αρχικά, η μουσική καταγραφόταν στο φωνόγραφο, ο οποίος μετέτρεπε την ακουστική ενέργεια σε μηχανική. Στην ουσία κατέγραφε τις δονήσεις του αέρα προκαλώντας διαφορετικού βάθους εγκοπές πάνω σε ένα κύλινδρο από κασσίτερο. Ο φωνόγραφος ήταν εφεύρεση του Thomas Edison το 1877. Ακολούθησαν στην συνέχεια διάφορες εφευρέσεις και ανακαλύψεις που εξέλισσαν όλο και περισσότερο τη διαδικασία καταγραφής του ήχου μέχρι και την εγγραφή σε δίσκους CD.

Σημαντικό βέβαια ρόλο στην σύνθεση και την επεξεργασία και παραγωγή ενός μουσικού dτων τεχνολογιών που μπορεί να διαχειριστεί, έχει τη δυνατότητα της παραγωγής μουσικού υλικού τόσο ένας επαγγελματίας και ένας ερασιτέχνης.

2.1.1 ΑΠΟ ΤΟΝ ΑΝΑΛΟΓΙΚΟ ΗΧΟ ΣΤΟ ΨΗΦΙΑΚΟ

Καθημερινά λαμβάνουμε και κατ' επέκταση αντιλαμβανόμαστε χιλιάδες πληροφορίες. Ακόμα και τις φυσικές μεταβολές (π.χ. την πίεση του αέρα), μπορούμε να την αντιληφθούμε και να αποτελέσει για εμάς πληροφορία. Οι πληροφορίες αυτές είναι σε αναλογική μορφή. Με τον όρο αναλογική εννοούμε πως το εκάστοτε σήμα μεταβάλλεται ανάλογα με τον τρόπο μεταβολής του φυσικού μεγέθους. Το αρχικό σήμα λοιπόν, είναι αναλογικό και μπορεί να περιγραφεί ως μια συνεχής κυματομορφή που αντιπροσωπεύει τις αλλαγές στην πίεση του αέρα, οι οποίες δημιουργούνται από τον ήχο.

Ο κόσμος του υπολογιστή ωστόσο, όπως και όλες οι υπόλοιπες τεχνολογίες είναι ψηφιακός[8], καθώς δεν αναγνωρίζει ο υπολογιστής συνεχή σήματα. Επικρατεί λοιπόν, στους υπολογιστές το δυαδικό σύστημα και το ψηφιακό σήμα παίρνει μόνο τις τιμές "0" και "1". Άρα ένα τέτοιο σήμα είναι ψηφιακό.

Για να μετατραπεί ένα σήμα από αναλογικό σε ψηφιακό, περνάει μέσα από τη διαδικασία της δειγματοληψίας για την οποία .χρειαζόμαστε κατάλληλο υλικό και κατάλληλο λογισμικό.

2.1.2 ΔΕΙΓΜΑΤΟΛΗΨΙΑ

Η δειγματοληψία είναι η διαδικασία κατά την οποία τα σήματα συνεχούς χρόνου υπόκεινται σε επεξεργασία με ψηφιακά μέσα[9]. Το αναλογικό σήμα μετριέται σε τακτά χρονικά διαστήματα ώστε να έχουμε όσο το δυνατόν λιγότερη απώλεια πληροφορίας. Αυτές οι μετρήσεις ονομάζονται δείγματα (samples).

Συχνότητα Δειγματοληψίας (Sampling Rate): Η συχνότητα με την οποία λαμβάνονται τα δείγματα από το αναλογικό σήμα. Συνήθεις συχνότητες δειγματοληψίας είναι 44.1 kHz (για CD) και 48 kHz (για επαγγελματικό ήχο). Αυτό σημαίνει ότι το σήμα μετριέται 44,100 ή 48,000 φορές το δευτερόλεπτο, αντίστοιχα.

2.1.3 ΚΒΑΝΤΟΠΟΙΗΣΗ

Μετά τη δειγματοληψία, τα δείγματα αυτά πρέπει να μετατραπούν σε αριθμητικές τιμές που μπορούν να αποθηκευτούν ψηφιακά.

Βάθος Δειγματοληψίας (Bit Depth): Ο αριθμός των bits που χρησιμοποιούνται για την αναπαράσταση κάθε δείγματος. Συνηθισμένα βάθη δειγματοληψίας είναι 16-bit (για CD) και 24-

[8] Βλ. Διονύσιος Πολίτης, Μουσική Πληροφορική, ο.π. σ. 75.
[9] Βλ. Τσιχριτζής Γεώργιος, Σήματα και Συστήματα, Πανεπιστήμιο Πειραιώς, Τμήμα Πληροφορικής Πρόγραμμα Μεταπτυχιακών Σπουδών "Πληροφορική", Αθήνα, σελ. 65.

bit (για επαγγελματικό ήχο). Όσο μεγαλύτερο είναι το βάθος δειγματοληψίας, τόσο πιο ακριβής είναι η αναπαράσταση του αρχικού σήματος.

2.1.4 ΚΒΑΝΤΟΠΟΙΗΣΗ ΚΑΙ ΣΦΑΛΜΑ ΚΒΑΝΤΟΠΟΙΗΣΗΣ

Η διαδικασία της κβαντοποίησης μετατρέπει τα αναλογικά δείγματα σε διακριτές τιμές. Αυτό οδηγεί σε σφάλμα κβαντοποίησης, που είναι η διαφορά μεταξύ της πραγματικής αναλογικής τιμής και της πλησιέστερης διακριτής τιμής.

2.1.5 ΚΩΔΙΚΟΠΟΙΗΣΗ

Τα κβαντοποιημένα δείγματα κωδικοποιούνται σε μια ακολουθία δυαδικών αριθμών (bits), που μπορούν να αποθηκευτούν ή να μεταδοθούν.

2.1.6. ΕΠΑΝΕΓΓΡΑΦΗ ΤΟΥ ΨΗΦΙΑΚΟΥ ΣΗΜΑΤΟΣ

Το ψηφιακό σήμα μπορεί να αναπαραχθεί σε αναλογική μορφή μέσω μιας διαδικασίας που ονομάζεται αποδειγματοληψία, όπου τα ψηφιακά δείγματα μετατρέπονται ξανά σε αναλογικό σήμα.

ΚΕΦΑΛΑΙΟ 3

3.1 ΟΙ ΕΦΑΡΜΟΓΕΣ ΤΗΣ ΜΟΥΣΙΚΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

Η μουσική πληροφορική είναι ένα διεπιστημονικό πεδίο που συνδυάζει την πληροφορική, την τεχνολογία, και τη μουσική για την ανάπτυξη νέων εργαλείων και εφαρμογών. Οι εφαρμογές της μουσικής πληροφορικής είναι ποικίλες και καλύπτουν διάφορους τομείς. Ακολουθούν μερικές από τις κύριες εφαρμογές:

1. ΜΟΥΣΙΚΗ ΣΥΝΘΕΣΗ ΚΑΙ ΠΑΡΑΓΩΓΗ

Η μουσική πληροφορική διευκολύνει τη σύνθεση και την παραγωγή μουσικής μέσω ειδικών λογισμικών και εργαλείων. Τα DAWs (Digital Audio Workstations) όπως το Ableton Live, το Logic Pro, και το FL Studio, επιτρέπουν στους μουσικούς να δημιουργήσουν, να επεξεργαστούν και να μιξάρουν μουσικά κομμάτια ψηφιακά.

2. ΑΝΑΛΥΣΗ ΚΑΙ ΕΠΕΞΕΡΓΑΣΙΑ ΗΧΟΥ

Τα εργαλεία ανάλυσης ήχου χρησιμοποιούνται για τη μελέτη των χαρακτηριστικών του ήχου, όπως το φάσμα, ο ρυθμός, και η μελωδία. Αυτές οι τεχνολογίες μπορούν να χρησιμοποιηθούν για την ανίχνευση και την επεξεργασία ήχου, όπως η αφαίρεση θορύβου, η εξίσωση, και η ανάλυση συχνοτήτων.

3. ΜΟΥΣΙΚΗ ΕΚΠΑΙΔΕΥΣΗ

Η μουσική πληροφορική υποστηρίζει την εκπαίδευση μέσω διαδραστικών λογισμικών και εφαρμογών. Πλατφόρμες όπως το SmartMusic και το Yousician παρέχουν μαθήματα μουσικής, εξάσκηση και ανατροφοδότηση σε πραγματικό χρόνο για διάφορα μουσικά όργανα.

4. ΜΟΥΣΙΚΑ ΟΡΓΑΝΑ ΚΑΙ ΕΦΕ

Η ανάπτυξη ψηφιακών μουσικών οργάνων (συνθεσάιζερ, samplers) και εφέ (reverb, delay, distortion) είναι σημαντική εφαρμογή της μουσικής πληροφορικής. Τα VST plugins (Virtual Studio Technology) επιτρέπουν στους μουσικούς να χρησιμοποιούν εικονικά όργανα και εφέ σε ψηφιακές ηχογραφήσεις.

5. ΜΟΥΣΙΚΗ ΒΙΟΜΗΧΑΝΙΑ ΚΑΙ ΔΙΑΝΟΜΗ

Η μουσική πληροφορική επηρεάζει επίσης τον τρόπο με τον οποίο η μουσική διανέμεται και καταναλώνεται. Πλατφόρμες όπως το Spotify, το Apple Music, και το SoundCloud χρησιμοποιούν τεχνολογίες όπως η ανάλυση δεδομένων και οι αλγόριθμοι σύστασης για να προσφέρουν εξατομικευμένες εμπειρίες ακρόασης.

6. ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΚΑΙ ΕΦΑΡΜΟΓΩΝ

Οι προγραμματιστές χρησιμοποιούν γλώσσες προγραμματισμού όπως την Python και τη C++ για την ανάπτυξη λογισμικού μουσικής. Οι εφαρμογές αυτές μπορούν να περιλαμβάνουν από προγράμματα σύνθεσης μουσικής έως και εφαρμογές για ζωντανές παραστάσεις.

7. ΤΕΧΝΗΤΉ ΝΟΗΜΟΣΎΝΗ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

Η τεχνητή νοημοσύνη (AI) και η μηχανική μάθηση χρησιμοποιούνται για την αυτόματη δημιουργία μουσικής, την αναγνώριση μοτίβων, και την ανάλυση μουσικών δεδομένων. Παραδείγματα

περιλαμβάνουν συστήματα όπως το OpenAI's MuseNet, το οποίο μπορεί να συνθέσει μουσική σε διάφορα στυλ.

8. ΔΙΑΔΡΑΣΤΙΚΗ ΜΟΥΣΙΚΗ ΚΑΙ ΕΙΚΟΝΙΚΗ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑ (VR)

Η μουσική πληροφορική συμβάλλει στη δημιουργία διαδραστικών μουσικών εμπειριών και εφαρμογών εικονικής πραγματικότητας. Αυτές οι εφαρμογές επιτρέπουν στους χρήστες να αλληλεπιδρούν με τη μουσική με νέους τρόπους, όπως μέσα από εικονικές συναυλίες ή διαδραστικά περιβάλλοντα.

Η μουσική πληροφορική συνεχίζει να εξελίσσεται, φέρνοντας μαζί της νέες τεχνολογίες και μεθόδους που επηρεάζουν όλους τους τομείς της μουσικής βιομηχανίας, από τη δημιουργία και την παραγωγή έως την εκπαίδευση και την κατανάλωση.

3.1.2 Η ΤΕΧΝΟΛΟΓΙΑ VST

Η τεχνολογία VST (Virtual Studio Technology) είναι μια πλατφόρμα που αναπτύχθηκε από την εταιρεία Steinberg και χρησιμοποιείται ευρέως στη μουσική παραγωγή. Τα VST plugins επιτρέπουν στους μουσικούς και τους παραγωγούς να επεκτείνουν τις δυνατότητες των ψηφιακών τους στούντιο (DAWs) με εικονικά όργανα και εφέ. Η τεχνολογία VST αναπτύχθηκε και κυκλοφόρησε για πρώτη φορά το 1996 από την Steinberg με το λογισμικό Cubase 3.0 και πολύ γρήγορα εξελίχθηκε και έγινε το πρότυπο για τα περισσότερα DAWs, επιτρέποντας τη χρήση τρίτων plugins.

Τύποι VST Plugins

1. VST Instruments (VSTi): Εικονικά όργανα που παράγουν ήχο, όπως συνθεσάιζερ, drum machines, και samplers.

2. VST Effects (VST fx): Εφέ που επεξεργάζονται ήχους, όπως reverb, delay, chorus, EQ, και distortion.

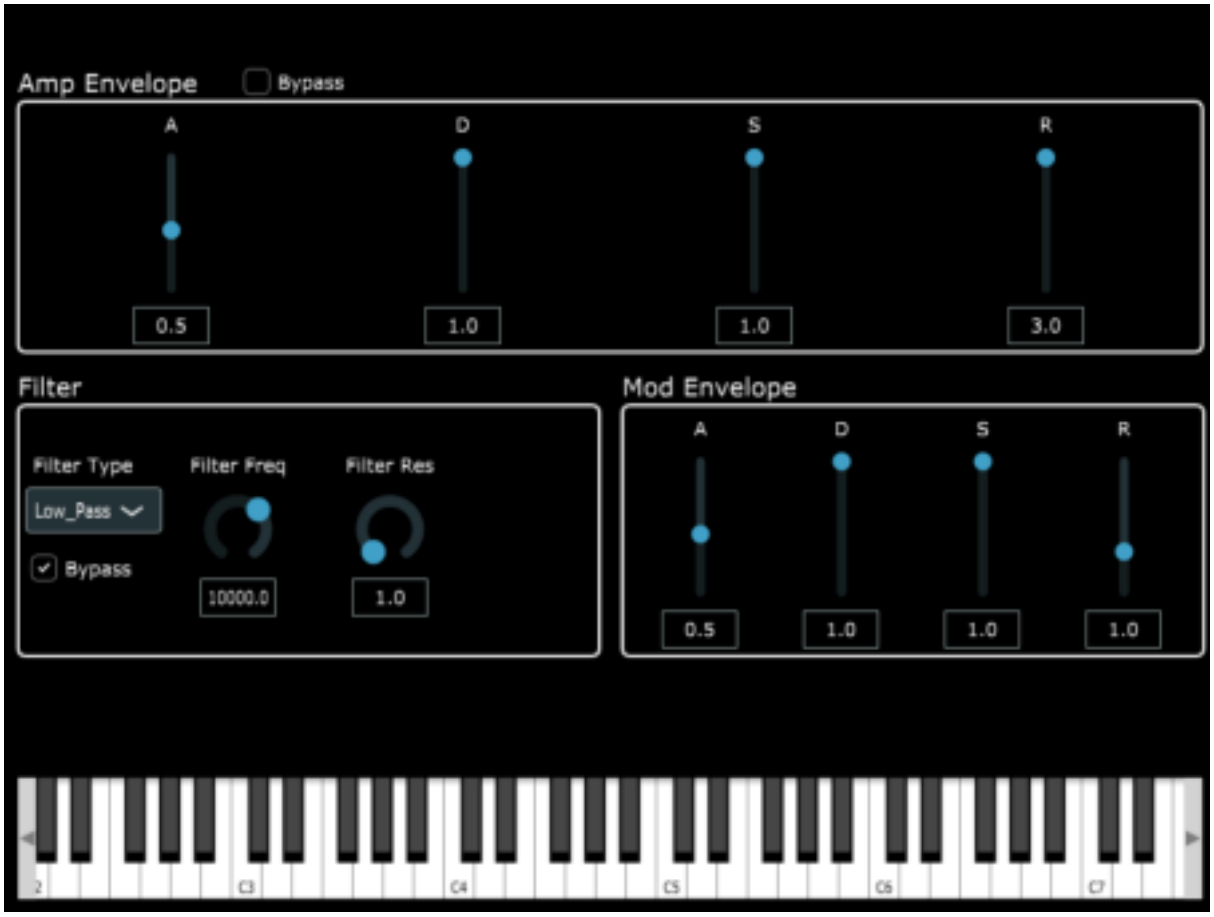
3. VST MIDI Effects: Επεξεργάζονται δεδομένα MIDI, επηρεάζοντας τον τρόπο με τον οποίο παίζεται ή παράγεται η μουσική.

Τα VST plugins προσφέρουν μεγάλη ευελιξία και δυνατότητες επέκτασης για τους χρήστες των DAWs. Τα περισσότερα DAWs υποστηρίζουν τη μορφή VST, καθιστώντας την μια από τις πιο συμβατές μορφές για ψηφιακά στούντιο. Επίσης, τα VST plugins μπορούν να προσφέρουν υψηλή ποιότητα ήχου, συχνά αντίστοιχη με αναλογικά όργανα και εφέ και επιπλέον, επιτρέπουν στους μουσικούς να πειραματιστούν με διάφορους ήχους και εφέ, προσφέροντας απεριόριστες δυνατότητες δημιουργίας.

ΚΕΦΑΛΑΙΟ 4

ΥΛΟΠΟΙΗΣΗ VST

Όπως έχει ήδη αναφερθεί, ουσιαστικά υλοποιήθηκε ένα VST. Η εικόνα που παρατίθεται παρακάτω αφορά στο VST, στο οποίο αναφέρεται η συγκεκριμένη εργασία.



Για την υλοποίηση του VST χρησιμοποιήθηκε το Visual Studio και η γλώσσα προγραμματισμού C++. Τα κεντρικά αρχεία του προγράμματος είναι τα εξής: Plugin Editor και Plugin Processor.

Ο επεξεργαστής Audio Processor Editor είναι ο ενορχηστρωτής για το εμφανισιακό κομμάτι, για αυτό δηλαδή που φαίνεται, και στον constructor γίνεται το “σετάρισμα”, όλων των επιμέρους κομματιών που απαρτίζουν το UI.

```
#include "PluginEditor.h"
#include "PluginProcessor.h"

//==============================================================================
TapInstAudioProcessorEditor::TapInstAudioProcessorEditor(
    TapInstAudioProcessor &p)
    : AudioProcessorEditor(&p), audioProcessor(p),
      adsr("Amp Envelope", audioProcessor.apvts, "ATTACK", "DECAY", "SUSTAIN",
           "RELEASE", "adsrBypass"),
      filter(audioProcessor.apvts, "FILTERTYPE", "FILTERFREQ", "FILTERRES",
             "filterBypass"),
      modAdsr("Mod Envelope", audioProcessor.apvts, "MODATTACK", "MODDECAY",
              "MODSUSTAIN", "MODRELEASE", ""),
      midiKeyboardComponent(
          p.midiKeyboardState,
          juce::MidiKeyboardComponent::horizontalKeyboard) // Corrected line


 setSize(620, 500);
 addAndMakeVisible(adsr);
 addAndMakeVisible(filter);
 addAndMakeVisible(modAdsr);
 midiKeyboardComponent.setMidiChannel(1);
 midiKeyboardComponent.setVelocity(1.0, true);
 midiKeyboardComponent.setKeyWidth(20);
 addAndMakeVisible(midiKeyboardComponent);
 setSize(800, 600);


TapInstAudioProcessorEditor::~TapInstAudioProcessorEditor() {}

//==============================================================================
void TapInstAudioProcessorEditor::paint(juce::Graphics &g) {
  g.fillAll(juce::Colours::black);
}

void TapInstAudioProcessorEditor::resized() {
  const auto paddingX = 5;
  const auto paddingY = 35;
  const auto paddingY2 = 235;
  // Calculate the full width available for the adsr component, subtracting
  // padding from both sides
  const auto fullWidth = getWidth() - (paddingX * 2);
  const auto height = 200;

  // Set the bounds of the adsr component to take the full width
  adsr.setBounds(paddingX, paddingY, fullWidth, height);

  // Adjust the positions of other components as necessary
  filter.setBounds(paddingX, paddingY2, fullWidth / 2 - paddingX / 2,
                   height); // Half width for filter, adjust if necessary
  modAdsr.setBounds(filter.getRight() + paddingX, paddingY2,
                    fullWidth / 2 - paddingX / 2,
                    height); // Half width for modAdsr, placed next to filter

  midiKeyboardComponent.setBounds(
      getLocalBounds().reduced(10).removeFromBottom(80));
}
```

Η κλάση Plugin Processor είναι υπεύθυνη για τον ήχο, για ότι έχει να κάνει δηλαδή με το ακουστικό κομμάτι. Στον constructor του Plugin Processor φορτώνεται μια κλάση που λέγεται synthesizer, η οποία παρέχεται από το ίδιο το framework JUCE και η οποία έχει δύο μεθόδους: την addSound και την addVoice, οι οποίες είναι υπεύθυνες για την παραγωγή του ήχου.

```cpp
#pragma once

#include <JuceHeader.h>

class InstSound : public juce::SynthesiserSound {
public:
  bool appliesToNote(int midiNoteNumber) override { return true; }
  bool appliesToChannel(int midiChannel) override { return true; }
};
```

```cpp
#include "InstVoice.h"
#include "PluginProcessor.h" // Include the full definition of TapInstAudioProcessor

InstVoice::InstVoice(TapInstAudioProcessor &processor)
    : processorRef(processor) {}

bool InstVoice::canPlaySound(juce::SynthesiserSound *sound) {
  return dynamic_cast<juce::SynthesiserSound *>(sound) != nullptr;
}

void InstVoice::startNote(int midiNoteNumber, float velocity,
                          juce::SynthesiserSound *, int) {
  auto it = processorRef.noteToSampleMap.find(midiNoteNumber);
  if (it != processorRef.noteToSampleMap.end()) {
    fillPreloadBuffer(it->second.first.get(), midiNoteNumber);
    hasSampleForNote = true;
    samplePlayhead = 0;
  } else {
    hasSampleForNote = false;
  }
```

Έτσι λοιπόν, στο αρχείο Plugin Processor, βάζουμε στο synthesizer το Voice και το Sound και φορτώνουμε τα samples:

```cpp
{
  mFormatManager.registerBasicFormats();
  // Determines what kind of sound this synthesiser voice can play.
  inst.addSound(new InstSound());
  loadFile();
  inst.addVoice(new InstVoice(*this));
  /* const int numberOfVoices = 8; // For example, 8 voices for polyphony */

  // Add multiple voices in a loop
  /* for (int i = 0; i < numberOfVoices; ++i) { */
  inst.addVoice(new InstVoice(*this));
  /* } */
}
```

Εδώ αξίζει να σημειωθεί, η συνάρτηση loadFile, η οποία είναι υπεύθυνη για την φόρτωση στη μνήμη των διαθέσιμων sample. Τα samples φορτώνονται από το Music Directory και για να είναι διαθέσιμα τα samples αυτά πρέπει όποιος έχει το πρόγραμμα αυτό, να έχει στο Music Directory του, το φάκελο “Samples”.

```
void TapInstAudioProcessor::loadFile() {
    juce::File appDirectory =
        juce::File::getSpecialLocation(juce::File::userMusicDirectory);
   juce::File sampleDirectory = appDirectory.getChildFile("Samples");

  // Get the user's home directory
 // juce::File homeDirectory =
      juce::File::getSpecialLocation(juce::File::userHomeDirectory);
  // Append the "Samples" directory to the path
  //juce::File sampleDirectory = homeDirectory.getChildFile("Dafni_Samples");

  // Check if the directory exists
  if (!sampleDirectory.exists() && !sampleDirectory.createDirectory()) {
    DBG("Failed to create sample directory.");
    return;
  }
```

Όπως προαναφέρθηκε, αφού διαβάσουμε αυτά τα samples, τα φορτώνουμε στην μνήμη και στην ουσία κάνουμε μια συσχέτιση μεταξύ του αρχείου και της οκτάβας και το κρατάμε στην μνήμη.

```
  std::vector<juce::File> files;
  for (auto entry :
       juce::RangedDirectoryIterator(sampleDirectory, false, "*.wav")) {
    files.push_back(entry.getFile());
  }

  // Step 2: Sort the vector by filenames
  std::sort(files.begin(), files.end(),
            [](const juce::File &a, const juce::File &b) {
              return a.getFileName() <
                     b.getFileName(); // Sort alphabetically by filename
            });

  // Step 3: Process files in sorted order
  for (const auto &file : files) {
    auto noteNumber =
        filenameToMidiNoteNumber(file.getFileNameWithoutExtension());
    std::unique_ptr<juce::AudioFormatReader> reader(
        mFormatManager.createReaderFor(file));
    if (reader) {
      DBG("Loaded file: " << file.getFullPathName()
                          << " as note: " << noteNumber);
      noteToSampleMap[noteNumber] =
          std::make_pair(std::make_unique<juce::AudioFormatReaderSource>(
                             reader.release(), true),
                         file.getFullPathName());
    }
  }
}
int TapInstAudioProcessor::filenameToMidiNoteNumber(
    const juce::String &filename) {
  auto noteName = filename.toLowerCase().retainCharacters("cdefgab#");
  auto octave = filename.retainCharacters("0123456789").getIntValue();
  auto midiNoteNumber = noteNameToMidiMap[noteName] + ((octave + 2) * 12);
  return midiNoteNumber;
}
```

Ωστόσο, όταν παίζεται μια νότα καλούνται κάποιες συναρτήσεις από τη βιβλιοθήκη του Juce: η prepareToPlay και η συνάρτηση processBlock. Αυτό που στην ουσία κάνει η prepareToPlay, η οποία καλείται από τον processor, είναι ότι καλεί το Voice και με αυτή τη διαδικασία αρχικοποιούνται οι παράμετροι των φίλτρων (Filter), η δειγματοληψία (sample Rate) κ.α., τη δεδομένη στιγμή που ο χρήστης παίξει μια συγκεκριμένη νότα.

```
void TapInstAudioProcessor::prepareToPlay(double sampleRate,
                                          int samplesPerBlock) {
  inst.setCurrentPlaybackSampleRate(sampleRate);

  for (int i = 0; i < inst.getNumVoices(); i++) {
    if (auto voice = dynamic_cast<InstVoice *>(inst.getVoice(i))) {
      voice->prepareToPlay(sampleRate, samplesPerBlock,
                           getTotalNumOutputChannels());
    }
  }
}

void InstVoice::startNote(int midiNoteNumber, float velocity,
                          juce::SynthesiserSound *, int) {
  auto it = processorRef.noteToSampleMap.find(midiNoteNumber);
  if (it != processorRef.noteToSampleMap.end()) {
    fillPreloadBuffer(it->second.first.get(), midiNoteNumber);
    hasSampleForNote = true;
    samplePlayhead = 0;
  } else {
    hasSampleForNote = false;
  }
}

void InstVoice::fillPreloadBuffer(juce::AudioFormatReaderSource *readerSource,
                                  int midiNoteNumber) {
  if (!readerSource)
    return;

  auto *reader = readerSource->getAudioFormatReader();

  // Original buffer size based on the file's sample rate
  juce::AudioSampleBuffer originalBuffer(
      reader->numChannels, static_cast<int>(reader->lengthInSamples));
  reader->read(&originalBuffer, 0, static_cast<int>(reader->lengthInSamples), 0,
               true, true);

  // Calculate the resample ratio
  double resampleRatio = reader->sampleRate / processorRef.getSampleRate();

  // Calculate the required size for the resampled buffer
  int resampledNumSamples = static_cast<int>(
      std::ceil(originalBuffer.getNumSamples() / resampleRatio));
  preloadBuffer.setSize(reader->numChannels, resampledNumSamples);

  // Perform resampling
  for (int channel = 0; channel < reader->numChannels; ++channel) {
    juce::LagrangeInterpolator interpolator;
    interpolator.process(resampleRatio, originalBuffer.getReadPointer(channel),
                         preloadBuffer.getWritePointer(channel),
                         resampledNumSamples);
  }
}
void InstVoice::stopNote(float velocity, bool allowTailOff) {
  adsr.noteOff();
  filterAdsr.noteOff();
  if (!allowTailOff || !adsr.isActive()) {
    clearCurrentNote();
    currentSampleSource = nullptr; // Ensure sample playback stops
    hasSampleForNote = false;
  }
}
```

Επομένως, το prepareToPlay προετοιμάζει το τί θα παίξει και το processBlock αφορά στην ώρα που ο ήχος παίζει. Έπειτα, καλείται η μέθοδος startNote στην κλάση Voice, η οποία αφού εντοπίσει το κατάλληλο sample με βάση τη νότα που έχει πατήσει ο χρήστης, κάνει resample στον ήχο, και αποθηκεύει αυτό το επεξεργασμένο τμήμα ήχου σε ένα προσωρινό buffer. Η processToBlock, η οποία καλείται στην συνέχεια, διαβάζει όλες τις παραμέτρους των φίλτρων και τις αναβαθμίζει στην κλάση Voice με βάση τις εκάστοτε επιλογές του χρήστη.

Ύστερα, καλείται το renderNextBlock, απ την ίδια τη βιβλιοθήκη, με έναν άδειο buffer, όπου διαβάζουμε το επεξεργασμένο sample που έχει αρχικοποιήσει η μέθοδος startNote και αντιγράφουμε αυτόν τον buffer, στον άδειο buffer, που θα χρησιμοποιηθεί τελικά από τη κλάση synthesizer. Τέλος, ο buffer αυτός, περνάει μια extra διαδικασία, μέσα από τα φίλτρα που είχαμε

αρχικοποιήσει και ανανεώσει στις προηγούμενες παραμέτρους, ώστε στο τέλος, να παραχθεί ο τελικός ήχος.

```
inst.renderNextBlock(buffer, midiMessages, 0, buffer.getNumSamples());
```

Όλα τα προαναφερθέντα φίλτρα βρίσκονται μέσα στα αρχεία: AdsrData και FilterData. Έχουμε δύο ίδια instances του Adsr Data γιατί το ένα ελέγχει το φίλτρο και το άλλο ελέγχει τον ανεπεξέργαστο ήχο. Η λειτουργία ωστόσο, είναι ίδια (Χειριζόμαστε τις παραμέτρους Attack, Delay, Sustain και Release).Το Adsr Data κρατάει την τιμή των παραμέτρων αυτών και το FilterData κρατάει τον τύπο του φίλτρου (υψιπερατό, βαθυπερατό και φίλτρο μεσαίων συχνοτήτων) και θέτει μεταξύ άλλων τα όρια των συχνοτήτων αυτών σύμφωνα με τις επιλογές του χρήστη.

Τέλος, έχουμε τον φάκελο UI, ο οποίος αφορά καθαρά στο εμφανισιακό μέρος. Αυτό περιλαμβάνει τα διάφορα sliders, checkboxes, labels, και γενικά το εμφανισιακό κομμάτι των επιμέρους components.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η μουσική αποτελεί έναν αναπόσπαστο μέρος της ανθρώπινης κουλτούρας και εμπειρίας, εξελισσόμενη διαρκώς από τις πρώτες μελωδίες των αρχαίων πολιτισμών μέχρι τις σύγχρονες ψηφιακές συνθέσεις. Η τεχνολογία της μουσικής έχει αναπτυχθεί δραματικά, επιτρέποντας νέες μορφές δημιουργίας και αναπαραγωγής ήχου που ήταν αδιανόητες πριν από λίγες δεκαετίες.

Η κατανόηση των βασικών αρχών του ήχου, όπως η φύση του ήχου, οι μουσικοί τόνοι και η μετάβαση από τον αναλογικό στον ψηφιακό ήχο, μας προσφέρει μια βαθύτερη εκτίμηση της πολυπλοκότητας και της ομορφιάς της μουσικής. Επιπλέον, τα μουσικά όργανα, όπως το κανονάκι, παίζουν καθοριστικό ρόλο στην δημιουργία και ερμηνεία της μουσικής, προσφέροντας μοναδικούς ήχους και χροιές.

Συνολικά, η μελέτη της μουσικής και της τεχνολογίας της ανοίγει νέους δρόμους για την καλλιτεχνική έκφραση και την πολιτιστική κατανόηση. Η συνεχής εξέλιξη αυτών των πεδίων μας επιτρέπει να διατηρούμε ζωντανή την παράδοση, ενώ παράλληλα να καινοτομούμε και να δημιουργούμε νέες μουσικές εμπειρίες για τις μελλοντικές γενιές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ελληνική

Πολίτης, Διονύσιος, editor. *Μουσική Πληροφορική*. Αθήνα, Κλειδάριθμος, 2007. Σπυρίδης, Χαράλαμπος Χ., editor. *Η Φυσική των Μουσικών Οργάνων*. Θεσσαλονίκη, grapholine.

Αβέρωφ, Έφη, editor. *Εισαγωγή στην Οργανογνωσία*. Πέμπτη ed., Αθήνα, Μουσικός Οίκος Φίλιππος Νάκας, 1992.

Ανωγειανάκης, Φοίβος, editor. *Ελληνικά λαϊκά μουσικά όργανα*. Β έκδοση ed., Αθήνα, Μέλισσα, 1991.

Τσιχριτζής, Γεώργιος, editor. *Σήματα και Συστήματα*. Πανεπιστήμιο Πειραιά ed., Αθήνα, Σημειώσεις διδασκαλίας.

Τσαρδάκας, Απόστολος, editor. *Το κανονάκι στις 78 στροφές*. Αθήνα, Fagottobooks - Τμήμα Λαϊκής και Παραδοσιακής Μουσικής ΤΕΙ Ηπείρου, 2008.

Ξένη

Grove, T., editor. *Dictionary of Music and Musicians*. Β έκδοση ed., vol. 20, Stanley Sadie, 2001.