**Ιστορική αναδρομή**

Η παρούσα εργασία παρέχει μια ιστορική ανασκόπηση για την εξέλιξη του μουσικού προγραμματισμού, όπως επίσης και για το πώς επηρεάζεται ο τρόπος με τον οποίο οι συνθέτες εργάζονται με τους υπολογιστές. Εξετάζονται τα πρώιμα εργαλεία προγραμματισμού για τον ήχο, και η άνοδος γλωσσών ειδικού πεδίου (domain-specific) για μουσική υπολογιστών, από τις Csound και Max/MSP, μέχρι και πιο πρόσφατες, όπως οι SuperCollider και ChucK.[[1]](#endnote-1)

O υπολογιστής θεωρείται εδώ και καιρό ένα εξαιρετικά ελκυστικό εργαλείο για τη δημιουργία και τον χειρισμό του ήχου. Η ακρίβεια και οι πάμπολλες δυνατότητές του το καθιστούν μια συναρπαστική πλατφόρμα για τον πειραματισμό και τη δημιουργία μουσικής - αλλά μόνο στο βαθμό που μπορούμε πραγματικά να πούμε σε έναν υπολογιστή τι να κάνει και πώς.

Ένα πρόγραμμα είναι μια ακολουθία οδηγιών για έναν υπολογιστή. Μια γλώσσα προγραμματισμού είναι μια συλλογή συντακτικών και σημασιολογικών κανόνων για τον προσδιορισμό αυτών των οδηγιών, και τελικά για την παροχή μετάφρασης από προγράμματα γραμμένα από ανθρώπους στις αντίστοιχες οδηγίες που εκτελούν οι υπολογιστές. Αποδεικνύεται, λοιπόν, ότι οι γλώσσες προγραμματισμού είναι οι πλέον θεμελιώδεις και διαρκείς στον χρόνο, όσον αφορά τις διεπαφές ανθρώπων-υπολογιστή. Αυτό διότι μπορεί να μην εκτελούν άμεσα συγκεκριμένες εργασίες (όπως επεξεργασία κειμένου ή επεξεργασία βίντεο), αλλά επιτρέπουν να δημιουργήσουμε λογισμικά που θα μπορούσαν να εκτελέσουν σχεδόν οποιαδήποτε προσαρμοσμένη λειτουργία. Η γλώσσα προγραμματισμού λειτουργεί ως μεσολαβητής μεταξύ της ανθρώπινης πρόθεσης και των αντίστοιχων οδηγιών που έχουν νόημα για έναν υπολογιστή. Είναι το πιο γενικό και συνάμα το πιο οικείο και ακριβές εργαλείο για να δίνουμε εντολές σε υπολογιστές.

**ΠΡΙΝ ΤΟΥΣ ΥΠΟΛΟΓΙΣΤΕΣ [6]**

Η ιδέα του προγραμματισμού υπολογιστικών αυτομάτων προς τη δημιουργία μουσικής μπορεί να ανιχνευθεί ήδη από το 1843, όταν η Λαίδη Ada Lovelace (που εργαζόταν με τον Charles Babbage) μίλησε για τις εφαρμογές του θεωρητικού Αναλυτικού Κινητήρα, διαδόχου της γνωστής Μηχανής Διαφορών του Babbage. Η αρχική Μηχανή Διαφοράς ήταν κυρίως μια "υπολογιστική μηχανή", ενώ ο Αναλυτικός Κινητήρας (ο οποίος ποτέ δεν κατασκευάστηκε) επρόκειτο να περιέχει μηχανισμούς λήψης αποφάσεων και βρόχου, αμφότεροι θεμελιώδεις για την ικανότητα προγραμματισμού (programmability). Η Lady Lovelace σωστά θεωρούσε τον Αναλυτικό Κινητήρα ως έναν υπολογιστή γενικής χρήσης, κατάλληλο για "ανάπτυξη [sic] και χαρτογράφηση οποιασδήποτε συνάρτησης... ο κινητήρας είναι η υλική έκφραση οποιασδήποτε αόριστης λειτουργίας οποιουδήποτε βαθμού γενικότητας και πολυπλοκότητας." Προέβλεψε περαιτέρω τα εξής: "Υποθέτοντας, παραδείγματος χάριν, ότι οι θεμελιώδεις σχέσεις των τονικών ήχων στην επιστήμη της αρμονίας και της μουσικής σύνθεσης υπόκειντο τέτοιου είδους έκφραση και προσαρμογές, ο κινητήρας θα μπορούσε να συνθέσει περίτεχνα και επιστημονικά κομμάτια μουσικής οποιουδήποτε βαθμού πολυπλοκότητας ή έκτασης."

Η πρόβλεψη της Lady Lovelace έγινε περισσότερο από εκατό χρόνια πριν από τον πρώτο ήχο που παρήγαγε ο υπολογιστής. Ωστόσο, οι ημι-προγραμματιζόμενες μηχανές παραγωγής μουσικής εμφανίστηκαν, με διάφορες μορφές, πριν από την υλοποίηση ενός πρακτικού υπολογιστή. Για παράδειγμα, το πιάνο αναπαραγωγής (player piano), που διαδόθηκε στις αρχές του 20ου αιώνα, ήταν ένα ενισχυμένο πιάνο που "έπαιζε τον εαυτό του", σύμφωνα με κυλίνδρους χαρτιού με διατρήσεις, που αντιπροσώπευαν τα μοτίβα που επρόκειτο να παιχτούν. Αυτοί οι εναλλάξιμοι κύλινδροι πιάνου μπορούν να θεωρηθούν απλά προγράμματα, που καθορίζουν ρητά τις μουσικές παρτιτούρες.

Καθώς η ηλεκτρονική μουσική εξελίχθηκε, κέρδισαν σε δημοτικότητα και οι αναλογικοί συνθετητές (γύρω στη δεκαετία του ’60), οι οποίοι υποστήριζαν τις διασυνδεόμενες και εναλλάξιμες μονάδες επεξεργασίας ήχου. Υπάρχει ένα ορισμένο επίπεδο ικανότητας προγραμματισμού που εμπλέκεται, και αυτό το πρότυπο βασισμένο σε μπλοκ επηρέασε τον μεταγενέστερο σχεδιασμό συστημάτων ψηφιακής σύνθεσης.

Με την έλευση της ψηφιακής εποχής, οι υπολογιστές αυξάνονται σε υπολογιστική ισχύ και αποθήκευση, και οι γλώσσες προγραμματισμού τείνουν να γίνονται ολοένα και πιο υψηλού επιπέδου, αντλώντας περισσότερες λεπτομέρειες για το υποκείμενο σύστημα. Το γεγονός αυτό επηρέασε σημαντικά τον τρόπο με τον οποίο προγραμματίζουμε τη μουσική.

**Η ΕΠΟΧΗ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ:** ΠΡΩΙΜΕΣ ΓΛΩΣΣΕΣ ΚΑΙ Η ΑΝΟΔΟΣ ΤΩΝ MUSIC-N

Η πρώτη εποχή των συστημάτων προγραμματισμού μουσικής βασισμένων σε υπολογιστή, συνέβη παράλληλα με αυτή των κεντρικών υπολογιστών (mainframes) και την αρχή των προσωπικών σταθμών εργασίας στα μέσα της δεκαετίας του '70. Οι κεντρικοί υπολογιστές ήταν γιγαντιαίοι, συχνά καταλαμβάνοντας δωμάτια ή ακόμα και ολόκληρους ορόφους. Τα πρώτα μοντέλα δεν είχαν οθόνες, τα προγράμματα έπρεπε να υποβάλλονται μέσω καρτών διάτρησης και τα αποτελέσματα να παραδίδονται ως εκτυπώσεις. Οι πόροι ήταν πραγματικά περιορισμένοι, με αποτέλεσμα να ήταν δύσκολη ακόμη και η πρόσβαση σε έναν κεντρικό υπολογιστή. Επιπλέον, η υπολογιστική ταχύτητα αυτών των πρώιμων υπολογιστών ήταν πολύ βραδύτερη από τις σημερινές μηχανές και με πολύ πιο περιορισμένη μνήμη. Μολοταύτα, οι κεντρικοί υπολογιστές υπήρξαν πρωτοποριακοί, και οι άνθρωποι που τους χρησιμοποιούσαν αξιοποιούσαν κατά το μέγιστο τους σχετικά πενιχρούς τους πόρους. Τα προγράμματα σχεδιάζονταν και συντονίζονταν προσεκτικά, προς μέγιστη απόδοση.

Η παραγωγή ήχου σε αυτά τα μηχανήματα έγινε μια πρακτική πραγματικότητα με την εμφάνιση των πρώτων μετατροπέων ψηφιακού σε αναλογικό, οι οποίοι μετέτρεψαν ψηφιακά ακουστικά δείγματα (ουσιαστικά ακολουθίες αριθμών), που είχαν δημιουργηθεί μέσω υπολογισμού, σε χρονικά μεταβαλλόμενες αναλογικές τάσεις, οι οποίες μπορούσαν να ενισχυθούν σε μεγάφωνα ή να καταγραφούν (π.χ. μαγνητοταινία).

**MUSIC I**

Το πρώτο περιβάλλον προγραμματισμού για τη σύνθεση ήχου, το MUSIC, εμφανίστηκε το 1957. Δεν ήταν μια πλήρης γλώσσα προγραμματισμού, με όρους σημερινούς, αλλά περισσότερο ένα "ακουστικό compiler", που αναπτύχθηκε από τον Max Mathews στα Εργαστήρια AT&T Bell. Η MUSIC I και οι αρχικοί απόγονοί της όχι μόνο ήταν οι πρώτες γλώσσες προγραμματισμού μουσικής που υιοθετήθηκαν ευρέως από ερευνητές και συνθέτες, αλλά και εισήγαγαν αρκετές βασικές ιδέες που εξακολουθούν να επηρεάζουν άμεσα γλώσσες και συστήματα σήμερα.

Η MUSIC I και οι άμεσοι απόγονοί της (που συνήθως αναφέρονται ως γλώσσες MUSIC-N), παρέχουν ένα μοντέλο για τον προσδιορισμό των ενοτήτων ηχητικής σύνθεσης, των συνδέσεών τους και του χρονικά μεταβαλλόμενου ελέγχου. Αυτό το μοντέλο οδήγησε τελικά (με τη MUSIC III) στην ιδέα των γεννητριών μονάδας (unit generators) ή αλλιώς UGen. Τα UGen είναι ατομικά, συχνά προκαθορισμένα, δομικά στοιχεία για την παραγωγή ή την επεξεργασία ηχητικών σημάτων. Εκτός από την είσοδο/έξοδο ήχου, ένα UGen μπορεί να υποστηρίξει μια σειρά εισόδων ελέγχου που ελέγχουν παραμέτρους που σχετίζονται με το UGen. Χαρακτηριστικά παραδείγματα UGen αποτελούν φίλτρα, ενισχυτές και γεννήτριες περιβάλλουσας. Συνδέοντας UGen με διατεταγμένο τρόπο, δημιουργούμε ένα λεγόμενο όργανο ή patch (ο όρος προέρχεται από αναλογικό συνθεσάιζερ που μπορεί να διαμορφωθεί με τη σύνδεση επιμέρους τμημάτων χρησιμοποιώντας καλώδια διασύνδεσης), η οποία καθορίζει τις ηχητικές ιδιότητες (π.χ., χροιά) ενός ήχου.

Στη φρασεολογία της MUSIC-N, μια συλλογή οργάνων αποτελεί μια ορχήστρα . Για να χρησιμοποιήσει την ορχήστρα αυτή προς τη δημιουργία μουσικής, ένας προγραμματιστής θα μπορούσε να σχεδιάσει έναν διαφορετικό τύπο εισόδου που περιέχει χρονικά σεσημασμένες ακολουθίες σημειώσεων ή αλλαγές σήματος ελέγχου, που ονομάζεται παρτιτούρα (score). Συνεπώς, η ορχήστρα καθορίζει πώς παράγονται οι ήχοι, ενώ η παρτιτούρα υπαγορεύει στην ορχήστρα τι πρέπει να παίξει και πότε. Αυτές οι δύο ιδέες – ο UGen και η έννοια μιας ορχήστρας έναντι μιας παρτιτούρας ως προγράμματα- έχουν ασκήσει τεράστια επιρροή στο σχεδιασμό των συστημάτων μουσικού προγραμματισμού, και στον τρόπο με τον οποίο προγραμματίζεται η μουσική υπολογιστών σήμερα.

Την εποχή αυτή, οι ίδιες οι γλώσσες προγραμματισμού υλοποιήθηκαν ως οδηγίες χαμηλού επιπέδου- assembly (ουσιαστικά αναγνώσιμος από τον άνθρωπο κώδικας), που συνδύασαν μια γλώσσα με τη συγκεκριμένη πλατφόρμα εξοπλισμού (hardware) στην οποία εφαρμόστηκε. Καθώς εισήχθησαν νέες γενιές μηχανών (πάντοτε με διαφορετικὀ σετ οδηγιών assembly), έπρεπε να δημιουργηθούν νέες γλώσσες ή τουλάχιστον νέες εφαρμογές για κάθε αρχιτεκτονική. Μετά τη δημιουργία της MUSIC I, ο Max Mathews σύντομα δημιούργησε τις MUSIC II (για το IBM 740), MUSIC III (για το IBM 7094) και MUSIC IV. Αρκετά χρόνια αργότερα, ο John Chowning, ο Andy Moorer και οι συνάδελφοί τους, ολοκλήρωσαν μια επανεγγραφή της MUSIC IV, που ονομάζεται MUSIC 10, καθώς και ένα πρόγραμμα που ονομάζεται SCORE (το οποίο δημιουργούσε λίστες σημειώσεων για το MUSIC 10).

Αξίζει να σημειωθεί ο τρόπος με τον οποίο οι συνθέτες έπρεπε να δουλεύουν με τους υπολογιστές εκείνη την περίοδο. Ο συνθέτης/προγραμματιστής θα σχεδίαζε το λογισμικό (συνήθως μακριά από τον υπολογιστή), θα δημιουργούσε κάρτες διάτρησης που θα συγκεκριμενοποιούσαν τις οδηγίες, και θα τις υπέβαλε ως εργασίες κατά τη διάρκεια του προγραμματισμένου χρόνου πρόσβασης στο mainframe. Η διαδικασία ήταν εξαιρετικά χρονοβόρα. Ένα λεπτό ήχου μπορεί να χρειαζόταν αρκετές ώρες για να δημιουργηθεί, και οι βάρδιες σε χρόνο αρκετών εβδομάδων δεν ήταν ασυνήθιστες. Επιπλέον, δεν υπήρχε κανένας τρόπος να γνωρίζει κανείς εκ των προτέρων εάν το ηχητικό αποτέλεσμα θα ήταν το επιθυμητό. Με το πέρας μιας εργασίας, ο παραγόμενος ήχος θα αποθηκευόταν στην ταινία του υπολογιστή και στη συνέχεια θα μετατρεπόταν από ψηφιακό σε αναλογικό, συνήθως από άλλο υπολογιστή. Μόνο τότε ο συνθέτης θα μπορούσε να ακούσει το αποτέλεσμα. Έτσι, θα απαιτούνταν συνήθως πολλές τέτοιες επαναλήψεις για να ολοκληρωθεί ένα πλήρες κομμάτι μουσικής.

Το 1968, η MUSIC V έφυγε από την πεπατημένη, με το να γίνει το πρώτο σύστημα προγραμματισμού μουσικής υπολογιστή εφαρμόσιμο στη FORTRAN, μια υψηλού επιπέδου γλώσσα προγραμματισμού γενικού σκοπού. Αυτό σήμαινε ότι η MUSIC V θα μπορούσε να μεταφερθεί σε οποιοδήποτε σύστημα υπολογιστών ἐτρεχε FORTRAN, γεγονός που βοήθησε σε μεγάλο βαθμό τόσο τη χρήση της στην κοινότητα των μουσικών υπολογιστών, όσο και την περαιτέρω ανάπτυξή της. Ενώ η MUSIC V ήταν η τελευταία και πιο ώριμη από τις τότε γλώσσες σύνθεσης Max Mathews/Bell Labs, παραμένει μάλλον η πιο καθοριστική γλώσσα μουσικής υπολογιστή. Οι άμεσοι απόγονοί της περιλαμβάνουν τη MUSIC 360 (για το IBM 360, η πιο γρήγορη γλώσσα της εποχής) και τη MUSIC 11 (για το PDP-11), και αργότερα τη cmusic. Αυτά και άλλα συστήματα προσέφεραν μεγάλη συντακτική και λογική ευελιξία, αλλά στον πυρήνα τους έμειναν πιστά στις αρχές των γλωσσών MUSIC-N: σύνδεση UGens και ξεχωριστή επεξεργασία ηχητικής σύνθεσης (ορχήστρα) και μουσικής οργάνωσης (παρτιτούρα). Η MUSIC V αποτέλεσε επίσης το μοντέλο για πολλές ύστερες γλώσσες προγραμματισμού μουσικής.

**The CARL System (or “UNIX for Music”) [5]**

Στη δεκαετία του '70 και '80 έλαβαν τόπο μεγάλες επαναστάσεις στον κόσμο της πληροφορικής. Η γλώσσα προγραμματισμού C, μία από τις πιο δημοφιλείς εν χρήσει, αναπτύχθηκε το 1972. Η δεκαετία του '70 ήταν επίσης μια δεκαετία ωρίμανσης για το σύγχρονο λειτουργικό σύστημα, το οποίο περιλαμβάνει την χρονοκατανομή των κεντρικών πόρων (π.χ. χρόνο CPU και μνήμη) πολλαπλών χρηστών, το factoring των λειτουργιών χρόνου εκτέλεσης μεταξύ μίας προνομιακής λειτουργίας πυρήνα (kernel mode) και μιας πιο προστατευμένης λειτουργίας χρήστη, καθώς και σαφή όρια διαδικασίας που προστατεύουν τις εφαρμογές τη μία από την άλλη. Από τις στάχτες του τιτάνιου λειτουργικού συστήματος Multics δημιουργήθηκε το απλούστερο και πιο πρακτικό UNIX, με στήριξη για πολλαπλές λειτουργίες προγραμμάτων, επικοινωνία μεταξύ πολλών χρηστών, ενδο-διαδικαστική επικοινωνία και μια μεγάλη συλλογή μικρών προγραμμάτων που μπορούν να χρησιμοποιηθούν και να διασυνδεθούν από μια γραμμή εντολών. Υλοποιούμενο τελικά σε γλώσσα C, το UNIX μπορεί να μεταφερθεί με σχετική ευκολία σε οποιαδήποτε νέα πλατφόρμα hardware για την οποία υπάρχει μεταγλωττιστής C.

Με βάση τις ιδέες που έδωσε το UNIX, ο F. Richard Moore, ο Gareth Loy και άλλοι στο Εργαστήριο Computer Audio Research (CARL) του Πανεπιστημίου της Καλιφόρνια, ανέπτυξαν και διένειμαν ένα φορητό σύστημα ανοιχτού κώδικα για την επεξεργασία σήματος και τη σύνθεση μουσικής, το σύστημα CARL. Σε αντίθεση με τα προηγούμενα συστήματα μουσικής υπολογιστή, το CARL δεν αποτελούσε ένα απλό software, αλλά μια συλλογή από μικρά προγράμματα γραμμής εντολών που θα μπορούσαν να στείλουν δεδομένα μεταξύ τους. Η "κατανεμημένη" προσέγγιση είχε ως πρότυπο το UNIX και τη συλλογή της των αλληλοσυνδεόμενων προγραμμάτων, κυρίως για την επεξεργασία κειμένων. Όπως και στο UNIX, μια διαδικασία CARL μπορεί να στείλει την έξοδό της σε μια άλλη διαδικασία μέσω του σωλήνα (|), όμως αντί για κείμενο, οι διαδικασίες CARL στέλνουν και λαμβάνουν ηχητικά δεδομένα (ως αλληλουχίες δειγμάτων κινητής υποδιαστολής, floatsam). Εκτός αυτά, τα προγράμματα CARL μπορούσαν να αποστέλλουν και πληροφορίες πλευρικών καναλιών, οι οποίες επέτρεπαν τη διάδοση ενδεχομένως συνολικών παραμέτρων (όπως το ποσοστό δειγματοληψίας) μέσα στο σύστημα. Σύνθετες εργασίες μπορούσαν να γραφτούν ως ακολουθίες εντολών. Το σύστημα CARL εφαρμόστηκε στη γλώσσα προγραμματισμού C, η οποία εξασφάλισε ένα μεγάλο βαθμό φορητότητας μεταξύ γενεών hardware.

Κατά μία έννοια, η CARL προσέγγισε την ιδέα της ψηφιακής σύνθεσης μουσικής μέσω της λογικής του διαίρει και βασίλευε. Αντί για ένα μονολιθικό πρόγραμμα, προσέφερε μια επίπεδη ιεραρχία μικρών εργαλείων λογισμικού. Το σύστημα προσέλκυσε ένα ευρύ φάσμα συνθετών και ερευνητών μουσικής υπολογιστών που χρησιμοποίησαν την CARL για να γράψουν μουσική και συνέβαλαν στην ανάπτυξή της. Με την πάροδο του χρόνου, η CARL Software Distribution περιλάμβανε πάνω από εκατό προγράμματα. Ενώ το σύστημα ήταν «σπονδυλωτό» (modular) και ευέλικτο για πολλές εργασίες ήχου, η αρχιτεκτονική δεν προοριζόταν για χρήση σε πραγματικό χρόνο. Ίσως για αυτόν το λόγο, το σύστημα CARL δεν χρησιμοποιείται πλέον ευρέως στο σύνολό του. Ωστόσο, χάρη στη φορητότητα της C και στο γεγονός ότι η CARL ήταν open source, ένα μεγάλο μέρος της εφαρμογής έχει φθάσει σε αμέτρητα άλλα ψηφιακά περιβάλλοντα ήχου και αίθουσες διδασκαλίας.

**Cmix, CLM, and Csound**

Την ίδια εποχή, η δημοτικότητα και η φορητότητα της C δημιούργησαν ένα άλλο μοναδικό σύστημα προγραμματισμού: το Cmix του Paul Lansky [4]. Το Cmix δεν πρήλθε απευθείας από τις γλώσσες MUSIC-N· στην πραγματικότητα δεν είναι μια γλώσσα προγραμματισμού, αλλά μια C βιβλιοθήκη χρήσιμων ρουτινών επεξεργαστών σήματος και χειρισμού ήχου, ενοποιημένων από ένα σαφώς ορισμένο API. Ο Lansky συνέταξε την αρχική εφαρμογή στα μέσα της δεκαετίας του '80 για να αναμίξει με ευελιξία τα αρχεία ήχου (εξ ου και το όνομα Cmix) σε αυθαίρετα σημεία. Αποβλέπει εν μέρει στην μείωση της ακαμψίας και του μεγάλου χρόνου παράδοσης για τη σύνθεση μέσω επεξεργασίας παρτιδών. Με την πάροδο του χρόνου, προστέθηκαν πολλές ακόμη οδηγίες επεξεργασίας σήματος και μακροεντολές (macros). Με το Cmix, οι προγραμματιστές μπόρεσαν να ενσωματώσουν λειτουργίες επεξεργασίας ήχου στα δικά τους προγράμματα C για τη σύνθεση ήχου. Επιπλέον, η παρτιτούρα θα μπορούσε να καθορίζεται σε μια γλώσσα Cmix, που ονομάζεται MINC. Η σύνταξη της MINC μοιάζει με εκείνη της C, και αποδείχθηκε ένα από τα πιο ισχυρά εργαλεία scoring της εποχής, λόγω της στήριξής της σε δομές ελέγχου (όπως οι βρόχοι). Το Cmix εξακολουθεί να διανέμεται και να χρησιμοποιείται ευρέως σήμερα, κυρίως με τη μορφή RTCmix (RT = real time).

Η Common Lisp Music είναι μια γλώσσα σύνθεσης ήχου γραμμένη από τον Bill Schottstaedt, στο Πανεπιστήμιο του Στάνφορντ στα τέλη της δεκαετίας του ‘80. [1] Το CLM προέρχεται από την οικογένεια MUSIC-N και χρησιμοποιεί μια σύνταξη βασισμένη στη Lisp για τον ορισμό των οργάνων και της παρτιτούρας, παρέχοντας ταυτόχρονα μια συλλογή λειτουργιών που δημιουργούν και χειρίζονται τον ήχο. Λόγω της φυσικά επαναλαμβανόμενης φύσης της LisP, πολλές ιεραρχικές μουσικές δομές μπόρεσαν να αντιπροσωπευθούν χρησιμοποιώντας κώδικα. Μια πιο πρόσφατη (και πολύ ισχυρή) γλώσσα βασισμένη στην LisP είναι η Nyquist, την οποία συνέταξε ο Roger Dannenberg.

Σήμερα, ο πιο ευρέως χρησιμοποιούμενος άμεσος απόγονος της MUSIC-N είναι η Csound, η οποία αρχικά συντάχθηκε από τον Barry Vercoe και τους συνεργάτες του στο MIT στα τέλη της δεκαετίας του ‘80. Στηρίζει UGens ως opcodes, αντικείμενα δηλαδή που παράγουν ή επεξεργάζονται ήχο. Λειτουργεί με το μοντέλο όργανο εναντίον παρτιτούρας: τα όργανα ορίζονται σε αρχεία ορχήστρας (.orc), ενώ η παρτιτούρα σε αρχεία .sco. Επιπλέον, η Csound υποστηρίζει την έννοια των ξεχωριστών ρυθμών ήχου και ελέγχου. Ο ρυθμός ήχου (συνώνυμος με το ρυθμό δειγματοληψίας) αναφέρεται στο ρυθμό με τον οποίο τα δείγματα ήχου υποβάλλονται σε επεξεργασία μέσω του συστήματος. Από την άλλη πλευρά, ο ρυθμός ελέγχου υπαγορεύει πόσο συχνά τα σήματα ελέγχου υπολογίζονται και διαδίδονται μέσω του συστήματος.

Αυτή η πρώτη εποχή του προγραμματισμού μουσικής υπολογιστή πρωτοστάτησε στον τρόπο με τον οποίο οι συνθέτες μπορούσαν να αλληλεπιδράσουν με τον ψηφιακό υπολογιστή, για να καθορίζουν και να δημιουργούν μουσική. Ο τρόπος λειτουργίας του συνδέθηκε με τις δυσκολίες των πρώιμων mainframes: προγραμματισμός εκτός σύνδεσης, υποβολή εργασιών παρτιδών, αναμονή για δημιουργία ήχου και μεταφορά σε επίμονα μέσα αναπαραγωγής ή συντήρησης.

**THE COMPUTER AGE (PART II): REAL-TIME SYSTEMS**

Στην επόμενη εποχή του μουσικού προγραμματισμού, η κύρια διαφορά είναι ότι ο τρόπος αλληλεπίδρασης εξελίχθηκε από τον προγραμματισμό εκτός σύνδεσης και την επεξεργασία παρτίδων σε συστήματα σύνθεσης ήχου σε πραγματικό χρόνο, τα οποία συχνά ελέγχονται από εξωτερικούς μουσικούς ελεγκτές. Μέχρι τις αρχές της δεκαετίας του 1980, οι υπολογιστές έχουν γίνει αρκετά γρήγοροι και μικροί ώστε να επιτρέπουν στους σταθερούς υπολογιστές (desktops) να ξεπεράσουν τα παλαιότερα mainframes. Καθώς οι προσωπικοί υπολογιστές άρχισαν να πολλαπλασιάζονται, έτσι έγινε και με τα νέα εργαλεία προγραμματισμού και τις εφαρμογές για τη δημιουργία μουσικής.

Graphical Music Programming: Max/MSP + Pure Data

Ένα από τα πιο δημοφιλή περιβάλλοντα προγραμματισμού μουσικής μέχρι σήμερα είναι το Max (αργότερα Max/MSP). Ο Miller S. Puckett εφάρμοσε την πρώτη έκδοση του Max στο IRCAM στο Παρίσι στα μέσα της δεκαετίας του ‘80, ως ένα περιβάλλον προγραμματισμού για τη δημιουργία διαδραστικής μουσικής υπολογιστή. [3] Σε εκείνο το στάδιο, το πρόγραμμα δεν παρήγαγε ούτε επεξεργαζόταν δείγματα ήχου. Ο κύριος σκοπός του ήταν να παρέχει μια γραφική αναπαράσταση για τη δρομολόγηση και τον χειρισμό σημάτων προς έλεγχο εξωτερικών σταθμών εργασίας σύνθεσης ήχου σε πραγματικό χρόνο.  Το 1996, ο Puckette κυκλοφόρησε ένα πλήρως επανασχεδιασμένο και ανοικτού κώδικα περιβάλλον, το Pure Data (PD), το οποίο επεξεργαζόταν δεδομένα ήχου, ενώ το Max σχεδιάστηκε κυρίως για έλεγχο (MIDI). Οι δυνατότητες επεξεργασίας σήματος ήχου του PD στη συνέχεια έγιναν ένα σημαντικό πρόσθετο στο Max, με το όνομα MSP (Max Signal Processing), το οποίο συντάχθηκε από τον Dave Zicarelli.

Programming Libraries for Sound Synthesis

Μέχρι τώρα, έχουν αναφερθεί ως επί το πλείστον αυτόνομα περιβάλλοντα προγραμματισμού, καθένα από τα οποία παρέχει μια εξειδικευμένη γλώσσα, σύνταξη και σημασιολογία. Σε αντίθεση με τέτοιες γλώσσες ή περιβάλλοντα, μια *βιβλιοθήκη* παρέχει ένα σύνολο εξειδικευμένων λειτουργιών για μια υπάρχουσα, ενδεχομένως πιο γενική γλώσσα. Για παράδειγμα, το Synthesis Toolkit (STK) είναι μια συλλογή δομικών στοιχείων για τη σύνθεση ήχου και τη φυσική μοντελοποίηση σε πραγματικό χρόνο, για τη γλώσσα προγραμματισμού C++. Το STK συντάχθηκε από τους Perry Cook και Gary Scavone και κυκλοφόρησε στις αρχές της δεκαετίας του ‘90. Το JSyn, που κυκλοφόρησε την ίδια εποχή, είναι μια συλλογή αντικειμένων σύνθεσης ήχου σε πραγματικό χρόνο για τη γλώσσα προγραμματισμού Java. Σε κάθε περίπτωση, η βιβλιοθήκη παρέχει ένα API (application programming interface), με το οποίο ένας προγραμματιστής μπορεί να γράψει προγράμματα σύνθεσης στη γλώσσα υποδοχής.

SuperCollider: Synthesis Language Redefined

Το SuperCollider είναι μια γλώσσα και περιβάλλον σύνθεσης ήχου βασισμένη σε κείμενο. Είναι απίστευτα ισχυρή ως γλώσσα προγραμματισμού, με μια εξαιρετικά βελτιστοποιημένη εφαρμογή της μηχανής σύνθεσης (synthesis engine). Συνδυάζει πολλές από τις βασικές ιδέες στον σχεδιασμό γλωσσών για υπολογιστές μουσικής, κάνοντας κάποιες θεμελιώδεις αλλαγές και προσθήκες. Το SuperCollider, όπως και οι προγενέστερες γλώσσες του, υποστηρίζει UGen για την επεξεργασία σήματος (ήχου και ελέγχου). Ωστόσο, δεν υπάρχει πλέον διάκριση μεταξύ της ορχήστρας (ηχητική σύνθεση) και της παρτιτούρας (μουσικά γεγονότα): και τα δύο μπορούν να εφαρμοστούν στο ίδιο πλαίσιο. Αυτή η στενότερη ενσωμάτωση οδηγεί σε πιο εκφραστικό κώδικα και στην ικανότητα πειραματισμού με συνθετικές και μουσικές ιδέες, με γρηγορότερο χρόνο παράδοσης. Επιπλέον, η γλώσσα, η οποία είναι αντικειμενοστρεφής, παρέχει ένα εύρος εκφραστικών δομών προγραμματισμού για τη σύνθεση ήχου και τον προγραμματισμό διεπαφής χρήστη. Αυτό καθιστά το SuperCollider κατάλληλο όχι μόνο για την εφαρμογή προγραμμάτων σύνθεσης, αλλά και για την κατασκευή μεγάλων διαδραστικών συστημάτων για σύνθεση ήχου, αλγοριθμική σύνθεση και έρευνα του ήχου.

Graphical vs. Text-based

Αξίζει να σημειωθούν διαφορές μεταξύ των γραφικών περιβαλλόντων προγραμματισμού των Max/MSP και PD, έναντι των γλωσσών και των βιβλιοθηκών με βάση το κείμενο (text-based) όπως το SuperCollider, το Csound και το STK. Η οπτική αναπαράσταση παρουσιάζει απευθείας τη ροή δεδομένων, τύπου “ό,τι βλέπεις παίρνεις”. Τα συστήματα που βασίζονται σε κείμενο δεν έχουν τέτοιου είδους αναπαράσταση, οπότε απαιτείται κατανόηση της σύνταξης και της σημασιολογίας για την αντίληψη των προγραμμάτων. Ωστόσο, πολλές εργασίες, όπως ο προσδιορισμός πολύπλοκων λογικών συμπεριφορών, εκφράζονται πιο εύκολα σε κώδικα που βασίζεται σε κείμενο.

Γενικώς, οι περισσότερες εργασίες σύνθεσης και μουσικής μπορούν να υλοποιηθούν σε οποιαδήποτε από αυτές τις γλώσσες. Αυτή είναι η ιδέα της καθολικότητας: δύο δομές (ή γλώσσες) μπορούν να θεωρηθούν ισοδύναμες, αν μπορούμε να μιμηθούμε τη συμπεριφορά της μίας χρησιμοποιώντας την άλλη και αντίστροφα. Ωστόσο, ορισμένοι τύποι καθηκόντων προσδιορίζονται ευκολότερα σε μια συγκεκριμένη γλώσσα απ’ ότι σε άλλες. Τέλος, είναι και θέμα προσωπικής προτίμησης - ορισμένοι προτιμούν την αμεσότητα των γραφικών γλωσσών, ενώ άλλοι την αίσθηση και την εκφραστικότητα του κώδικα με βάση το κείμενο.

THE COMPUTER AGE (PART III) : NEW LANGUAGE EXPLORATIONS

Με την ανάπτυξη υπολογιστών χαμηλού κόστους και υψηλής απόδοσης, την πληθώρα υλικών και την έκρηξη νέων υψηλού επιπέδου εργαλείων προγραμματισμού γενικού σκοπού (και ανθρώπων που θέλουν να τα χρησιμοποιήσουν), περισσότεροι συνθέτες και μουσικοί κατασκευάζουν όχι μόνο λογισμικά για τη δημιουργία μουσικής, αλλά και λογισμικά για *προγραμματισμό* μουσικής.

Στο πλαίσιο αυτής της νέας εποχής εξερεύνησης, έχει διαμορφωθεί ένα πρόσφατο κίνημα, που αφορά στην άνοδο των δυναμικών γλωσσών, και συνεπώς στη χρήση προγραμματισμού ως μουσικό όργανο. Αυτό, κατά κάποιον τρόπο, μπορεί να θεωρηθεί ως κάτι το θυγατρικό της αλληλεπίδρασης σε πραγματικό χρόνο, αλλά όσον αφορά στον προγραμματισμό της μουσικής, αυτή η ιδέα είναι θεμελιώδης. Για πρώτη φορά, έχουμε υπολογιστές που μπορούν να παράγουν ήχο και μουσική σε πραγματικό χρόνο (και σε αφθονία) σύμφωνα με τις προδιαγραφές του προγράμματός μας. Ένας από τους τομείς που ερευνήθηκαν σε τούτη την εποχή του προγραμματισμού είναι οι δυνατότητες αλλαγής του ίδιου του προγράμματος σε πραγματικό χρόνο, καθώς τρέχει.

Η ιδέα της τροποποίησης των προγραμμάτων για τη δημιουργία μουσικής (live coding, προγραμματισμός on-the-fly, διαδραστικός προγραμματισμός) δεν είναι εντελώς νέα. Από τις αρχές της δεκαετίας του '80, ερευνητές όπως ο Ron Kuivila, και ομάδες όπως η Hub έχουν πειραματιστεί με μουσικά συστήματα τροποποίησης χρόνου εκτέλεσης. Αξιοσημείωτη είναι και η γλώσσα ιεραρχικής μουσικής βαθμολόγησης(HMSL), που βασίζεται στην Forth, και συντάχθηκε από τον Larry Polansky, τον Phil Burk και άλλους στη δεκαετία του '80, της οποίας η σύνταξη ενθαρρύνει τον runtime προγραμματισμό. Αυτοί είναι οι πρόδρομοι του live coding, και οι γρήγοροι υπολογιστές του σήμερα επιτρέπουν ένα πρόσθετο βασικό στοιχείο: σύνθεση ήχου σε πραγματικό χρόνο.

ChucK: Strongly-timed and On-the-fly Programming Language [2]

Το ChucK είναι ένα από τα νεότερα μέλη της οικογένειας γλωσσών προγραμματισμού σύνθεσης ήχου, και απορρέει έμμεσα από το πρότυπο της MUSIC-N, με κάποιες βασικές διαφορές. Πρώτον, ο χειριστής chuck χρησιμοποιείται για την εκτέλεση ενεργειών από αριστερά προς δεξιά, συμπεριλαμβανομένης της σύνδεσης UGen. Η γλώσσα καθιερώνει, επίσης, μια ισχυρή αντιστοιχία μεταξύ χρόνου και ήχου στο ότι ο προγραμματιστής ελέγχει τη ροή (μέσω ειδικής σύνταξης γλώσσας) για να επιτρέπεται η δημιουργία ήχου (strongly-timed). Επιπλέον, διαφορετικές διαδικασίες μπορούν να συγχρονιστούν μεταξύ τους σύμφωνα με την ίδια έννοια του χρόνου ή με δεδομένα. Δεν υπάρχει διαχωρισμός ορχήστρας/παρτιτούρας, και ο ρυθμός ελέγχου προσδιορίζεται προγραμματικά ως συνέπεια χειρισμού του χρόνου. Με αυτή την έννοια, το ChucK είναι η πρώτη γλώσσα σύνθεσης σε πραγματικό χρόνο που απομακρύνεται από την κλασική έννοια του ρυθμού ελέγχου.

Custom Music Programming Software

Μια ζωντανή πτυχή της εποχής είναι και ο πολλαπλασιασμός των “home-brew” λογισμικών για προγραμματισμό ήχου. Η έκρηξη νέων πλατφορμών προγραμματισμού υψηλού επιπέδου επέτρεψε και ενθάρρυνε τους προγραμματιστές και τους συνθέτες να δημιουργήσουν συστήματα προσαρμοσμένα στις προτιμήσεις τους. Ο Alex McLean περιγράφει το live coding, χρησιμοποιώντας την υψηλού επιπέδου γλώσσα προγραμματισμού Perl. Παρόμοια πλαίσια έχουν αναπτυχθεί στην Python και διάφορες διαλέκτους των Lisp, Forth, Ruby, κλπ. Ορισμένα συστήματα παράγουν ήχο, ενώ άλλα τον οπτικοποιούν. Πολλά συστήματα στέλνουν μήνυμα δικτύου (στο OpenSoundControl) σε μηχανές σύνθεσης όπως το SuperCollider, PD, Max και ChucK. Με αυτόν τον τρόπο, οι μουσικοί και οι συνθέτες μπορούν να εκμεταλλευτούν την εκφραστικότητα της front-end γλώσσας, για τη δημιουργία μουσικής.

Αυτή η εποχή είναι πολλά υποσχόμενη, διότι επιτρέπει και ενθαρρύνει νέες δυνατότητες σύνθεσης και απόδοσης που είναι διαθέσιμες όχι μόνο σε επαγγελματίες μουσικούς, ερευνητές και ακαδημαϊκούς, αλλά και σε όσους επιθυμούν να μάθουν και να εξερευνήσουν τον προγραμματισμό και τη μουσική. Επίσης, τα νέα δυναμικά περιβάλλοντα για τον προγραμματισμό αλλάζουν τον τρόπο με τον οποίο πλησιάζουμε την πιο "παραδοσιακή" σύνθεση μουσικής από υπολογιστή - παρέχοντας ταχύτερους πειραματισμούς και πιο άμεσο feedback.

**Ζητήματα**

Οι υπολογιστές δεν είναι ευέλικτοι και απαιτούν ακρίβεια από τον προγραμματιστή. Δεν ξέρουν να μας ενημερώνουν ούτε για τα πιο προφανή λάθη μας (εκτός αν κάποιος έχει πει στον υπολογιστή τι ακριβώς να ψάξει). Επιπλέον, δεδομένου ότι πρέπει να διατυπώσουμε τις προθέσεις μας όσον αφορά τη σύνταξη της υποκείμενης γλώσσας, οι μουσικές ιδέες δεν είναι πάντα απλό να μεταφραστούν σε κώδικα. Από την άλλη, ο προγραμματισμός μάς επιτρέπει να διερευνήσουμε ήχους και μουσικές διαδικασίες που υπό άλλες συνθήκες δεν είναι διαθέσιμες ή είναι αδύνατο να δημιουργηθούν. Για κάποιον που δεν έχει ασχοληθεί ξανά με προγραμματισμό, η δημιουργία προγραμμάτων για τη σύνθεση μουσικής μπορεί να είναι ένας από τους καλύτερους τρόπους για να μάθει τον προγραμματισμό: το feedback τις περισσότερες φορές είναι άμεσο, και ο προγραμματισμός μαθαίνεται θα μπορούσαμε να πούμε περισσότερο «παράπλευρα» παρά ως αυτοσκοπός.

Δικαίως, η μουσική μέσω υπολογιστή θεωρείται μια μορφή τέχνης, καθώς ο υπολογιστής ως εργαλείο δύναται να επεκτείνει την δημιουργικότητα, δεν την περιορίζει. Με τις εξελίξεις αυτές όμως, τίθενται και κάποια βασικά ζητήματα/ερωτήματα, όσον αφορά στην ίδια τη φύση της μουσικής και τον τρόπο δημιουργίας της. Ο παραδοσιακός διαχωρισμός μεταξύ συνθέτη, οργανοπαίκτη και οργανοποιού έχει πάψει να υφίσταται, και η μουσική μέσω υπολογιστή δεν είναι πια κάτι το αμφιλεγόμενο, τουναντίον θεσμοθετείται ολοένα και περισσότερο.

**ΒΙΒΛΙΟΓΡΑΦΙΑ**

Csound (sel 21): Pope, S. T. 1993. ‘Machine Tongues XV: Three Packages for Software Sound Synthesis’, Computer Music Journal, 17(2):25-54.

[1] W. Schottstaedt, *Common Lisp Music Documentation* (1992), σελ. 19

[2] Ge Wang, The *ChucK Audio Programming Language: A Strongly-timed and On-the-fly*

*Environ/mentality*, (διδακτορική διατριβή, Πανεπιστήμιο Princeton, 2008), σελ. 35

[3] Puckett, *Combining Event and Signal Processing in the MAX Graphical*

*Programming Environment* (1991)

[4] Paul Lansky, *The Architecture and Musical Logic of Cmix* (1990)

[5] Ge Wang, σελ. 15

[6] Θ. Λώτης και Τ. Διαμαντόπουλος, *Μουσική Πληροφορική & Μουσική με Υπολογιστές,* (ΣΕΑΒ 2015), σελ. 8

1. Csound (sel 21): Pope, S. T. 1993. ‘Machine Tongues XV: Three Packages for Software Sound Synthesis’, Computer Music Journal, 17(2):25-54. [↑](#endnote-ref-1)