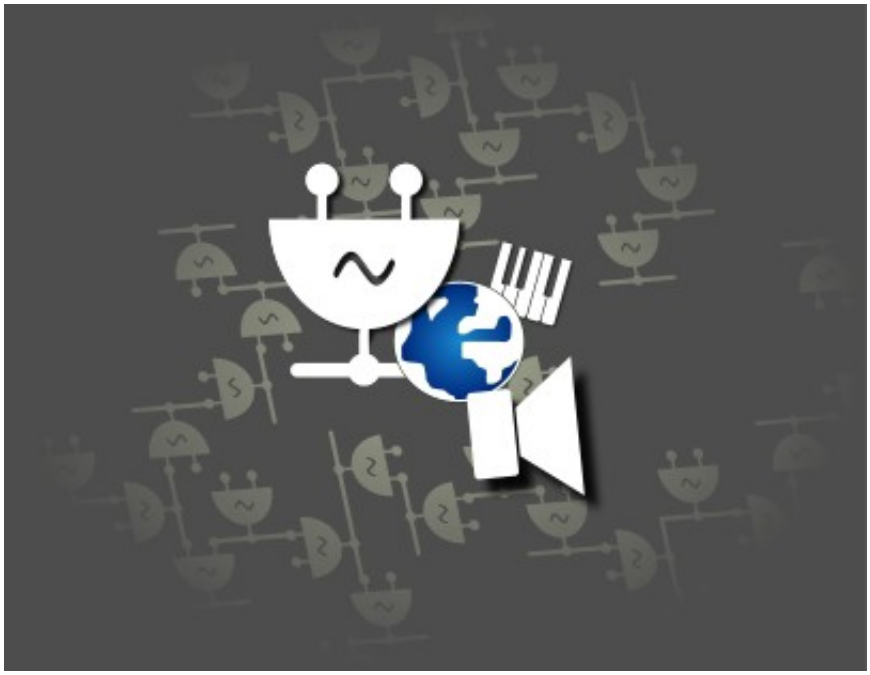
Α.Τ.Ε.Ι. ΚΡΗΤΗΣ – ΠΑΡΑΡΤΗΜΑ ΡΕΘΥΜΝΟΥ
ΤΜΗΜΑ ΜΟΥΣΙΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ & ΑΚΟΥΣΤΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Γραφικό περιβάλλον δικτυακής-συνεργατικής ανάπτυξης αλγόριθμων σύνθεσης ήχου



Γαστεράτος Πέτρος (Α.Μ. 573), Ρέθυμνο 2008

Επίβλεψη: Αλεξανδράκη Χρυσούλα

Περίληψη

Η πρόοδος στην τεχνολογία των υπολογιστών, καθιστά πλέον δυνατή την ανάπτυξη εφαρμογών με μεγάλες απαιτήσεις σε επεξεργαστική ισχύ. Η ανάπτυξη και η διάδοση των δικτύων ηλεκτρονικών υπολογιστών σε συνδυασμό με ικανά υπολογιστικά συστήματα γέννησε μια νέα κατηγορία εφαρμογών: τα εικονικά περιβάλλοντα δικτυακής συνεργασίας. Οι εφαρμογές αυτές είναι αντικείμενο μελέτης μίας πρόσφατης σχετικά ερευνητικής περιοχής η οποία ασχολείται με το σχεδιασμό και την υλοποίηση τεχνολογικών συστημάτων που υποστηρίζουν την εξ' αποστάσεως συνεργασία.

Η παρούσα εργασία εξερευνά τον τρόπο με τον οποίο μπορεί να υλοποιηθεί ένα σύστημα δικτυακής-συνεργατικής ανάπτυξης αλγόριθμων σύνθεσης ήχου με βάση τις σύγχρονες τεχνολογίες και τα σύγχρονα πρότυπα. Η αναζήτηση αυτή διενεργείται μέσω της υλοποίησης ενός τέτοιου συστήματος το οποίο αποτελείται από δύο αυτόνομες μονάδες. Η πρώτη επικεντρώνεται στην εξασφάλιση των εργαλείων που απαιτούνται για την υλοποίηση αλγόριθμων σύνθεσης ήχου και την άμεση και ταχεία εκτέλεση τους, ενώ η δεύτερη επικεντρώνεται στην υποστήριξη συνεργατικής υλοποίησης τέτοιων αλγόριθμων, όπου τα συνεργαζόμενα μέλη επικοινωνούν μεταξύ τους μέσω δικτύου υπολογιστών.

Abstract

Current progress in computer technology enables the development of CPU demanding applications. The proliferation of computer networks in combination with the wide availability of efficient computer systems has lead to the emergence of a new category of software applications, namely "virtual collaboration environments". The development of virtual collaboration environments are the result of a relatively recent and multidisciplinary research field, named Computer Supported Cooperative Work (CSCW), which aims at supporting people at their work through the facilitation of various technological means.

This thesis presents the development of a virtual collaboration environment that provides the means for collaboratively building sound synthesis algorithms. This environment is comprised of two separate units; the first unit provides the user with the necessary tools needed to develop and execute sound synthesis algorithms, while the second unit deals with supporting remote collaboration between distant users by utilizing several Internet technologies.

Περιεχόμενα

1 Εισαγωγή

1.1 Βασική ιδέα

Η παρούσα πτυχιακή εργασία αφορά την ανάπτυξη του λογισμικού που στοχεύει στη γραφική συνεργατική ανάπτυξη αλγορίθμων σύνθεσης ήχου. Το λογισμικό αυτό αποτελείται από δύο κύριες μονάδες: Α) το περιβάλλον ανάπτυξης αλγορίθμων σύνθεσης ήχου και Β) το περιβάλλον δικτυακής συνεργασίας.

Το περιβάλλον σχεδίασης αλγορίθμων είναι ένα γραφικό περιβάλλον το οποίο παρέχει κάποια βασικά δομοστοιχεία (modules) τα οποία μπορεί ο χρήστης να συνδυάσει προκειμένου να υλοποιεί αλγορίθμους σύνθεσης ήχου.

Το περιβάλλον δικτυακής συνεργασίας είναι υλοποιημένο ως μία εφαρμογή ιστού (web application), η οποία παρέχει τη δυνατότητα συνεργατικής υλοποίησης ενός αλγόριθμου σύνθεσης ήχου από δύο (ή περισσότερους) χρήστες, οι οποίοι μπορούν να επεμβαίνουν ταυτόχρονα στην ίδια γραφική αναπαράσταση του αλγόριθμου μέσω ενός κοινού φυλλομετρητή ιστοσελίδων (web browser).

Επειδή το περιβάλλον σχεδίασης μπορεί να λειτουργήσει αυτόνομα και ανεξάρτητα από το το περιβάλλον συνεργασίας μπορούμε να δούμε τα μέρη της εργασίας ως δύο ξεχωριστές εφαρμογές. Για το λόγο αυτό το κάθε μέρος έχει το δικό του όνομα. Το περιβάλλον σχεδίασης έχει ονομαστεί «SoundEngine» και το περιβάλλον συνεργασίας «SharedEngine».

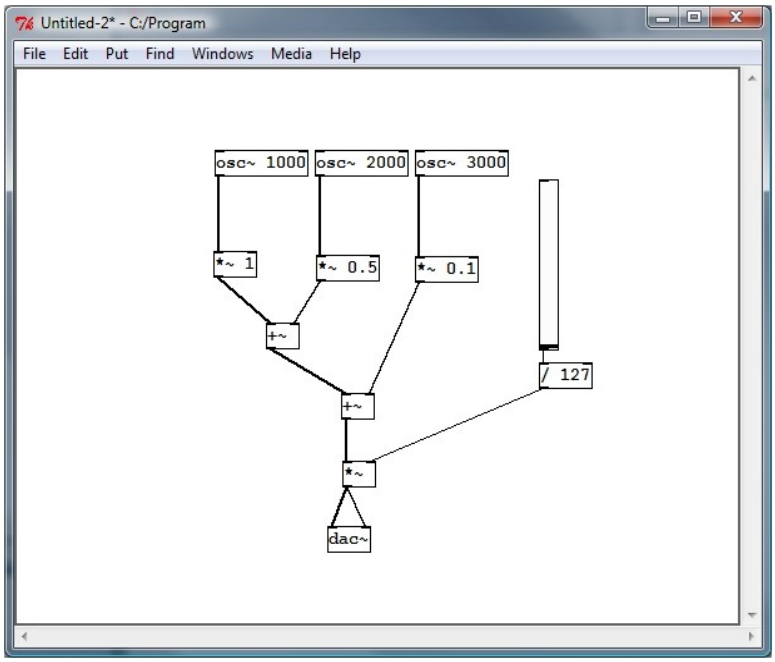
1.2 Modular synthesizers

Όπως έχει ήδη αναφερθεί, το περιβάλλον σχεδίασης αλγορίθμων αποτελεί ένα περιβάλλον στο οποίο ο χρήστης χρησιμοποιώντας κάποια βασικά δομοστοιχεία μπορεί να κατασκευάσει πολύπλοκες ηχητικές διατάξεις. Τα δομοστοιχεία αυτά παρέχονται με τη μορφή γραφικών αντικειμένων τα οποία μπορεί ο χρήστης να τοποθετήσει-μετακινήσει στο χώρο εργασίας καθώς και να συνδέσει με εικονικά καλώδια. Η γραφική αυτή μεταφορά (Graphical User Interface metaphor) είναι δανεισμένη από τα παλαιότερα αναλογικά (ελεγχόμενα από ηλεκτρικές τάσεις) synthesizers της κατηγορίας των “Modular Synthesizers”. Ένα τέτοιο αναλογικό synthesizer αναπαρίσταται στην *Εικόνα 1*.

Τα “modular synthesizers” είναι ένας τύπος ηλεκτρονικών συνθετητών (synthesizers), οι οποίοι δίνουν τη δυνατότητα στο χρήστη να συνδέει με καλώδια ένα σύνολο από γεννήτριες ήχου και επεξεργαστές σήματος τα οποία καλούνται “δομοστοιχεία” [1].

Εικόνα 1: Ένα αναλογικό modular synthesizer

Τα πρώτα modular synthesizers ήταν αναλογικά και είχαν περιορισμένες δυνατότητες καθώς η τεχνολογία της εποχής δεν επέτρεπε κάτι καλύτερο. Σήμερα υπάρχουν modular synthesizers βασισμένα εξολοκλήρου σε λογισμικό[1](*Εικόνα 2*) που εκτελείται σε προσωπικούς υπολογιστές.



Εικόνα 2: Pure Data - Ένα modular synthesizer λογισμικού

1 Για παράδειγμα το περιβάλλον Pure Data.

1.3 Δίκτυα υπολογιστών

Δίκτυο υπολογιστών ονομάζουμε ένα σύνολο από δύο ή περισσότερους υπολογιστές συνδεδεμένους μεταξύ τους με στόχο την ανταλλαγή δεδομένων. Υπάρχουν πολλοί τύποι δικτύων και η ταξινόμηση τους γίνεται με διάφορους τρόπους ανάλογα με τα χαρακτηριστικά που εξετάζονται κάθε φορά [2].

Σήμερα το πιο διαδεδομένο δίκτυο είναι το Internet, ένα υπερ-δίκτυο υπολογιστών με 1,5 δισεκατομμύρια χρήστες (στις 30 Ιουνίου 2008)[1], το οποίο επιτρέπει την επικοινωνία υπολογιστών σε παγκόσμια κλίμακα. Το Internet παρέχει ένα πλήθος υπηρεσιών, όπως το γνωστό ηλεκτρονικό ταχυδρομείο (e-mail) και τον παγκόσμιο ιστό (www).

Εκτός απ' αυτές τις υπηρεσίες το Διαδίκτυο και γενικότερα τα δίκτυα χρησιμοποιούνται για πολλούς και διαφορετικούς σκοπούς, π.χ. για τη ζωντανή επικοινωνία με άλλους, για την ανταλλαγή αρχείων, για δικτυακά παιχνίδια, ηλεκτρονικό εμπόριο, ηλεκτρονικές βιβλιοθήκες κ.ο.κ.

1.4 Περιβάλλοντα συνεργασίας

Μία κατηγορία εφαρμογών που έχουν άμεση σχέση με τα δίκτυα είναι τα **περιβάλλοντα εικονικής συνεργασίας** (virtual collaboration environments).

Τα περιβάλλοντα εικονικής συνεργασίας είναι εφαρμογές λογισμικού οι οποίες επιτρέπουν σε ομάδες ανθρώπων να συνεργάζονται για την επίτευξη ενός κοινού στόχου [3]. Ένα τέτοιο παράδειγμα είναι η ελεύθερη εγκυκλοπαίδεια "Wikipedia"[2]. Ένα περιβάλλον συνεργασίας απαιτεί όλα τα χαρακτηριστικά που έχει μια εφαρμογή που είναι φτιαγμένη για ένα χρήστη, ενώ ταυτόχρονα απαιτεί και επιπλέον λειτουργίες, οι οποίες να επιτρέπουν την εκτέλεση συνεργατικών καθηκόντων [4]. Αυτό το γεγονός καθιστά την ανάπτυξη συνεργατικών εφαρμογών πολυπλοκότερη διαδικασία από την ανάπτυξη εφαρμογών μονού χρήστη.

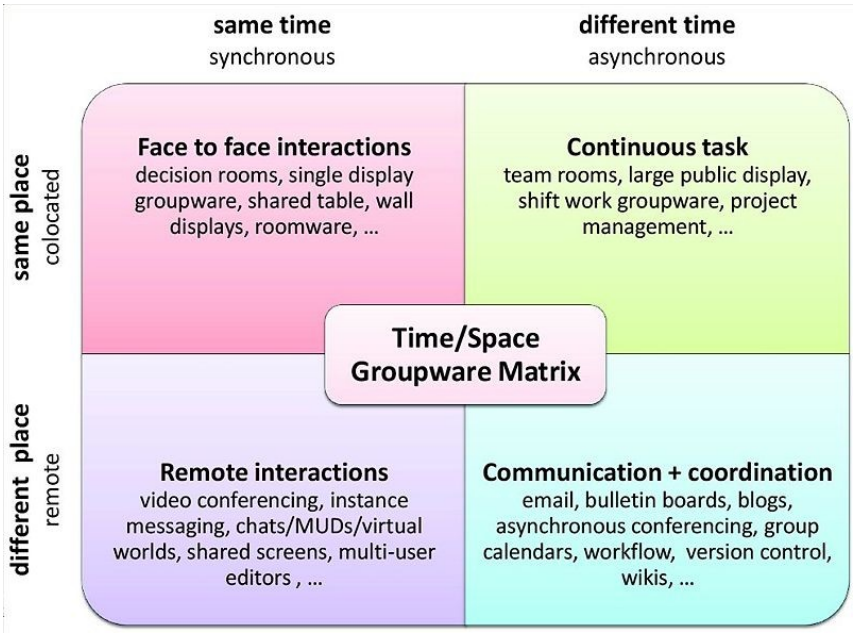
Η υποβοηθούμενη από ηλεκτρονικό υπολογιστή συνεργατική εκτέλεση καθηκόντων (Computer Supported Cooperative Work - CSCW) αποτελεί ένα διεπιστημονικό πεδίο έρευνας, το οποίο ασχολείται με το σχεδιασμό συστημάτων και τεχνολογιών τα οποία να υποστηρίζουν την εκτέλεση συνεργατικών καθηκόντων [5].

Καθώς οι τρόποι να επιτευχθεί η συνεργασία μεταξύ ατόμων είναι πολλοί και διαφορετικοί, αντίστοιχα ένα σύστημα εικονικής συνεργασίας μπορεί να υλοποιηθεί με πολλούς διαφορετικούς τρόπους. Ο τρόπος που θα υλοποιηθεί ένα τέτοιο σύστημα εξαρτάται από τεχνολογικούς και οικονομικούς παράγοντες καθώς και από τα ίδια τα καθήκοντα.

Μια ενδεικτική ταξινόμηση τέτοιων συστημάτων παρουσιάστηκε το 1988 από τον R. Johansen. Η ταξινόμηση αυτή βασίζεται στην έννοια του χώρου και του χρόνου στον οποίο επιτελείται η συνεργασία, δηλαδή εάν τα συνεργαζόμενα μέρη εργάζονται ταυτόχρονα (**σύγχρονα**) ή σε διαφορετικές χρονικές στιγμές (**ασύγχρονα**) καθώς και κατά πόσο η συνεργασία επιτελείται σε κοινό ή σε διαφορετικούς χώρους (βλ. *Εικόνα 3*) [5].

1 Πηγή: World Internet Usage Statistics News and World Population Stats, http://www.internetworldstats.com/stats.htm

2 http://www.wikipedia.org



Εικόνα 3: Ο διαχωρισμός των συστημάτων συνεργασίας με βάση την προσέγγισης του Johansen

Ένα ερώτημα που προκύπτει κατά το σχεδιασμό μιας συνεργατικής εφαρμογής, είναι με ποιο τρόπο μπορεί να επιτελείται η προσπέλαση των πληροφοριών κατά διάρκεια της συνεργασίας. Η κοινή χρήση πόρων μπορεί να προκαλέσει προβλήματα στη συνέπεια των δεδομένων ή των υπηρεσιών που παρέχονται, είτε πρόκειται για ταυτόχρονη προσπέλαση αρχείων, είτε πρόκειται για απομακρυσμένη συνδιάσκεψη, είτε πρόκειται για ταυτόχρονη χρήση βάσεων δεδομένων και γενικότερα για οποιαδήποτε κατηγορία συνεργατικών δραστηριοτήτων [6].

Ένας τρόπος αντιμετώπισης αυτού του προβλήματος είναι ο **έλεγχος του λόγου** (floor control). Ο **έλεγχος του λόγου**, επιτρέπει στους χρήστες δικτυακών εφαρμογών την κοινή χρήση πόρων, αποτρέποντας τις διενέξεις που μπορούν να προκληθούν από την ταυτόχρονη προσπέλαση. Αυτό επιτυγχάνεται μέσω ενός μηχανισμού ελέγχου της πρόσβασης στους πόρους, ο οποίος προβλέπει ότι σε κάθε χρονική στιγμή υπάρχει ένας μόνο χρήστης ο οποίος έχει δικαίωμα πρόσβασης σε αυτούς [6]. Ο μηχανισμός αυτός αξιοποιείται στην εφαρμογή «SharedEngine».

1.5 Σχετιζόμενη έρευνα

Από τα τέλη της δεκαετίας του 1970, από τη στιγμή δηλαδή που άρχισε η εμπορευματοποίηση των προσωπικών ηλεκτρονικών υπολογιστών, καταγράφονται και οι πρώτες προσπάθειες για τη ανάπτυξη εφαρμογών, που αξιοποιούν τα δίκτυα υπολογιστών με στόχο τη συνεργασία στα πλαίσια μουσικών δραστηριοτήτων, δηλαδή τη "μουσική συνεργασία" [7]. Με τον όρο μουσική συνεργασία εννοείται κάθε είδους συνεργασία που σχετίζεται με τη δημιουργία μουσικής. Από τις πρώτες προσπάθειες μέχρι σήμερα, εμφανίστηκαν πολλά πειραματικά συστήματα, τα οποία καταδεικνύουν τους διαφορετικούς τρόπους μουσικής συνεργασίας. Όπως περιγράφει ο Álvaro Barbosa στο [7], τα συστήματα αυτά μπορούν να ταξινομηθούν σε:

- **Τοπικά μουσικά δίκτυα** (Local interconnected musical networks), δηλαδή τοπικά δίκτυα υπολογιστών τα οποία επιτρέπουν τη μουσική αλληλεπίδραση μεταξύ μουσικών εκτελεστών σε πραγματικό χρόνο.
- **Συστήματα μουσικής σύνθεσης** (Musical composition support systems), δηλαδή συστήματα τα οποία βοηθούν τη συνεργατική μουσική σύνθεση και παραγωγή.
- **Συστήματα απομακρυσμένης μουσικής εκτέλεσης** (Remote music performance systems), δηλαδή συστήματα τα οποία επιτρέπουν σε μουσικούς εκτελεστές οι οποίοι βρίσκονται σε διαφορετικές τοποθεσίες να αυτοσχεδιάζουν και να αλληλεπιδρούν σε πραγματικό χρόνο.
- **Κοινόχρηστα ηχητικά περιβάλλοντα** (Shared sonic environments), δηλαδή διαδικτυακές εφαρμογές οι οποίες δεν προσανατολίζονται σε χρονικά-περιορισμένα σενάρια και είναι συνήθως κατάλληλες για συγχρονισμένο αυτοσχεδιασμό.

Στις μέρες μας, τα περιβάλλοντα αλγοριθμικής σύνθεσης ήχου παρέχουν στους χρήστες τη δυνατότητα δικτυακής ανταλλαγής δεδομένων κατά συνέπεια και τη δυνατότητα δικτυακής συνεργασίας. Η δικτυακή επικοινωνία επιτυγχάνεται μέσω του πρωτοκόλλου OSC[1] και περιλαμβάνει τον απομακρυσμένο έλεγχο των παραμέτρων ενός αλγόριθμου καθώς επίσης (όπως συμβαίνει στο περιβάλλον SuperCollider) και την απομακρυσμένη ανάπτυξη αλγόριθμων. Σε αντίθεση με τις υπάρχουσες εφαρμογές, το περιβάλλον που παρουσιάζεται εδώ παρουσιάζει την καινοτομία, ότι η εκτέλεση και η συνεργατική ανάπτυξη αλγόριθμων σύνθεσης ήχου επιτελείται μέσα από ένα κοινό φυλλομετρητή ιστοσελίδων, χωρίς να απαιτείται η απόκτηση ειδικού λογισμικού από τον τελικό χρήστη.

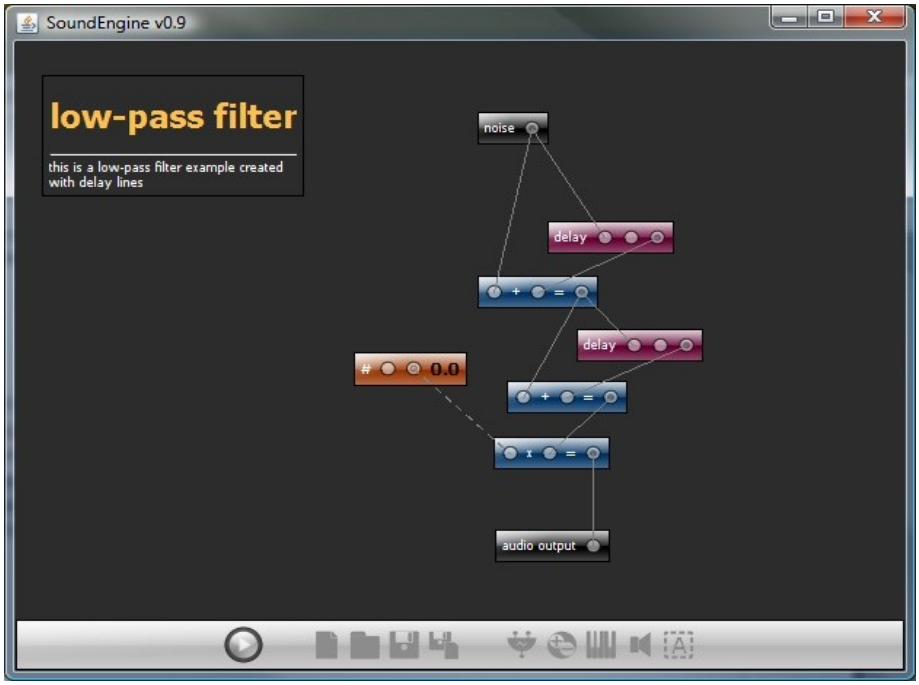
1 Το OSC (OpenSound Control) είναι ένα πρωτόκολλο για την επικοινωνία μεταξύ μουσικών συσκευών και υπολογιστών.

2 Σχεδιασμός

Όπως αναφέρθηκε και στο εισαγωγικό κεφάλαιο η εργασία αυτή αποτελείται από δύο βασικές μονάδες. Το «SoundEngine», δηλαδή το περιβάλλον ανάπτυξης αλγόριθμων σύνθεσης ήχου και το «SharedEngine», δηλαδή το περιβάλλον συνεργασίας. Στο κεφάλαιο αυτό θα περιγραφεί η διαδικασία σχεδιασμού των δύο μονάδων.

2.1 SoundEngine

Το «SoundEngine» είναι ένα modular synthesizer, δηλαδή ένα περιβάλλον στο οποίο ο χρήστης χρησιμοποιεί δομοστοιχεία για την υλοποίηση αλγόριθμων σύνθεσης και επεξεργασίας ήχου. Οι αλγόριθμοι αυτοί από εδώ και στο εξής θα αναφέρονται και ως **soundgraphs**. Κάθε τέτοιο soundgraph απαρτίζεται από γραφικά αντικείμενα τα οποία χειρίζεται ο χρήστης με το ποντίκι και το πληκτρολόγιο και ο τρόπος με τον οποίο συνδυάζονται τα δομοστοιχεία μεταξύ τους είναι με εικονικά καλώδια.



Εικόνα 4: Το SoundEngine

2.1.1 Οι τύποι δεδομένων και οι θύρες επικοινωνίας

Στην *Εικόνα 4* απεικονίζεται ένα soundgraph, όπου μπορεί κανείς να διακρίνει τα δομοστοιχεία και τα καλώδια. Οι συνδέσεις των καλωδίων γίνονται μεταξύ των κατάλληλων θυρών επικοινωνίας οι οποίες χωρίζονται σε εισόδους και εξόδους. Εφόσον έχουμε εισόδους και εξόδους, η επικοινωνία σε κάθε σύνδεση μεταξύ δύο θυρών είναι πάντα μονόδρομη. Η εφαρμογή υποστηρίζει δύο διαφορετικούς τύπους δεδομένων: τα *σήματα* και τα *μηνύματα*.

Ο πρώτος τύπος δεδομένων είναι τα *σήματα*, δηλαδή ακολουθίες από δείγματα τα οποία περιγράφουν ένα ψηφιακό σήμα. Κατά τη διάρκεια εκτέλεσης ενός αλγόριθμου τα σήματα παράγονται ή επεξεργάζονται από τα δομοστοιχεία με συνεχή και σταθερό ρυθμό. Ένα ένα σήμα μπορεί να δρομολογηθεί μέσω του κατάλληλου δομοστοιχείου στην έξοδο της κάρτας ήχου και να

ακουστεί από το ηχοσύστημα του υπολογιστή.

Ο δεύτερος αλλά εξίσου σημαντικός τύπος δεδομένων είναι τα *μηνύματα*. Τα μηνύματα αποτελούν δεδομένα μη προκαθορισμένης μορφής που μπορούν να παραχθούν οποιαδήποτε χρονική στιγμή. Η παραγωγή τους εξαρτάται από το δομοστοιχείο που θα τα παράξει και προκαλούνται συνήθως από κάποιο συμβάν όπως η είσοδος ενός MIDI μηνύματος από κάποιον ελεγκτή MIDI (MIDI controller).

Όπως θα προσέξει κανείς κατά τη χρήση της εφαρμογής υπάρχουν τριών τύπων θύρες, οι οποίες ξεχωρίζουν απ' το διαφορετικό σχεδιασμό τους (βλ. *Εικόνα 5*). Όπως αναφέρθηκε σε προηγούμενη παράγραφο οι θύρες χωρίζονται σε εισόδους και εξόδους, έτσι έχουμε: την έξοδο σήματος, την έξοδο μηνυμάτων και την είσοδο. Η έξοδος σήματος χρησιμοποιείται για εξαγωγή σήματος, η έξοδος μηνυμάτων χρησιμοποιείται για εξαγωγή μηνυμάτων και η είσοδος χρησιμοποιείται για την εισαγωγή και των δύο τύπων δεδομένων.

Πέραν από τον τύπο της θύρας που χρησιμοποιείται για την εξαγωγή δεδομένων, ο τύπος των δεδομένων που μεταφέρονται μεταξύ δύο δομοστοιχείων διαφοροποιείται και από τη γραφική αναπαράσταση του καλωδίου που τα συνδέει. Ειδικότερα τα σήματα μεταφέρονται από καλώδια που απεικονίζονται με συνεχείς γραμμές ενώ τα μηνύματα μεταφέρονται από καλώδια που απεικονίζονται με διακεκομμένες γραμμές.

Εικόνα 5: (α) είσοδος, (β) έξοδος σήματος, (γ) έξοδος μηνυμάτων

2.1.2 Τα δομοστοιχεία

Το SoundEngine σχεδιάστηκε με στόχο να παρέχει στο χρήστη κάποια βασικά δομοστοιχεία και όχι ένα πλήρες σετ από δομοστοιχεία. Στον παρακάτω πίνακα παρουσιάζονται τα δομοστοιχεία που περιλαμβάνονται στο SoundEngine[1].

Όνομα δομοστοιχείου	**Περιγραφή**
Oscillator	Παράγει ημιτονοειδής, τριγωνικές και τετραγωνικές ταλαντώσεις
Noise	Παράγει λευκό θόρυβο
Delay	Καθυστερεί τις τιμές ενός σήματος
Addition	Προσθέτει σήματα και μηνύματα
Subtraction	Αφαιρεί σήματα και μηνύματα
Multiplication	Πολλαπλασιάζει σήματα και μηνύματα

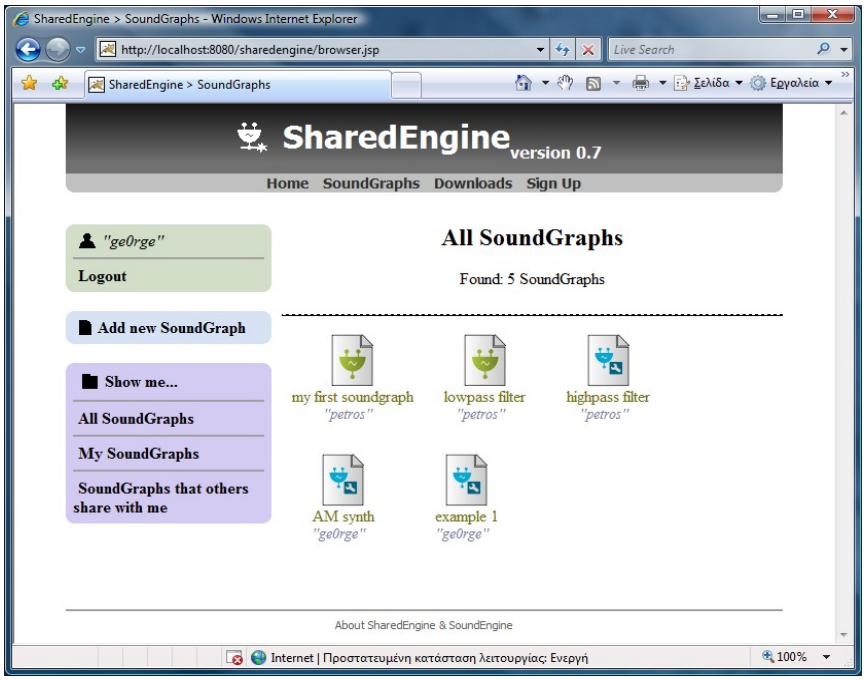
1 Περισσότερες πληροφορίες σχετικά με τη χρήση των δομοστοιχείων παρουσιάζονται στο *Παράρτημα Α*

Όνομα δομοστοιχείου	Περιγραφή
Division	Διαιρεί σήματα και μηνύματα
MIDI input	Εισάγει MIDI μηνύματα από κάποια εξωτερική συσκευή
MIDI output	Εξάγει MIDI μηνύματα σε κάποια εξωτερική συσκευή
MIDI channel filter	Επιτρέπει μόνο στα MIDI μηνύματα που απευθύνονται στο επιλεγμένο κανάλι να περάσουν και απορρίπτει τα υπόλοιπα
MIDI controller filter	Εάν τα “Control Change” μηνύματα που φτάνουν στην είσοδο του ανήκουν στον επιλεγμένο controller τότε εξάγει την τιμή του
OSC receiver	Εισάγει τα OSC μηνύματα που φτάνουν στην επιλεγμένη δικτυακή θύρα
OSC transmitter	Αποστέλλει OSC μηνύματα στην επιλεγμένη δικτυακή διεύθυνση
OSC address filter	Επιτρέπει στα μηνύματα που απευθύνονται στο επιλεγμένο OSC address να περάσουν και απορρίπτει τα υπόλοιπα
Number controller	Χρησιμεύει στον έλεγχο άλλων δομοστοιχείων καθώς με το σύρσιμο του ποντικιού πάνω στην επιφάνεια του εξάγει αριθμητικά μηνύματα ανάλογα της μετατόπισης του δείκτη
Audio input	Μεταφέρει σήμα από την είσοδο της κάρτας ήχου
Audio output	Μεταφέρει σήμα στην έξοδο της κάρτας ήχου
Comment	Χρησιμεύει για την προσθήκη σχολίων στους αλγόριθμου

Πίνακας 1: Τα δομοστοιχεία του SoundEngine

2.2 SharedEngine

Το «SharedEngine», δηλαδή το περιβάλλον δικτυακής συνεργασίας (*Εικόνα 6*), είναι μια εφαρμογή ιστού η οποία επιτρέπει σε δύο ή περισσότερα άτομα την κοινή ανάπτυξη αλγόριθμων σύνθεσης ήχου με σύγχρονο και ασύγχρονο τρόπο. Η σύγχρονη και η ασύγχρονη ανάπτυξη επιτυγχάνεται μέσω του μηχανισμού διαμοιρασμού αλγορίθμων, ο οποίος επιτρέπει στους χρήστες οποιαδήποτε χρονική στιγμή, είτε ατομικά είτε ομαδικά, να παρακολουθούν και να επεξεργάζονται τους αλγόριθμους.



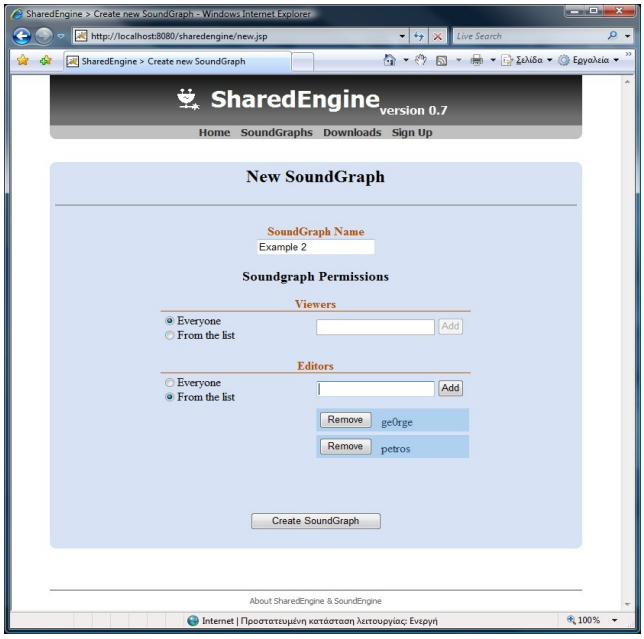
Εικόνα 6: Το SharedEngine

2.2.1 Χρήστες και ρόλοι

Από την εφαρμογή υποστηρίζονται δύο τύποι χρηστών: τα *μέλη* και οι *επισκέπτες*. Τα μέλη αποτελούν μοναδικές οντότητες για την εφαρμογή και η πρόσβαση στις υπηρεσίες για τους χρήστες-μέλη πραγματοποιείται μέσω της διαδικασίας εγγραφής και ταυτοποίησης του χρήστη. Σε αντίθεση με τα μέλη, οι επισκέπτες αποκτούν "ελεύθερη" πρόσβαση στις υπηρεσίες, δηλαδή χωρίς να ταυτοποιηθούν, με αποτέλεσμα η αναγνώριση τους από την εφαρμογή να είναι αδύνατη.

Ενώ η ελεύθερη πρόσβαση στις υπηρεσίες αποτελεί προσόν της εφαρμογής καθώς προσθέτει δυνατότητες όπως η δημοσιοποίηση των soundgraphs, αποτελεί ταυτόχρονα δίοδο για κακόβουλη χρήση. Η αποφυγή κακόβουλης χρήσης της εφαρμογής από τους χρήστες-επισκέπτες επιτυγχάνεται με αφαίρεση της δυνατότητας δημιουργίας νέων αλγορίθμων για αυτή την ομάδα χρηστών.

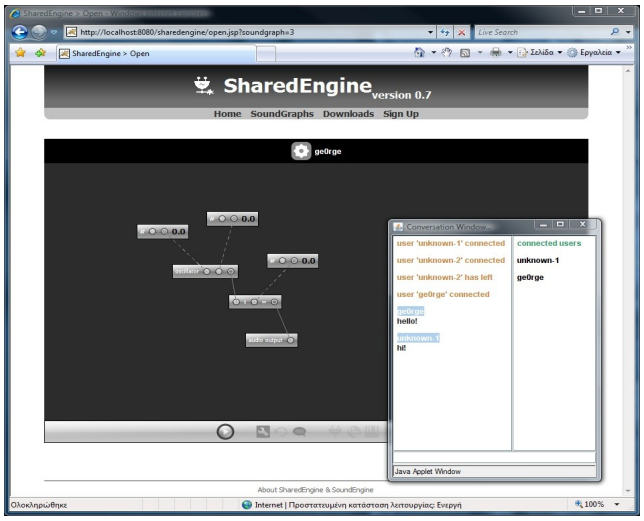
Κάθε soundgraph που δημιουργείται στο «SharedEngine» σχετίζεται με ένα πλήθος από χρήστες. Ο χρήστης με τα περισσότερα προνόμια είναι ο κάτοχος του soundgraph, δηλαδή ο δημιουργός του. Ο δημιουργός επιλέγει το όνομα του soundgraph καθώς και τους χρήστες που θα έχουν πρόσβαση σε αυτό. Αναλυτικότερα επιλέγει ποιοι χρήστες μπορούν να το επεξεργαστούν (και να το χρησιμοποιήσουν) και ποιοι μπορούν μόνο να το χρησιμοποιήσουν. Οι χρήστες που μπορούν να επεξεργαστούν ένα soundgraph ονομάζονται *editors*, ενώ οι χρήστες που μπορούν μόνο να το χρησιμοποιήσουν ονομάζονται *viewers*.



Εικόνα 7: Η καρτέλα δημιουργίας νέου soundgraph

2.2.2 Η διαδικασία συνεργασίας

Μόλις ολοκληρωθεί η διαδικασία δημιουργίας νέου soundgraph (*Εικόνα 7*), τότε οι επιλεγμένοι χρήστες αποκτούν πρόσβαση σε αυτό μέσω της δικτυακής εκδοχής του «SoundEngine» (*Εικόνα 8*). Η χρονική περίοδος κατά την οποία ένας οι περισσότεροι χρήστες επεξεργάζονται ή χρησιμοποιούν ένα soundgraph ονομάζεται *συνεδρία* (session). Σε μία συνεδρία μπορούν να συμμετέχουν ταυτόχρονα όλοι οι χρήστες που έχουν δικαίωμα να χρησιμοποιήσουν ή να επεξεργαστούν το soundgraph. Η επεξεργασία του soundgraph επιτυγχάνεται μέσω ενός μηχανισμού *ελέγχου του λόγου*, ο οποίος διαχειρίζεται τα αιτήματα ελέγχου του soundgraph επιτρέποντας σε ένα χρήστη κάθε φορά να έχει τον έλεγχο και να εφαρμόζει αλλαγές. Η αποστολή αιτημάτων ελέγχου του soundgraph, ενεργοποιείται με το πάτημα ενός ειδικού κουμπιού που βρίσκεται στη διεπαφή της εφαρμογής και ο χρήστης που αποκτά τον έλεγχο μπορεί να τον διατηρήσει για οσοδήποτε χρονικό διάστημα. Ο συντονισμός των χρηστών επιτυγχάνεται μέσω ενός συστήματος συνομιλίας.



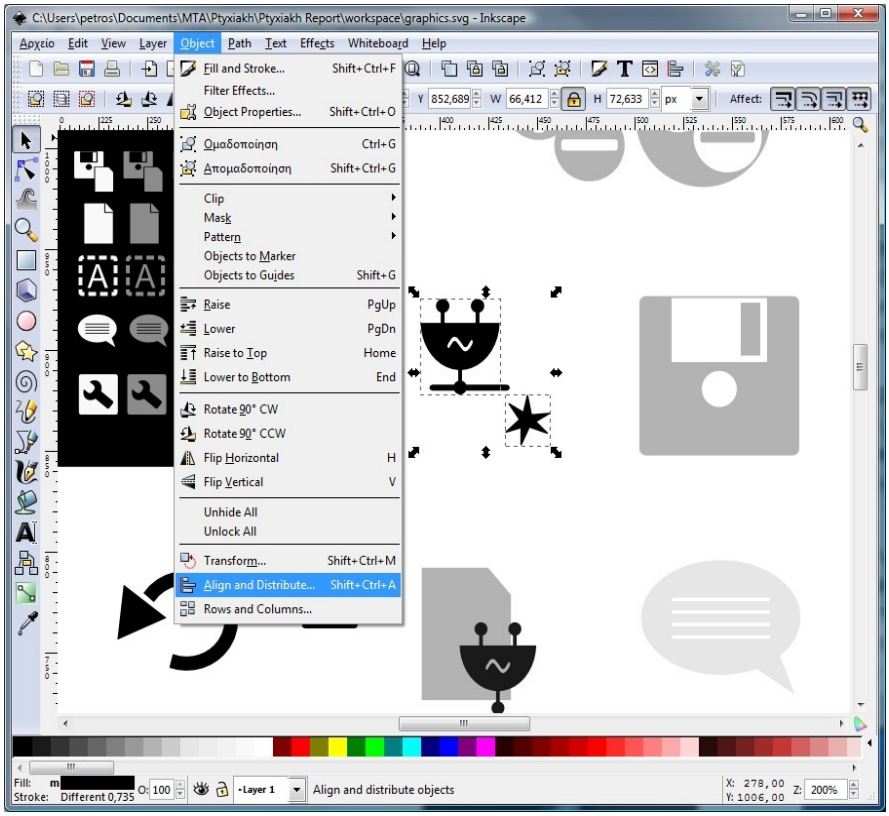
Εικόνα 8: Μια συνεδρία στο SharedEngine

2.3 Γραφικός Σχεδιασμός

Κατά την υλοποίηση του «SoundEngine» και του «SharedEngine» χρειάστηκε να κατασκευαστούν γραφικά αντικείμενα τα οποία ήταν απαραίτητα για τη λειτουργία των εφαρμογών καθώς επίσης σχεδιάστηκαν γραφικά με μοναδικό σκοπό την αισθητική βελτιστοποίηση των εφαρμογών (βλ. *Εικόνα 9*).

Αναλυτικότερα σχεδιάστηκαν:

- η επιφάνεια εργασίας του SoundEngine, δηλαδή ο χώρος στον οποίο αναπτύσσονται τα soundgraphs
- τα δομοστοιχεία
- οι θύρες επικοινωνίας των δομοστοιχείων
- το μενού εργαλείων του SoundEngine
- καθώς επίσης και όλα τα γραφικά του SharedEngine



Εικόνα 9: Στιγμιότυπο από τη σχεδίαση των γραφικών

3 Υλοποίηση

Στο κεφάλαιο αυτό θα παρουσιαστούν οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση των εφαρμογών καθώς επίσης θα περιγραφεί και η διαδικασία της υλοποίησης.

Για την ανάπτυξη του λογισμικού χρησιμοποιήθηκαν οι γλώσσες Java και JavaScript[1]. Η γλώσσα Java είναι μια υψηλού επιπέδου, αντικειμενοστραφής (object oriented) και interpreted γλώσσα προγραμματισμού. Η διερμηνεία του κώδικα σε γλώσσα μηχανής γίνεται από μια εφαρμογή διερμηνέα, η οποία λέγεται Java Virtual Machine (JVM). Η χρήση εικονικής μηχανής για τη διερμηνεία του κώδικα επιτρέπει την εκτέλεση του κώδικα σε οποιαδήποτε πλατφόρμα είναι εφοδιασμένη με ένα JVM, γεγονός που καθιστά τα προγράμματα σε γλώσσα Java ιδιαίτερα μεταφέρσιμα. Ένα επιπλέον χαρακτηριστικό της γλώσσας είναι ότι επιτρέπει την ενσωμάτωση Java εφαρμογών σε ιστοσελίδες. Οι ενσωματωμένες σε ιστοσελίδες εφαρμογές ονομάζονται *Applets*.

Η γλώσσα JavaScript όπως προμηνύει και το όνομά της είναι μια "scripting" γλώσσα, δηλαδή μια γλώσσα που χρησιμοποιείται για τον έλεγχο εφαρμογών από το χρήστη με προγραμματιστικό τρόπο [8]. Παρόλο που η γλώσσα JavaScript έχει παρόμοια σύνταξη με τη γλώσσα Java και το όνομα της θυμίζει τη Java, οι δύο γλώσσες δεν σχετίζονται μεταξύ τους.

Ένας από τους στόχους της εργασίας ήταν να συμβαδίσει με τις πρότυπες τεχνολογίες, καθώς κάτι τέτοιο βοηθάει στη δημιουργία στιβαρών εφαρμογών. Οι τεχνολογίες και οι βιβλιοθήκες που χρησιμοποιήθηκαν για την υλοποίηση των εφαρμογών είναι:

- το πρότυπο δημιουργίας περιγραφικών γλωσσών XML
- το πρότυπο περιγραφής XML εγγράφων Document Type Definition (DTD)
- το πρωτόκολλο ανταλλαγής δομημένων πληροφοριών μεταξύ εφαρμογών SOAP
- η περιγραφική γλώσσα HTML σε συνδυασμό με κώδικα JavaScript (δηλαδή Dynamic HTML)
- η τεχνολογία δυναμικής ανάπτυξης ιστοσελίδων JavaServer Pages (η αλλιώς JSP)
- η βιβλιοθήκη διαχείρισης OSC μυνημάτων "Java OSC"[2]
- και η βιβλιοθήκη "MySQL Connector/J"[3] που επιτρέπει την επικοινωνία με βάσεις δεδομένων

Τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση των εφαρμογών είναι:

- το Eclipse, ένα περιβάλλον ανάπτυξης εφαρμογών (IDE)
- το Notepad++, ένας επεξεργαστής κώδικα
- ο εξυπηρετητής Apache Tomcat
- το σύστημα διαχείρισης βάσεων δεδομένων MySQL
- το GIMP, μια εφαρμογή επεξεργασίας και σχεδιασμού γραφικών
- και το Inkscape, μια εφαρμογή σχεδιασμού διανυσματικών γραφικών (vector graphics)

Μπορεί εύκολα να διαπιστωθεί ότι όλα τα παραπάνω προγράμματα και προγραμματιστικά εργαλεία διατίθενται ελεύθερα, ενώ τα περισσότερα από αυτά είναι και "ανοικτού κώδικα" (open source). Η φιλοσοφία του λογισμικού ανοικτού κώδικα αποτελεί μια διαφορετική προσέγγιση στον τομέα

1 Εάν και γνωστή με το όνομα JavaScript, η τυποποιημένη έκδοση της γλώσσας ονομάζεται ECMAScript

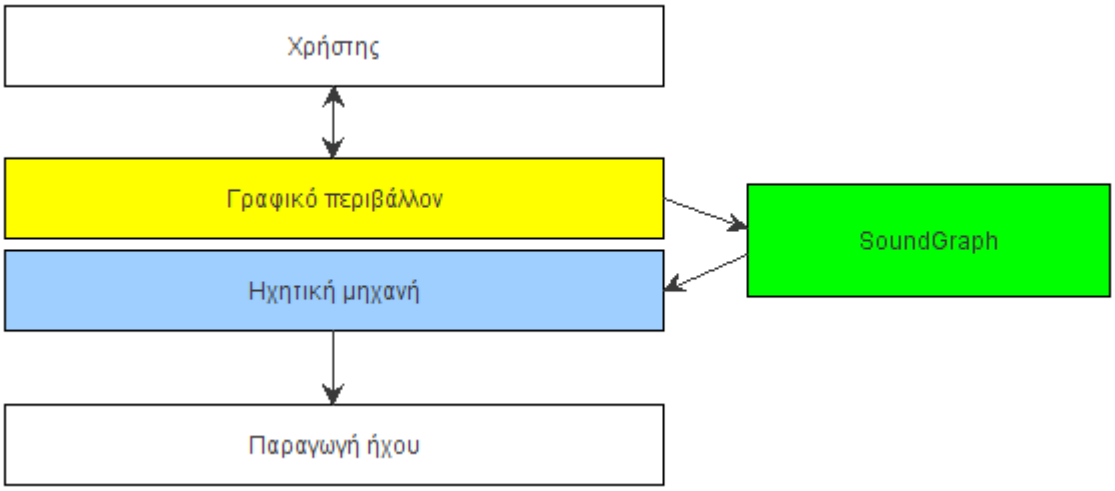
2 http://www.illposed.com/software/javaosc.html

3 http://www.mysql.com/products/connector/j/

της δημιουργίας εφαρμογών λογισμικού, η οποία αποκαλύπτει τα οφέλη της κοινωνικότητας και βοηθά στη διάδοση του πολιτισμού, προσφέροντας στην ανθρωπότητα νέα εργαλεία, καθώς και την τεχνογνωσία που συνοδεύει την ανάπτυξη τους.

3.1 Υλοποίηση του SoundEngine

Όπως φαίνεται και στο ακόλουθο σχήμα το SoundEngine χωρίζεται σε δύο βασικά επίπεδα (layers): το γραφικό περιβάλλον και την ηχητική μηχανή. Τα επίπεδα αυτά συνδέονται μεταξύ τους μέσω των soundgraphs, δηλαδή μέσω των ίδιων των αλγόριθμων σύνθεσης ήχου. Καθώς ο χρήστης μέσω του γραφικού περιβάλλοντος της εφαρμογής δημιουργεί αλγόριθμους σύνθεσης ήχου, αυτοί με τη μορφή αντικειμένων τύπου *SoundGraph* μεταφέρονται στην ηχητική μηχανή όπου και εκτελούνται.



Εικόνα 10: Η βασική δομή του SoundEngine

3.1.1 Το γραφικό περιβάλλον

Το γραφικό περιβάλλον αποτελεί μία γραφική διεπαφή (GUI) η οποία επιτρέπει την αλληλεπίδραση του χρήστη με την εφαρμογή. Για την υλοποίηση του γραφικού περιβάλλοντος χρησιμοποιήθηκαν έτοιμα γραφικά αντικείμενα καθώς επίσης όπως αναφέρθηκε σε προηγούμενο κεφάλαιο κατασκευάστηκαν και νέα. Αναλυτικότερα κατασκευάστηκαν:

- η επιφάνεια εργασίας, δηλαδή η επιφάνεια πάνω στην οποία αναπτύσσονται οι αλγόριθμοι
- η εργαλειοθήκη, η οποία περιλαμβάνει μεταξύ άλλων τα κουμπιά προσθήκης δομοστοιχείων στην επιφάνεια εργασίας καθώς και το κουμπί εκτέλεσης των αλγόριθμων
- τα κουμπιά της εργαλειοθήκης
- η γραφική αναπαράσταση των δομοστοιχείων
- και οι θύρες επικοινωνίας των δομοστοιχείων

Η υλοποίηση των γραφικών αντικειμένων χωρίζεται σε δύο μέρη: την υλοποίηση των γραφικών και την υλοποίηση των αντικειμένων σε προγραμματιστικό επίπεδο.

Η υλοποίηση των γραφικών πραγματοποιήθηκε με την εφαρμογή Inkscape, στην οποία σχεδιάστηκαν τα γραφικά και την εφαρμογή Gimp, η οποία χρησιμοποιήθηκε για την κοπή και την μετατροπή στο κατάλληλο μορφότυπο (format) των γραφικών.

Για την προγραμματιστική υλοποίηση των αντικειμένων χρησιμοποιήθηκαν ως βάση κλάσεις της γλώσσας Java όπως η κλάση *JComponent*, η οποία αποτελεί τη βάση για όλα τα γραφικά αντικείμενα του πακέτου[1] javax.swing .

3.1.2 Τα Soundgraphs

Για την διαχείριση των αλγόριθμων σύνθεσης ήχου υλοποιήθηκε η κλάση *SoundGraph*. Κάθε αντικείμενο τύπου SoundGraph περιγράφει έναν αλγόριθμο σύνθεσης ήχου και μέσω των κατάλληλων μεθόδων της κλάσης δίνεται η δυνατότητα μετατροπής των αλγόριθμων σύνθεσης ήχου σε ροές κειμένου[2] (text streams).

Οι ροές που δημιουργούνται από την κλάση έχουν τη μορφή XML εγγράφων και χρησιμοποιούνται τόσο για την αποθήκευση των αλγόριθμων σε κάποιο μέσο αποθήκευσης όσο και για τη δικτυακή τους μετάδοση (στην περίπτωση της δικτυακής συνεργατικής ανάπτυξης).

Για την αναπαράσταση των αλγόριθμων σε XML μορφή αναπτύχθηκε μια περιγραφική γλώσσα με την ονομασία “Soundgraph v1” η οποία ορίζεται από το ακόλουθο DTD[3].

```
<!ELEMENT soundgraph (spu | connection)*>
<!ATTLIST soundgraph
        version CDATA #IMPLIED
        samplerate (22050 | 44100 | 48000) #REQUIRED
>

<!ELEMENT spu (param*)>
<!ATTLIST spu
        class NMTOKEN #REQUIRED
        id ID #REQUIRED
        x CDATA #IMPLIED
        y CDATA #IMPLIED
>

<!ELEMENT param EMPTY>
<!ATTLIST param
        name CDATA #REQUIRED
        value CDATA #REQUIRED
>

<!ELEMENT connection EMPTY>
<!ATTLIST connection
        source IDREF #REQUIRED
        src-port CDATA #REQUIRED
        destination IDREF #REQUIRED
        dest-port CDATA #REQUIRED
>
```

Πλαίσιο 1: Το DTD της γλώσσας "Soundgraph v1"

1 Τα πακέτα στη γλώσσα Java αποτελούν έναν τρόπο ταξινόμησης και διαχωρισμού των κλάσεων.

2 Η διαδικασία αυτή ονομάζεται *serialization*.

3 Ένα DTD αποτελεί τον ορισμό μιας XML γλώσσας καθώς καθορίζει τη δομή και τα επιτρεπόμενα στοιχεία ενός XML εγγράφου.

Μεταξύ άλλων το DTD της γλώσσας “Soundgraph v1” ορίζει τους εξής κανόνες:

- το ριζικό στοιχείο (root element) ενός soundgraph είναι το στοιχείο <soundgraph>
- το στοιχείο soundgraph περιέχει ένα ή περισσότερα στοιχεία τύπου <spu> και <connection>
- τα στοιχεία τύπου <spu> περιέχουν υποχρεωτικά τις παραμέτρους “id” και “class”
- τα στοιχεία τύπου <connection> περιέχουν υποχρεωτικά τις παραμέτρους “source”, “src-port”, “destination” και “dest-port”

Στο *Πλαίσιο 2* παρατίθεται μια ενδεικτική αναπαράσταση ενός αλγόριθμου σύνθεσης ήχου με τη γλώσσα Soundgraph v1.

```
<soundgraph version="1.0" samplerate="44100">
 <spu id="obj0" class="gui.objects.Oscillator" x="298" y="215">
  <param name="object-color" value="0x808080"/>
  <param name="waveform" value="sine"/>
 </spu>
 <spu id="obj1" class="gui.objects.AudioOutputMono" x="322" y="261">
  <param name="object-color" value="0x808080"/>
  <param name="channels" value="mono"/>
 </spu>
 <spu id="obj2" class="gui.objects.NumberController" x="309" y="159">
  <param name="object-color" value="0x808080"/>
  <param name="controller-initial-value" value="0.0"/>
 </spu>
 <spu id="obj3" class="gui.objects.NumberController" x="391" y="159">
  <param name="object-color" value="0x808080"/>
  <param name="controller-initial-value" value="0.0"/>
 </spu>
 <connection source="obj0" src-port="0" destination="obj1" dest-port="0"/>
 <connection source="obj2" src-port="0" destination="obj0" dest-port="0"/>
 <connection source="obj3" src-port="0" destination="obj0" dest-port="1"/>
</soundgraph>
```

Πλαίσιο 2: Η αναπαράσταση ενός αλγόριθμου στην περιγραφική γλώσσα "Soundgraph v1"

3.1.3 Η ηχητική μηχανή

Η ηχητική μηχανή είναι υπεύθυνη για την εκτέλεση των αλγόριθμων σύνθεσης ήχου και αποτελείται από ένα πλήθος κλάσεων. Για να γίνει κατανοητή η λειτουργία της θα περιγραφούν τρεις βασικές κλάσεις: η κλάση *SoundEngine,* η κλάση *SignalProcessingUnit* και η κλάση *Connection*.

Η κλάση SignalProcessingUnit αποτελεί την υπερκλάση όλων των δομοστοιχείων της εφαρμογής και εφοδιάζει τα δομοστοιχεία με τις μεθόδους *process()* και *asynchronousEvent()*. Η μέθοδος *process()* κατά τη διάρκεια εκτέλεσης ενός αλγόριθμου καλείται συνεχόμενα και χρησιμεύει στην παραγωγή και την επεξεργασία σημάτων, αφού σε κάθε κλήση λαμβάνει και παράγει δείγματα σήματος. Η μέθοδος *asynchronousEvent()* χρησιμοποιείται για τη λήψη μηνυμάτων από άλλα δομοστοιχεία και αποτελεί ένα μέσο ελέγχου των παραμέτρων του δομοστοιχείου – παραλήπτη.

Η κλάση Connection περιγράφει όπως προμηνύει και το όνομά της μια σύνδεση μεταξύ των θυρών επικοινωνίας ενός δομοστοιχείου.

Η κλάση SoundEngine αποτελεί τον πυρήνα της μηχανής καθώς αναλαμβάνει τη διανομή των δειγμάτων του σήματος και τη διανομή των μηνυμάτων μεταξύ των δομοστοιχείων. Αναλυτικότερα η διαδικασία διανομής των δειγμάτων του σήματος πραγματοποιείται συνεχόμενα και περιλαμβάνει:

- κλήση της μεθόδου process() του δομοστοιχείου
- ανάκτηση των παραγόμενων δειγμάτων σήματος
- εντοπισμό των εξερχόμενων συνδέσεων του δομοστοιχείου
- δρομολόγηση των δειγμάτων στα δομοστοιχεία – παραλήπτες
- επανάληψη της διαδικασίας για κάθε δομοστοιχείο

Η διανομή των μηνυμάτων λόγο του μη προκαθορισμένου ρυθμού παραγωγής τους, δεν αποτελεί μέρος του παραπάνω κύκλου εργασίας. Η διαδικασία διανομής των μηνυμάτων περιλαμβάνει:

- λήψη του μηνύματος
- εντοπισμό του δομοστοιχείου – αποστολέα
- εντοπισμό των εξερχόμενων συνδέσεων του δομοστοιχείου – αποστολέα
- δρομολόγηση του μηνύματος στα δομοστοιχεία - παραλήπτες

3.2 Υλοποίηση του SharedEngine

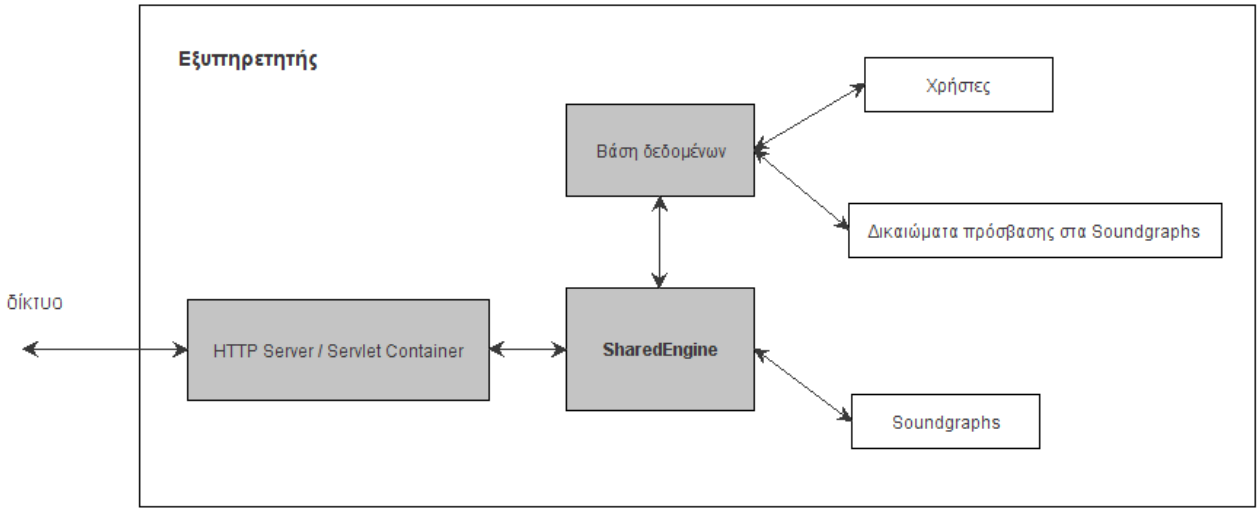
Η υλοποίηση του SharedEngine είναι αποτέλεσμα του συνδυασμού πολλών δικτυακών τεχνολογιών όπως είναι η γλώσσα *HTML*, το πρότυπο ανάπτυξης δυναμικών ιστοσελίδων *JSP* και το πρωτόκολλο *SOAP*. Όπως έχει ήδη αναφερθεί το SharedEngine είναι μια εφαρμογή ιστού, δηλαδή παρέχεται στο χρήστη σε μορφή ιστοσελίδων και η πρόσβαση σε αυτές γίνεται μέσω ενός κοινού φυλλομετρητή ιστοσελίδων (web browser).

3.2.1 Ο SharedEngine εξυπηρετητής

Εκτός από τον κώδικα που αναπτύχθηκε για την υλοποίηση του SharedEngine, για την λειτουργία του είναι απαραίτητη η ύπαρξη κάποιων εφαρμογών. Αναλυτικότερα το σύστημα – εξυπηρετητής (server) που θα παρέχει την εφαρμογή πρέπει να είναι εφοδιασμένο με ένα HTTP[1] εξυπηρετητή τύπου Servlet Container[2] και ένα σύστημα διαχείρισης βάσεων δεδομένων MySQL. Ο HTTP εξυπηρετητής αποτελεί το μέσω επικοινωνίας των χρηστών με την εφαρμογή, καθώς αναλαμβάνει τη μεταφορά των δεδομένων από το SharedEngine προς το χρήστη και αντίστροφα ενώ το σύστημα διαχείρισης βασεων δεδομένων χρησιμοποιείται για τη διαχείριση της βάσης δεδομένων της εφαρμογής (βλ. *Εικόνα 11*). Η βάση δεδομένων χρησιμοποιείται για την αποθήκευση των μελών και των σχέσεων τους με τα soundgraphs. Τα soundgraphs αποθηκεύονται στο δίσκο του εξυπηρετητή ως ξεχωριστά αρχεία με τη μορφή κώδικα γραμμένου στην περιγραφική γλώσσα “Soundgraph v1” που παρουσιάστηκε σε προηγούμενη ενότητα.

1 HTTP (Hypertext Tranfer Protocol) είναι το πρωτόκολλο πάνω στο οποίο στηρίζεται ο παγκόσμιος ιστός και χρησιμοποιείται για τη μεταφορά των ιστοσελίδων.

2 Αποτελεί μια εξειδικευμένη μορφή HTTP εξυπηρετητή η οποία επιτρέπει τη χρήση κώδικα γραμμένου σε γλώσσα Java με στόχο την παραγωγή δυναμικών ιστοσελίδων.



Εικόνα 11: Η δομή του εξυπηρετητή

3.2.2 Το SharedEngine

Η διεπαφή του SharedEngine υλοποιήθηκε με ένα συνδυασμό από HTML κώδικα, CSS κώδικα, server-side scripting και client-side scripting, . Με τον όρο server-side scripting εννοείται η δυναμική παραγωγή κώδικα HTML στην πλευρά του εξυπηρετητή, ενώ με τον όρο client-side scripting εννοείται η δυναμική παραγωγή HTML κώδικα στην πλευρά του χρήστη.

Το server-side scripting πραγματοποιήθηκε με την τεχνολογία JavaServer Pages (JSP), η οποία επιτρέπει την δημιουργία σελίδων από κώδικα γραμμένο σε γλώσσα Java. Οι JSP σελίδες που αναπτύχθηκαν επιτρέπουν στο χρήστη να διαχειριστεί τα soundgraphs, δηλαδή να περιηγηθεί ανάμεσα σε αυτά, να δημιουργήσει καινούργια, να διαγράψει τα ανεπιθύμητα καθώς και να τα προσπελάσει για χρήση ή επεξεργασία. Εκτός από τη δυναμική δημιουργία περιεχομένου στην πλευρά του εξυπηρετητή, προστέθηκε στις σελίδες και κώδικας σε γλώσσα JavaScript ο οποίος εκτελείται στην πλευρά του χρήστη (client-side scripting). Ο JavaScript κώδικας χρησιμοποιήθηκε κυρίως για να αποφευχθούν οι συνεχόμενες συναλλαγές με τον εξυπηρετητή οι οποίες επιβαρύνουν άσκοπα τον εξυπηρετητή, το δίκτυο και μπορεί να προκαλέσουν σύγχυση στο χρήστη.

Όσον αφορά τη δικτυακή επεξεργασία και χρήση των soundgraphs αυτό επιτυγχάνεται μέσω του SoundEngine. Αναλυτικότερα έγιναν κάποιες προσθήκες στον κώδικα του SoundEngine, έτσι ώστε να υποστηρίζει τις απαραίτητες για τη δικτυακή επεξεργασία λειτουργίες. Οι προσθήκες αυτές περιλαμβάνουν την ανάπτυξη μιας βιβλιοθήκης που καθιστά εφικτή την επικοινωνία με τον εξυπηρετητή καθώς και τον εμπλουτισμό της γραφικής διεπαφής με τα απαραίτητα για τη δικτυακή λειτουργία εργαλεία.

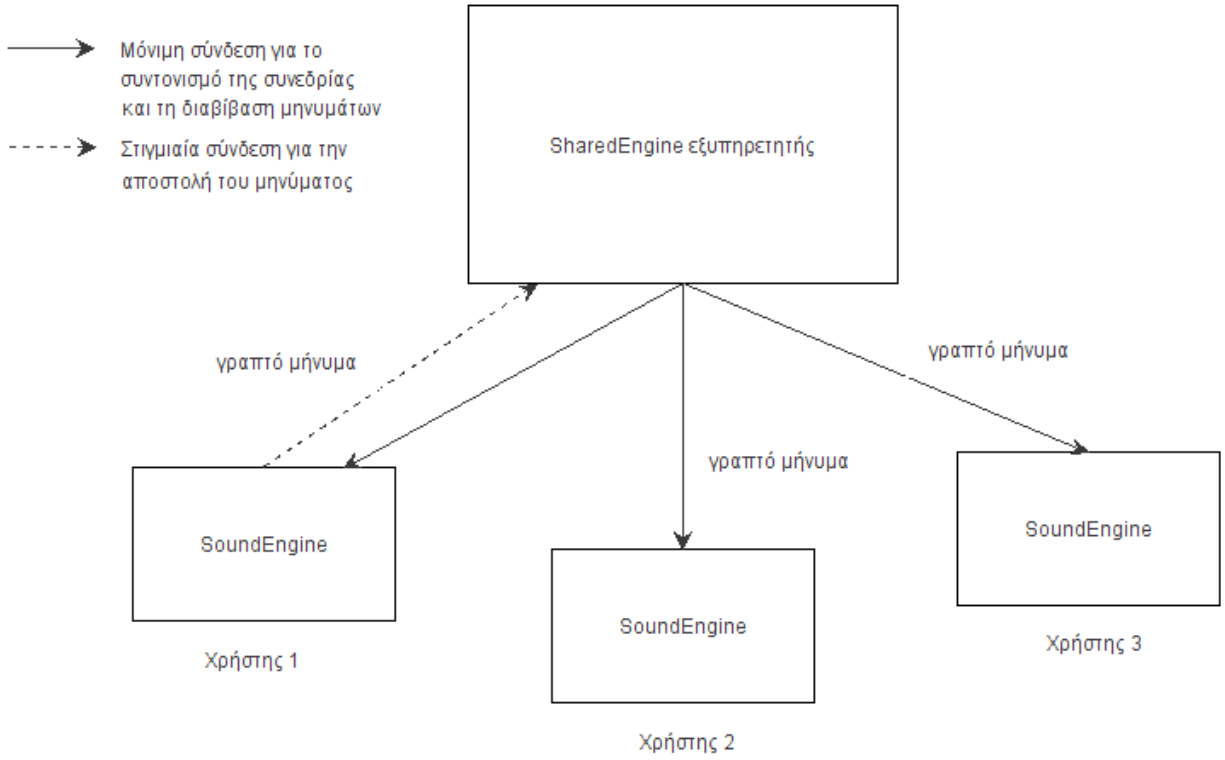
Εκτός από την βιβλιοθήκη που αναπτύχθηκε για την επικοινωνία του SoundEngine με τον εξυπηρετητή αναπτύχθηκε και η αντίστοιχη βιβλιοθήκη στην πλευρά του εξυπηρετητή, η οποία διαχειρίζεται τις συνεδρίες (sessions) και διεκπεραιώνει τα αιτήματα των χρηστών. Οι συνεδρίες αυτές περιλαμβάνουν συνεδρίες *μονού χρήστη* (single user), συνεδρίες *πολλαπλών χρηστών* (multi user) και οι χρήστες μπορεί να είναι δύο τύπων: editors ή viewers[1].

1 "Editors" είναι οι χρήστες οι οποίοι επιτρέπεται να τροποποιήσουν ένα soundgraph, ενώ "viewers" οι χρήστες που μπορούν μόνο να χρησιμοποιήσουν (να εκτελέσουν) ένα soundgraph.

3.2.3 Ο μηχανισμός συνεργατικής ανάπτυξης

Οι κύριες κλάσεις που σχετίζονται με την συνεργατική ανάπτυξη αλγόριθμων σύνθεσης ήχου είναι: η κλάση *DownloadSoundgraph* η οποία πραγματοποιεί την αποστολή των soundgraphs στους χρήστες και η κλάση *Server* η οποία συντονίζει τις συνεργατικές διαδικασίες μέσω ενός συστήματος *ελέγχου του λόγου* (floor control), καθώς επίσης διαβιβάζει γραπτά μηνύματα μεταξύ των χρηστών.

Όταν κάποιος χρήστης ζητήσει να ανοίξει ένα soundgraph, τότε το SoundEngine καταφορτώνει (downloads) το soundgraph και στη συνέχεια δημιουργεί μία μόνιμη σύνδεση με τον εξυπηρετητή μέσω της οποίας λαμβάνει γραπτά μηνύματα από τους άλλους χρήστες, καθώς επίσης ενημερώνεται για τις αλλαγές που συμβαίνουν στο soundgraph από τους άλλους χρήστες. Μόλις κάποιος εφαρμόσει μια αλλαγή τότε ειδοποιούνται όλοι οι χρήστες του soundgraph και το καταφορτώνουν ξανά. Για να μπορέσει κάποιος να επεξεργαστεί το soundgraph αποστέλει ένα αίτημα στον εξυπηρετητή, ο οποίος το "κλειδώνει" για τους άλλους χρήστες και του επιτρέπει την επεξεργασία. Αφού ολοκληρωθεί η επεξεργασία το τροποποιημένο soundgraph αποστέλλεται στον εξυπηρετητή και εφαρμόζονται οι αλλαγές.



Εικόνα 12: Αποστολή μηνύματος κατά τη διάρκεια μιας συνεδρίας

Στο ακόλουθο πλαίσιο παρουσιάζεται ο πυρήνας της κλάσης Server, ο οποίος αποτελείται από τις μεθόδους *doGet()* και *doPost()*. Οι μέθοδοι αυτές διαχειρίζονται τα δεδομένα που αποστέλλονται από την εφαρμογή-πελάτη (δηλαδή τη δικτυακή εκδοχή του SoundEngine) στον εξυπηρετητή κατά τη διάρκεια μιας συνεδρίας. Τα δεδομένα μεταφέρονται μέσω του πρωτοκόλλου HTTP και οι μέθοδοι *doGet()* και *doPost()* διαχειρίζονται τα αντίστοιχα μηνύματα *GET* και *POST* του πρωτοκόλλου.

Τα μηνύματα τύπου *GET* αποστέλλονται από την εφαρμογή-πελάτη στον εξυπηρετητή:

- για την διαβίβαση γραπτών μηνυμάτων στους άλλους χρήστες
- για να ζητηθεί το δικαίωμα επεξεργασίας του soundgraph
- και για να "απελευθερώθεί" το soundgraph έτσι ώστε να μπορέσει να το επεξεργαστεί κάποιος άλλος

Τα μηνύματα τύπου *POST* χρησιμοποιούνται από την εφαρμογή-πελάτη για την αποστολή των επεξεργασμένων soundgraphs στον εξυπηρετητή με σκοπό να καταχωρηθούν οι αλλαγές και να ειδοποιηθούν οι υπόλοιποι συμμετέχοντες στη συνεδρία.

```
	public void doGet(HttpServletRequest req, HttpServletResponse resp) {
		String action;
		String sessionid;
		String username=null;
		int soundgraphid;
		String message;
		SharedSession sharedsession;
		Contributor contributor=null;
		SharedSoundGraph sg;

		//get the sessionid
		sessionid=req.getParameter("session");
		if(sessionid==null) {
			sendBadRequest(resp,"Missing 'session' parameter");
			return;
		}

		//get the soundgraph
		try{

	soundgraphid=Integer.parseInt(req.getParameter("soundgraph"));
			sg=db.getSoundGraph(soundgraphid);
		}catch(Exception ex) {
			sendBadRequest(resp,"Invalid or missing 'soundgraph'
parameter");
			return;
		}
		if(sg==null) {
			sendBadRequest(resp,"Invalid or missing 'soundgraph'
parameter");
			return;
		}

		//get the action-command
		action=req.getParameter("action");
		if(action==null) {
			sendBadRequest(resp,"Missing 'action' parameter");
			return;
		}

		if(action.equalsIgnoreCase("eventlistener")) {

			//Identify new user
			try{
				username=db.getSessionUser(sessionid);
			}catch(Exception ex) {
				username=null;
			}

			try{
				//Check if this user has access to this soundgraph
				if(!db.hasAccess(sg, username)) {
					sendBadRequest(resp,"You can't access this
soundgraph");
					return;
				}

				//Create new contributor
				contributor=new
Contributor(sessionid,username,resp,db.canEdit(sg, username));
```

```
                        //Get (or create a new) the SharedSession for this
SoundGraph
                        sharedsession=getSharedSession(soundgraphid,true);

                  }catch(Exception ex) {
                        //Something went wrong
                        ex.printStackTrace();
                        return;
                  }

                  try{
                        //Add the contributor to this SharedSession
                        sharedsession.addContributor(contributor);

                  }catch(Exception ex) {
                        //Something went wrong
                        ex.printStackTrace();
                        //Try to remove the contributor from this SharedSession
                        try{
                              sharedsession.removeContributor(contributor);
                        }catch(Exception ex2) {ex2.printStackTrace();}
                        return;
                  }

                  //Retain the connection by blocking
                  try{
                        for(;;) {
                              //Rest for a while
                              Thread.sleep(10000);
                              //Send a 'connection-check' message
                              sharedsession.sendConnectionCheck(contributor);
                              //Check if the connection is still active

      if(contributor.getResponse().getWriter().checkError()) {

      sharedsession.removeContributor(contributor);
                                    break;
                              }
                        }
                  }catch(Exception ex) {
                        ex.printStackTrace();
                        try{
                              sharedsession.removeContributor(contributor);
                        }catch(Exception ex2) {ex2.printStackTrace();}
                  }

            } else {

                  try{
                        //Get the SharedSession for this SoundGraph
                        sharedsession=getSharedSession(soundgraphid,false);

                  }catch(Exception ex) {
                        sendBadRequest(resp, "There is no SharedSession for
this SoundGraph");
                        return;
                  }

                  //Identify user
```

```
                  contributor=sharedsession.getContributor(sessionid);
                  if(contributor==null) {
                        sendBadRequest(resp, "No such user for this
SharedSession");
                        return;
                  }

                  if(action.equalsIgnoreCase("message")) {
                        //Get the message
                        message=req.getParameter("message");
                        if(message==null) {
                              sendBadRequest(resp, "Missing 'message'
parameter");
                              return;
                        }

                        try{
                              //Propagate the message to the contributors
                              System.out.println("Sending..."+message);
                              sharedsession.sendMessage(message,
contributor.getUsername());
                        }catch(Exception ex) {
                              return;
                        }
                  } else if(action.equalsIgnoreCase("takecontrol")) {
                        try{
                              //Give the SoundGraph's control to contributor
                              if(!contributor.isEditor()) {
                                    sendBadRequest(resp, "You can't edit this
SoundGraph");
                                    return;
                              }
                              sharedsession.giveControlTo(contributor);

                        }catch(Exception ex) {
                              return;
                        }
                  } else if(action.equalsIgnoreCase("givecontrol")) {
                        try{
                              //Take the SoundGraph's control from contributor
                              sharedsession.takeControlFrom(contributor);
                        }catch(Exception ex) {
                              return;
                        }
                  }

            }

      }

      public void doPost(HttpServletRequest req, HttpServletResponse resp) {
            String sessionid;
            String action;
            int soundgraphid;
            SharedSession sharedsession;
            Contributor contributor=null;
            SharedSoundGraph sg;

            //get the sessionid
            sessionid=req.getParameter("session");
```

```
            if(sessionid==null) {
                  sendBadRequest(resp,"Missing 'session' parameter");
                  return;
            }

            //get the soundgraph
            try{

      soundgraphid=Integer.parseInt(req.getParameter("soundgraph"));
                  sg=db.getSoundGraph(soundgraphid);
            }catch(Exception ex) {
                  sendBadRequest(resp,"Invalid or missing 'soundgraph'
parameter");
                  return;
            }
            if(sg==null) {
                  sendBadRequest(resp,"Invalid or missing 'soundgraph'
parameter");
                  return;
            }

            //get the action-command
            action=req.getParameter("action");
            if(action==null) {
                  sendBadRequest(resp,"Missing 'action' parameter");
                  return;
            }

            try{
                  //Get the SharedSession for this SoundGraph
                  sharedsession=getSharedSession(soundgraphid,false);

            }catch(Exception ex) {
                  sendBadRequest(resp, "There is no SharedSession for this
SoundGraph");
                  return;
            }

            //Identify user
            contributor=sharedsession.getContributor(sessionid);
            if(contributor==null) {
                  sendBadRequest(resp, "No such user for this SharedSession");
                  return;
            }

            if(action.equalsIgnoreCase("commit")) {
                  if(!contributor.isEditor()) {
                        sendBadRequest(resp, "You can't edit this SoundGraph");
                        return;
                  }
                  try{
                        gr.teicrete.mta.soundengine.soundgraph.SoundGraph
soundgraph;
                        soundgraph=new
gr.teicrete.mta.soundengine.soundgraph.SoundGraph(req.getInputStream());
                        soundgraph.writeXML(System.out);
                        db.replaceSoundGraph(soundgraphid, soundgraph);
                        sharedsession.commit(contributor);
                  }catch(Exception ex) {
                        ex.printStackTrace();
```

```
                        return;
                    }
                }

            }
```

Πλαίσιο 3: Το τμήμα της κλάσης Server που είναι υπεύθυνο για τη διαχείριση των συνεδριών (sessions)

4 Επίλογος

4.1 Επιδόσεις

Όσον αφορά τις επιδόσεις του SoundEngine παρατηρήθηκαν φαινόμενα χρονικής καθυστέρησης κατά την διανομή μηνυμάτων ελέγχου, τα οποία καθιστούν δύσκολή έως και ανέφικτη τη χρήση της εφαρμογής για τη ζωντανή εκτέλεση μουσικής. Εάν και κάτι τέτοιο μοιάζει περιοριστικό, οι χρήσεις της εφαρμογής δεν περιορίζονται στη ζωντανή εκτέλεση μουσικής. Μεταξύ άλλων το SoundEngine μπορεί να χρησιμοποιηθεί για την ανάπτυξη αλγόριθμων σύνθεσης ήχου, την επίδειξη τεχνικών σύνθεσης ήχου καθώς επίσης τη διδασκαλία και την εκμάθηση τεχνικών σύνθεσης ήχου.

Όσον αφορά τις επιδόσεις του SharedEngine, το σημαντικότερο χαρακτηριστικό της εφαρμογής είναι ο αριθμός των χρηστών που μπορεί να εξυπηρετήσει ταυτόχρονα. Αν και η εφαρμογή έχει υλοποιηθεί με πρόνοια κλιμάκωσης (scalability), ώστε δηλαδή να υποστηρίζει απεριόριστο αριθμό χρηστών, στην πράξη ο αριθμός αυτός περιορίζεται από τα τεχνικά χαρακτηριστικά του υπολογιστή στον οποίο εκτελείται η εφαρμογή και από τα τεχνικά χαρακτηριστικά του δικτύου στο οποίο στηρίζεται η επικοινωνία. Δηλαδή όσο πιο “δυνατό” είναι το σύστημα του εξυπηρετητή και όσο μεγαλύτερο το εύρος ζώνης (bandwidth) του δικτύου, τόσο περισσότεροι χρήστες θα μπορούν να χρησιμοποιήσουν την εφαρμογή.

4.2 Δυνατότητες βελτίωσης και μελλοντικές προοπτικές

Η υλοποίηση του γραφικού περιβάλλοντος συνεργατικής ανάπτυξης αλγόριθμων σύνθεσης ήχου επικεντρώθηκε στην ανάπτυξη της βέλτιστης ηχητικής μηχανής, όσον αφορά το SoundEngine και στην ανάπτυξη του μηχανισμού διαμοιρασμού και ταυτόχρονης χρήσης των soundgraphs, όσον αφορά το SharedEngine. Καθώς ο στόχος της εργασίας ήταν η διερεύνηση του τρόπου κατασκευής και η κατασκευή των κεντρικών μηχανισμών του περιβάλλοντος, οι λειτουργίες και τα χαρακτηριστικά τα οποία υλοποιήθηκαν είναι ενδεικτικά.

Μια εκτεταμένη υλοποίηση δομοστοιχείων για τη κατασκευή αλγόριθμων σύνθεσης ήχου ήταν πέρα από τους στόχους της παρούσας εργασίας. Εκτός από δομοστοιχεία, θα μπορούσαν να υλοποιηθούν και μηχανισμοί σχετικοί με τη λειτουργικότητα της εφαρμογής, όπως ένας μηχανισμός αναίρεσης των τελευταίων τροποποιήσεων ενός soundgraph (λειτουργία “undo”).

Η σημαντικότερη προσθήκη που θα μπορούσε να γίνει στο SharedEngine αφορά την ασφάλεια του συστήματος και συγκεκριμένα την αποτροπή ανεξέλεγκτης εγγραφής νέων μελών. Η εγγραφή ενός νέου μέλους μπορεί να γίνει μηχανικά, με αποτέλεσμα μια εφαρμογή παραγωγής πλαστών μελών να μπορεί να γεμίσει ολοκληρωτικά τον αποθηκευτικό χώρο του εξυπηρετητή, γεγονός που θα καθιστούσε αδύνατη τη λειτουργία της εφαρμογής.

4.3 Συμπεράσματα

Στην παρούσα εργασία παρουσιάστηκαν τα βασικότερα μέρη της διαδικασίας σχεδιασμού και υλοποίησης ενός περιβάλλοντος δικτυακής-συνεργατικής ανάπτυξης αλγόριθμων σύνθεσης ήχου. Η ανάπτυξη του περιβάλλοντος χωρίστηκε στην ανάπτυξη δύο επιμέρους εφαρμογών: στο SoundEngine και στο SharedEngine. Το SoundEngine αποτελεί μια εφαρμογή ανάπτυξης και επεξεργασίας αλγόριθμων σύνθεσης ήχου και το SharedEngine αποτελεί μια εφαρμογή ιστού, η οποία ενσωματώνοντας το SoundEngine παρέχει υπηρεσίες συνεργατικής ανάπτυξης αλγόριθμων σύνθεσης ήχου.

Όπως διαπιστώθηκε κατά τη διεξαγωγή της εργασίας, η υλοποίηση ενός συστήματος δικτυακής-

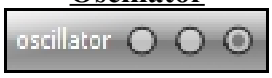
συνεργατικής ανάπτυξης αλγόριθμων σύνθεσης ήχου αποτελεί διεπιστημονική διαδικασία καθώς συνδυάζει καταγεγραμμένη γνώση από διαφορετικά πεδία, όπως τη μουσική τεχνολογία, τον προγραμματισμό εφαρμογών λογισμικού, τα δίκτυα Η/Υ και την αλληλεπίδραση ανθρώπου-μηχανής. Παρόλες όμως τις δυσκολίες του εγχειρήματος, όπως αποδεικνύεται με τις εφαρμογές SoundEngine και SharedEngine, η ανάπτυξη ενός τέτοιου περιβάλλοντος είναι εφικτή.

Παράρτημα Α: Τα βασικά χαρακτηριστικά των δομοστοιχείων

Σε αυτή την ενότητα παρουσιάζονται τα χαρακτηριστικά των δομοστοιχείων του SoundEngine.

Σημείωση: Στους πίνακες που συνοδεύουν τα δομοστοιχεία οι θύρες αναφέρονται με σειρά από τα αριστερά προς τα δεξιά.

Oscillator



Παράγει ημιτονοειδής, τριγωνικές και τετραγωνικές ταλαντώσεις.

Θύρα	**Τύπος**	**Δεδομένα**	**Τύπος μηνύματος**	**Σχόλιο**
1	Είσοδος	Σήμα/Μήνυμα	float	Ορίζει το πλάτος ταλάντωσης
2	Είσοδος	Σήμα/Μήνυμα	float	Ορίζει τη συχνότητα ταλάντωσης
3	Έξοδος	Σήμα	-	-

Delay

Μονάδα καθυστέρησης σήματος.

Θύρα	**Τύπος**	**Δεδομένα**	**Τύπος μηνύματος**	**Σχόλιο**
1	Είσοδος	Σήμα	-	Δέχεται το σήμα που θα καθυστερήσει
2	Είσοδος	Σήμα/Μήνυμα	float	Δέχεται το χρόνο καθυστέρησης σε δείγματα
3	'Εξοδος	Σήμα	-	Εξάγει το καθυστερημένο σήμα

Noise

Γεννήτρια λευκού θορύβου.

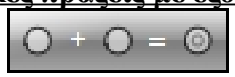
Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	'Εξοδος	Σήμα	-	Εξάγει λευκό θόρυβο

Μαθηματικές πράξεις

Πρόσθεση, αφαίρεση, πολλαπλασιασμός και διαίρεση.

Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Είσοδος	Σήμα/Μήνυμα	float	-
2	Είσοδος	Σήμα/Μήνυμα	float	-
3	'Εξοδος	Σήμα	-	-

Μαθηματικές πράξεις με έξοδο Μήνυμα



Πρόσθεση, αφαίρεση, πολλαπλασιασμός και διαίρεση.

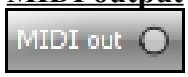
Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Είσοδος	Μήνυμα	float	-
2	Είσοδος	Μήνυμα	float	-
3	'Εξοδος	Μήνυμα	float	-

MIDI input

Εξάγει MIDI μηνύματα που προέρχονται από κάποια MIDI συσκευή.

Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Έξοδος	Μήνυμα	MIDI	-

MIDI output



Μεταφέρει MIDI μηνύματα σε κάποια MIDI συσκευή.

Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Είσοδος	Μήνυμα	MIDI	-

MIDI channel filter

Ελέγχει τα εισερχόμενα μηνύματα και εξάγει αυτά που ανήκουν στο επιλεγμένο κανάλι.

Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Είσοδος	Μήνυμα	MIDI	-
2	Έξοδος	Μήνυμα	MIDI	Εξάγει τα μηνύματα που ανήκουν στο επιλεγμένο κανάλι

MIDI controller filter

Ελέγχει εάν τα εισερχόμενα μηνύματα είναι “Control Change” μηνύματα και για αυτά που ανήκουν στον επιλεγμένο ελεγκτή, εξάγει την τιμή τους.

Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Είσοδος	Μήνυμα	MIDI	-
2	Έξοδος	Μήνυμα	float	Εξάγει την τιμή του controller

OSC receiver

Εξάγει τα OSC μηνύματα που καταφθάνουν στην επιλεγμένη δικτυακή θύρα.

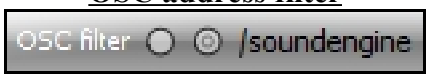
Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Έξοδος	Μήνυμα	OSC	-

OSC transmitter

Δέχεται δεκαδικούς αριθμούς και τους αποστέλλει με OSC μηνύματα στον επιλεγμένο δικτυακό παραλήπτη.

Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Είσοδος	Μήνυμα	float	-

OSC address filter



Δέχεται OSC μηνύματα και για αυτά που απευθύνονται στην επιλεγμένη OSC διεύθυνση (OSC address), εξάγει την τιμή τους.

Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Είσοδος	Μήνυμα	OSC	-
2	'Εξοδος	Μήνυμα	float	-

Number Controller

Εξάγει αριθμούς όταν γίνει “drag” με το ποντίκι η τιμή του controller και χρησιμεύει για τον έλεγχο δομοστοιχείων. Μια άλλη χρήση του είναι η παρακολούθηση τιμών, αφού όταν παραλάβει έναν αριθμό στην είσοδο του, τότε τον εμφανίζει.

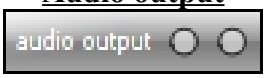
Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Είσοδος	Μήνυμα	float	Εμφανίζει τις εισερχόμενες τιμές
2	Έξοδος	Μήνυμα	float	Εξάγει την τιμή του controller όταν αυτή αλλάξει

Audio input

Καταγράφει σήμα από την επιλεγμένη είσοδο της κάρτας ήχου και το εξάγει.

Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Έξοδος	Σήμα	-	-

Audio output



Προωθεί στερεοφωνικό σήμα στη έξοδο της κάρτας ήχου.

Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Είσοδος	Σήμα	-	Σήμα αριστερού καναλιού
2	Είσοδος	Σήμα	-	Σήμα δεξιού καναλιού

Audio output (mono)

Προωθεί μονοφωνικό σήμα στη έξοδο της κάρτας ήχου.

Θύρα	Τύπος	Δεδομένα	Τύπος μηνύματος	Σχόλιο
1	Είσοδος	Σήμα	-	-

Comment

Το αντικείμενο αυτό χρησιμεύει στη συγγραφή σχολίων και δεν θύρες επικοινωνίας.

Παράρτημα Β: Οι κλάσεις του SoundEngine και του SharedEngine

Καθώς οι δύο εφαρμογές για τη λειτουργία τους χρησιμοποιούν κοινές κλάσεις, δεν γίνεται διαχωρισμός των κλάσεων ανά εφαρμογή.

Σημείωση: Από τα πακέτα έχει παραληφθεί το πρόθεμα “gr.teicrete.mta”.

Πακέτο	**Κλάσεις & *Διεπαφές***
soundengine	Applet StandAloneApp
soundengine.core	Connection Message SignalProcessingUnit SoundEngine *AsynchronousEventListener*
soundengine.core.spu.general	AudioInput AudioOutput AudioOutputStereo Delay Noise NullSPU NumberController Oscillator
soundengine.core.spu.math	Addition Division Multiplication Subtraction
soundengine.core.spu.midi	ChannelFilter ControllerFilter MidiInput MidiOutput
soundengine.core.spu.osc	AddressFilter Receiver Transmitter

Πακέτο	**Κλάσεις & *Διεπαφές***
soundengine.gui	AnimationBox Button ConversationWindow FlashingButton Label MessageList Panel ResourceLoader SoundEnginePanel SoundGraphEditor StartButton ToggleButton *PropertyListener*
soundengine.gui.spu	AudioInput AudioOutputMono AudioOutputStereo Comment Delay GenericObject MidiChannelFilter MidiControllerFilter MidiInput MidiOuput MsgAddition MsgDivision MsgMultiplication MsgSubtraction Noise NumberController OSCAddressFilter Oscillator OSCReveiver OSCTransmitter SigAddition SigDivision SigMultiplication SigSubtraction
soundengine.gui.spu.ports	Input Output
soundengine.soundgraph	HtmlEntityEncoder SoundGraph

Πακέτο	**Κλάσεις & *Διεπαφές***
sharedengine.client	Client MultipartReader *RemoteEventListener*
sharedengine.client.soap	Message
sharedengine.server	DownloadSoundgraph Setup
sharedengine.server.database	Database SharedSoundGraph
sharedengine.server.sharedsession	Contributor Server SharedSession
sharedengine.server.sharedsession.soap	Message

Παράρτημα Γ: Απαιτήσεις συστήματος και οδηγός εγκατάστασης

1 Εγκατάσταση της εφαρμογής SoundEngine

Το SoundEngine αποτελεί μια ατομική εφαρμογή τοπικής εμβέλειας, που χρησιμεύει για την επεξεργασία αλγόριθμων σύνθεσης ήχου.

Σημείωση: Η δικτυακή εκδοχή του SoundEngine συμπεριλαμβάνεται στο πακέτο του SharedEngine εξυπηρετητή και παρέχεται αυτόματα στους χρήστες κατά τη χρήση της εφαρμογής, χωρίς να χρειάζεται εγκατάσταση ειδικού λογισμικού στον υπολογιστή του χρήστη. Για πληροφορίες σχετικά με την εγκατάσταση του SharedEngine εξυπηρετητή βλέπε παρακάτω.

Απαιτήσεις συστήματος:

- Επεξεργαστής: 2.6GHz Pentium 4 (ή καλύτερος)
- Μνήμη RAM: 512MB (ή περισσότερο)
- Κάρτα ήχου με υποστήριξη εγγραφής/αναπαραγωγής με ανάλυση 16bit – 44.1KHz (ή περισσότερο)
- 2MB ελεύθερος χώρος στο σκληρό δίσκο[1]
- Java SE Runtime Environment (JRE) 6 ή Java SE Development Kit (JDK) 6

Εγκατάσταση από το επισυναπτόμενο CD:

1. **Εγκατάσταση του JRE 6 (εάν δεν είναι ήδη εγκατεστημένο στον υπολογιστή)**
   Για την εγκατάσταση του JRE 6 εκτελούμε την εφαρμογή εγκατάστασης "jre-6-windows-i586.exe" που βρίσκεται στο φάκελο "Applications\JRE 6" του CD και ακολουθούμε τα βήματα μέχρι να ολοκλρωθεί η εγκατάσταση.
2. **Εγκατάσταση του SoundEngine**
   Για να εγκαταστήσουμε το SoundEngine εκτελούμε την εφαρμογή εγκατάστασης "SoundEngine v1.00 Installer.exe" που βρίσκεται στο φάκελο "Applications\SoundEngine" του CD και ακολουθούμε τα βήματα μέχρι να ολοκληρωθεί η εγκατάσταση.

Αφού ολοκληρωθεί η εγκατάσταση η εφαρμογή μπορεί να εκτελεστεί από το μενού "Έναρξη" επιλέγοντας το στοιχείο "Όλα τα προγράμματα > SoundEngine > soundengine".

2 Εγκατάσταση του SharedEngine εξυπηρετητή

Απαιτήσεις συστήματος:

- Επεξεργαστης: 2.6GHz Pentium 4 (ή καλύτερος)
- Μνήμη RAM: 1GB (ή περισσότερο)
- 25MB ελεύθερος χώρος στο σκληρό δίσκο[2]

---

1 Στο χώρο αυτό δεν συνυπολογίζεται ο χώρος που χρειάζεται για την εγκατάσταση του JRE.

2 Στο χώρο αυτό δεν συνυπολογίζεται ο χώρος που χρειάζεται για την εγκατάσταση των προαπαιτούμενων εφαρμογών (JRE, Tomcat, MySQL κτλ.)

- Java SE Runtime Environment (JRE) 6 ή Java SE Development Kit (JDK) 6
- Apache Tomcat
- MySQL
- Web browser (π.χ. Firefox)

Εγκατάσταση από το επισυναπτόμενο CD:

1. **Εγκατάσταση του JRE 6 (εάν δεν είναι ήδη εγκατεστημένο στον υπολογιστή)**
   Για την εγκατάσταση του JRE 6 εκτελούμε την εφαρμογή εγκατάστασης "jre-6-windows-i586.exe" που βρίσκεται στο φάκελο "Applications\JRE 6" του CD και ακολουθούμε τα βήματα μέχρι να ολοκλρωθεί η εγκατάσταση.
2. **Εγκατάσταση του εξυπηρετητή MySQL (εάν δεν είναι ήδη εγκατεστημένος στον υπολογιστή)**
   Για να εγκαταστήσουμε ο εξυπηρετητής MySQL αποσυμπιέζουμε το αρχείο "mysql-5.1.30-win32.zip" που βρίσκεται στο φάκελο "Applications\MySQL" του CD και εκτελούμε την περιεχόμενη εφαρμογή εγκατάστασης "Setup.exe". Στη συνέχεια ακολουθούμε τα βήματα μέχρι να ολοκληρωθεί η εγκατάσταση.
3. **Εγκατάσταση του εξυπηρετητή Apache Tomcat (εάν δεν είναι ήδη εγκατεστημένος στον υπολογιστή)**
   Για την εγκατάσταση του εξυπηρετητή Apache Tomcat εκτελούμε την εφαρμογή εγκατάστασης "apache-tomcat-6.0.16.exe" που βρίσκεται στο φάκελο "Applications\Apache Tomcat" του CD και ακολουθούμε τα βήματα μέχρι να ολοκληρωθεί η εγκατάσταση.
4. **Εγκατάσταση του SharedEngine**
   Για την εγκατάσταση του SharedEngine εκτελούμε την εφαρμογή εγκατάστασης "SharedEngine v1.01 Installer.exe" που βρίσκεται στο φάκελο "Applications\SharedEngine" του CD και ακολουθούμε τα βήματα του οδηγού εγκατάστασης.
   Προσοχή! Στη σελίδα επιλογής της διεύθυνσης εγκατάστασης επιλέγουμε τη διεύθυνση που εγκαταστάθηκε ο εξυπηρετητής Apache Tomcat (π.χ. C:\Program Files\Tomcat 6.0\).
   Αφού ολοκληρωθεί ο οδηγός εγκατάστασης ανοίγουμε τη διεύθυνση http://HOST[1]/sharedengine με έναν φυλλομετρητή ιστοσελίδων και πραγματοποιούμε κάποιες απαραίτητες ρυθμίσεις.

Μετά την εφαρμογή των ρυθμίσεων η εγκατάσταση ολοκληρώνεται και το SharedEngine είναι προσβάσιμο στους χρήστες μέσω της διεύθυνσης http://HOST[1]/sharedengine .

1 Όπου HOST η διεύθυνση ή το όνομα του υπολογιστή στον οποίο εγκαταστάθηκε ο εξυπηρετητής Apache Tomcat και το SharedEngine.

Βιβλιογραφικές Αναφορές

[1] Modular synthesizer – Wikipedia, the free encyclopedia,
http://en.wikipedia.org/wiki/Modular_synthesizer

[2] Computer network – Wikipedia, the free encyclopedia,
http://en.wikipedia.org/wiki/Computer_network

[3] Collaborative software – Wikipedia, the free encyclopedia,
http://en.wikipedia.org/wiki/Collaborative_software

[4] Groupware and Social Dynamics: Eight Challenges for Developers
http://research.microsoft.com/~jgrudin/past/Papers/CACM94/cacm94.html

[5] Computer supported cooperative work – Wikipedia, the free encyclopedia,
http://en.wikipedia.org/wiki/CSCW

[6] H.-P. Dommel and J.J. Garcia-Luna-Aceves, "Floor Control for Multimedia Conferencing and Collaboration", *Multimedia Systems* (ACM/Springer), Vol. 5, No. 1, January 1997.

[7] A. Barbosa, "Displaced Soundscapes: A Survey of Network Systems for Music and Sonic Art Creation", *LEONARDO MUSIC JOURNAL*, Vol. 13, pp. 53-59, 2003

[8] Scripting language – Wikipedia, the free encyclopedia,
http://en.wikipedia.org/wiki/Scripting_language