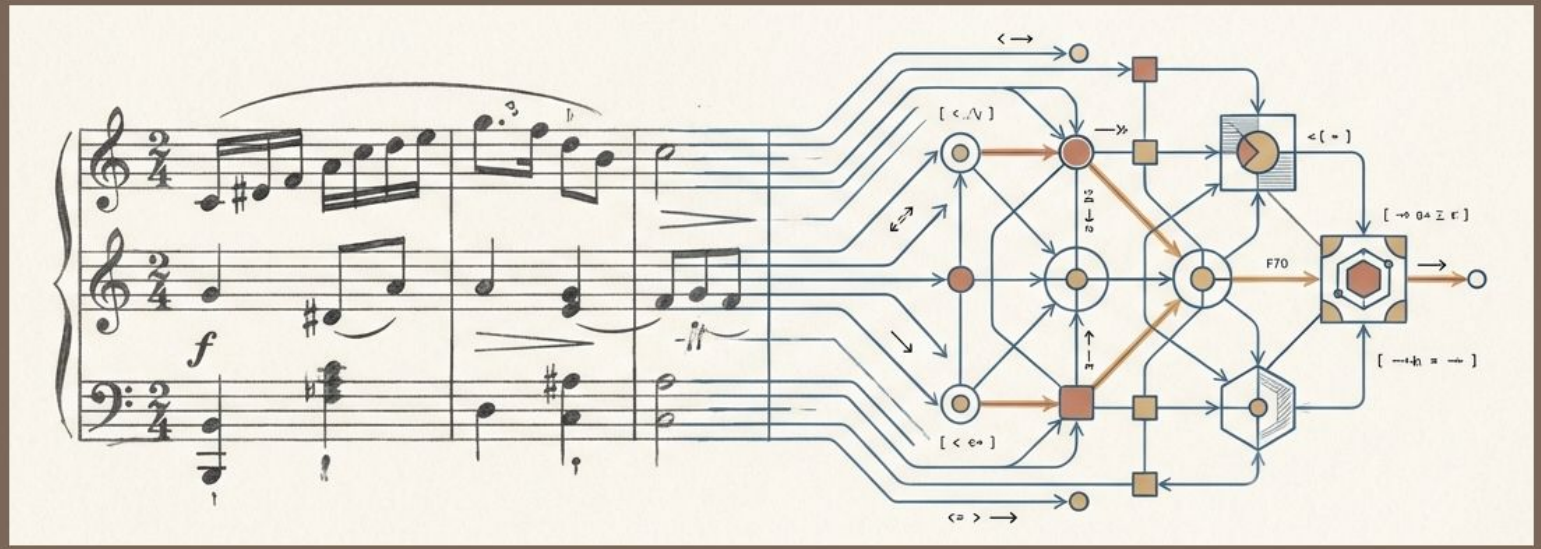
ΑΠΟ ΤΗΝ ΜΟΥΣΙΚΗ ΙΔΕΑ ΣΤΟ ΑΛΓΟΡΙΘΜΙΚΟ ΕΡΓΑΛΕΙΟ



My DIGITAL GENEALOGY

- Amstrad 464 (Tapes)
- Intel 8088 (Disks)
- Windows 95 (CD-Rom)
- Windows XP (USB) (AMIS) (CSound)
- MacOS (Pure Data)
- Cloud Computing
- AI Tools

https://tokeno.net

https://ihearcolors.online

Τί είναι αυτή η παρουσίαση

Αλγοριθμική σκέψη στη μουσική

- τρόποι να σκεφτόμαστε τη μουσική σαν **σύστημα**
- χρήση **κανόνων και διαδικασιών**
- μουσική που **εξελίσσεται μέσα από (user) feedback**
- σύνθεση ως **δυναμική διαδικασία**, όχι στατικό έργο

Τί ΔΕΝ είναι αυτή η παρουσίαση

Δεν είναι:

- μάθημα προγραμματισμού
- τεχνικό tutorial software
- παραγωγή μουσικής σε DAW

Η εστίαση είναι:

Πώς σκεφτόμαστε τη μουσική με αλγοριθμικό τρόπο.

Μέρος 1

Η Ιστορία ενός Αλγόριθμου

Αλγόριθμος

Μια σαφής διαδικασία από **κανόνες ή βήματα** που παράγουν ένα αποτέλεσμα.

Παραδείγματα εκτός μουσικής:

- Μια συνταγή μαγειρικής
- Οδηγίες πλοήγησης
- Ένας μαθηματικός υπολογισμός

Κεντρική ιδέα

Στην μουσική οι αλγόριθμοι χρησιμοποιούνται για να δημιουργούν **αυτόνομα ή ημιαυτόνομα μουσικά σχήματα μέσα στον χρόνο**.

Αντί να γράφουμε κάθε νότα η να επαναλαμβάνουμε μια μουσική διαδικασία:

Σχεδιάζουμε **κανόνες που παράγουν (αυτόματα) - αυτόνομα μουσικα δεδομένα.**

Η σύνθεση γίνεται:

Σύστημα Συμπεριφοράς.

Αλγόριθμοι παντού

Ableton Live

- Follow Actions → rule-based playback
- Probability MIDI tools → stochastic behavior

Logic Pro

- Step sequencer με probability
- Arpeggiators

Pure Data (full algorithmic composition)

- πλήρης έλεγχος
- χτίζεις δικούς σου αλγορίθμους
- interaction / installations

SuperCollider

- synthesis + algorithmic composition
- patterns, stochastic systems

Αλγόριθμοι παντού

Οι περισσότεροι χρήστες:

Χρησιμοποιούν αλγοριθμικά εργαλεία χωρίς να το ξέρουν

π.χ.

- arpeggiator = algorithm
- random velocity = stochastic system

Γιατί να χρησιμοποιούμε αλγορίθμους στη μουσική;

Σπάσιμο Συνηθειών > Μηχανισμός παραγωγής ιδεών **vs**

ίδιες κλίμακες - ίδια patterns - ίδια transitions

Ογκος μουσικών δεδομένων > μεγαλύτερο πεδίο πιθανών αποτελεσμάτων -

Μακροσκοπική ακουστική αντιληψη

Πολυπλοκότητα > το σύστημα κρατάει την πολυπλοκότητα εσύ κρατάς τον έλεγχο

Γιατί να χρησιμοποιούμε αλγορίθμους στη μουσική;

Αυτόνομος οργανισμός > μουσική που ζει > μεταλλάσσεται, ιδανικό για πολύωρα installations

Απλή τυχαιότητα > οχι ενδιαφέρον **vs** αλγόριθμος > random + constraints + feedback

Δηλαδή: δίνει > κατεύθυνση - χαρακτήρα - ταυτότητα

Μετακίνηση από αποτέλεσμα σε διαδικασία > από “μου αρέσει αυτό το κομμάτι;” >

“μου αρέσει ο τρόπος που έχει σχεδιαστεί αυτό το σύστημα;”

Απόφαση αντί για νότα

Παραδοσιακά:

Συνθέτης → Γράφει νότες ή δημιουργεί ηχητικά γεγονότα (per step).

Αλγοριθμική προσέγγιση:

Συνθέτης → σχεδιάζει **κανόνες αποφάσεων**

π.χ.

- Αν υπάρχει επανάληψη → αλλαγή αρμονίας
- Αν υπάρχει χάος → μείωση πυκνότητας

(Ableton > Follow Actions)

Αλγόριθμοι στη Φύση → Μουσική

Fibonacci sequence

Εμφανίζεται σε φυτά, κοχύλια κτλ

- Pattern: 0, 1, 1, 2, 3, 5, 8...

Μουσική χρήση:

- ρυθμικές δομές
- phrase lengths

Fibonacci sequence

Στη φύση το Fibonacci:

- Δεν είναι aesthetic choice
- Είναι **optimization strategy**

Μεγιστοποιεί:

- Χώρο
- Κατανομή
- Αποφυγή επανάληψης

Fibonacci sequence στη μουσική

Μη επαναλαμβανόμενη επανάληψη (controlled variation)

Να ακούγεται οικείο χωρίς να γίνεται στεγνό loop

Distribution στον χρόνο (rhythmic spacing)

Όπως τα φύλλα κατανέμονται για να μην σκιάζονται,

τα μουσικά events κατανέμονται για να μην “πατάνε” το ένα το άλλο

Density control (χώρος vs πληροφορία)

Στη φύση: πόσα seeds χωράνε χωρίς να πνιγούν.

Στη μουσική: πόσα events αντέχει το αυτί.

Scaling / self-similarity (fractal thinking)

το ίδιο pattern σε διαφορετικές κλίμακες

‘Feedback’ Composition

Ένα αλγοριθμικό σύστημα μπορεί να:

1. **Παράγει αυτόνομα μουσικό υλικό**
2. Να **παρατηρεί το αποτέλεσμα**
3. Να **αλλάζει τη συμπεριφορά του**

Οι κανόνες μπορεί να ελέγχουν:

- **Μορφή**
- **Πυκνότητα**
- **Ρυθμική συμπεριφορά**
- **Αρμονική μετατόπιση**

Το μουσικό αποτέλεσμα προκύπτει:

Από τη δυναμική του συστήματος σε σχέση με τους κανόνες που έχουμε ορίσει.

Σύνθεση ως οικοσύστημα

Μπορούμε να δούμε ένα αλγοριθμικό μουσικό σύστημα σαν:

ένα **οικοσύστημα πρακτόρων (Agents)**

Που:

- Αλληλεπιδρούν
- Ανταποκρίνονται
- Μεταβάλλονται
- Αντιδρούν σε εξωτερικές συνθήκες *

Ο ρόλος του συνθέτη

Γίνεται:

Σχεδιαστής και Ρυθμιστής Διαδικασιών.

Σχεδιάζει:

- **Κανόνες**
- **Μουσικές συμπεριφορές**
- **Μακροσκοπικά**
- **Παραδίδοντας ολικά ή εν μέρη τον έλεγχο.**

Το Νέο Παράδειγμα: Από τη Νότα στον Κανόνα

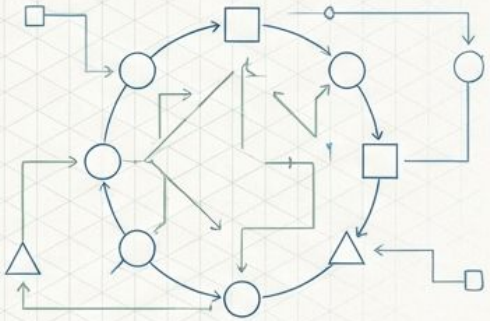
Παραδοσιακή Σύνθεση

Αντικείμενο: Στατικό έργο (Timeline-based).

Μονάδα Ελέγχου: Απόφαση ανά νότα (`Pitch`, `Duration`).

Ρόλος Συνθέτη: Δημιουργός του τελικού ηχητικού αποτελέσματος.

Αλγοριθμική Σύνθεση



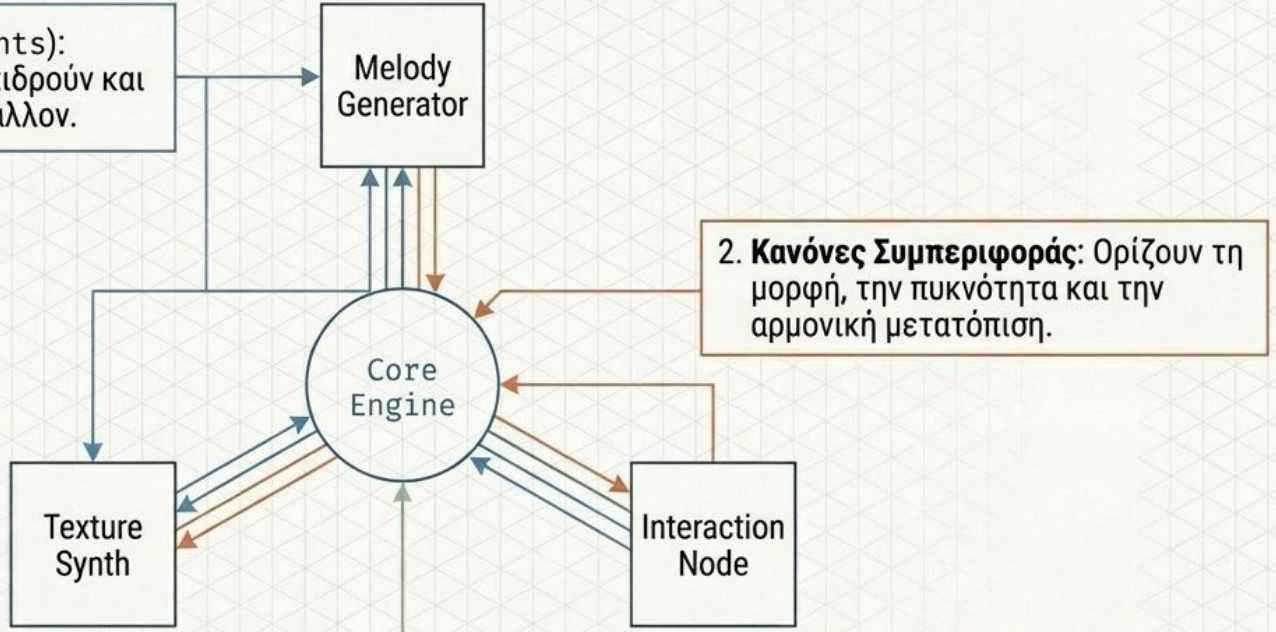
Αντικείμενο: Δυναμικό οικοσύστημα (`Behavior-based`).

Μονάδα Ελέγχου: Σχεδιασμός κανόνων και περιορισμών.

Ρόλος Συνθέτη: Σχεδιαστής διαδικασιών & συστημάτων.

Ο Συνθέτης ως Αρχιτέκτονας Συστημάτων

1. **Αυτόνομοι Πράκτορες** (`Agents`): Υποσυστήματα που αλληλεπιδρούν και ανταποκρίνονται στο περιβάλλον.



2. **Κανόνες Συμπεριφοράς**: Ορίζουν τη μορφή, την πυκνότητα και την αρμονική μετατόπιση.

3. **Μουσικό Οικοσύστημα**: Το αποτέλεσμα προκύπτει από τη δυναμική εξέλιξη, όχι από προκαθορισμένη παρτιτούρα.

Αλγόριθμοι στη Φύση → Μουσική

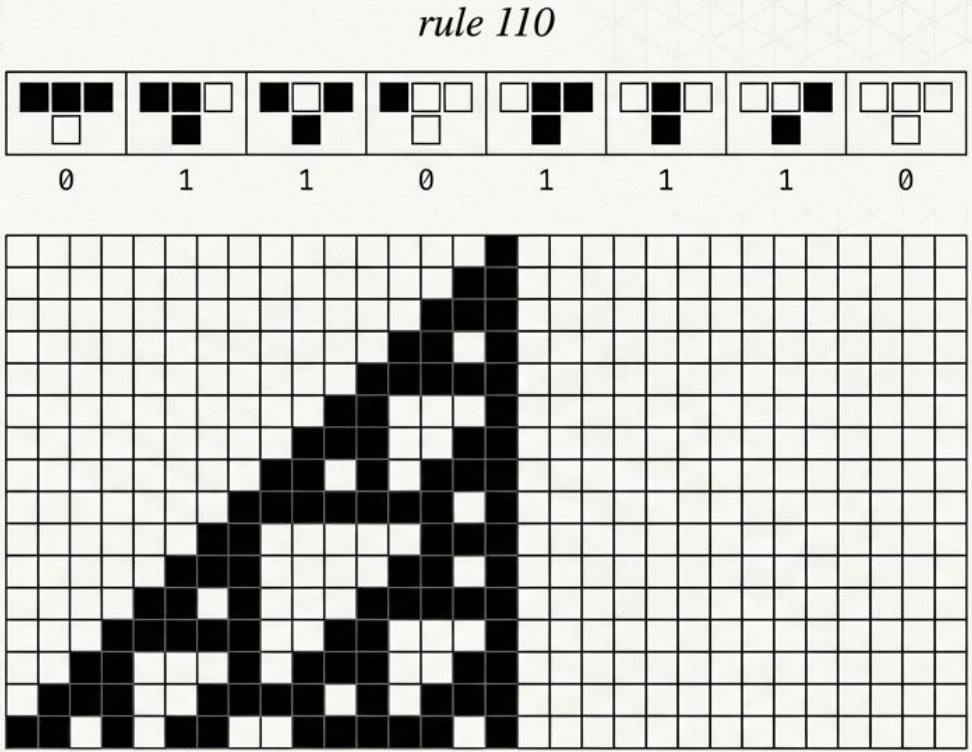
Cellular Automata - Κυτταρικά Αυτόματα

- Απλοί τοπικοί κανόνες → σύνθετα patterns
- (π.χ. Φυσική, Βιολογία (ανάπτυξη οργανισμών και γενικά για ανάλυση σύνθετων αυτοοργανωμένων συστημάτων)

Μουσική χρήση:

- Evolving rhythms
- Generative textures

Οπτική Γκαλερί: Cellular Automata (Κανόνας 110)



Concept

Απλοί, τοπικοί κανόνες που εφαρμόζονται σε ένα πλέγμα παράγουν εξαιρετικά σύνθετα, αυτοοργανωμένα patterns.

Musical Application

Χρήση σε generative ρυθμούς (evolving textures) και αρμονικές εξελίξεις. Μικρές μεταβολές στην αρχική κατάσταση δημιουργούν τεράστιες, εξελισσόμενες μουσικές αποκλίσεις (Ντετερμινιστικό Χάος).

Αλγόριθμοι στη Φύση → Μουσική

Oscillation (ταλάντωση)

- Περιοδική κίνηση (κύματα, καρδιακός ρυθμός)

Μουσική χρήση:

- LFOs
- Ρυθμός

Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Τα θεωρητικά θεμέλια της αλγοριθμικής σκέψης στη μουσική εμφανίζονται από την αρχαιότητα.

Ο **Πυθαγόρας** (περ. 500 π.Χ.) καθιέρωσε τη μαθηματική βάση της μουσικής θεωρίας, θεμελιώνοντας την πεποίθηση ότι **η κατανόηση των αριθμών είναι το κλειδί για την κατανόηση του σύμπαντος.**

Οι αρχαίοι Έλληνες **ενέτασσαν τη μουσική στις κλασικές τέχνες** μαζί με την αριθμητική, τη γεωμετρία και την αστρονομία, γεγονός που υποδηλώνει μια βαθιά αλγοριθμική αντίληψη της μουσικής δομής.

Πυθαγόρας

Στη μελέτη του για τη μουσική παρατήρησε ότι τα **αρμονικά μουσικά διαστήματα αντιστοιχούν σε απλές μαθηματικές αναλογίες**.

Παράδειγμα:

- 2:1 → οκτάβα
- 3:2 → πέμπτη
- 4:3 → τετάρτη

Αυτές οι αναλογίες εμφανίζονται π.χ. όταν αλλάζει το μήκος μιας χορδής σε ένα όργανο.

Για τον Πυθαγόρα αυτό σήμαινε ότι **η μουσική και τα μαθηματικά είναι βαθιά συνδεδεμένα**.

Music of the Spheres

- Τα ουράνια σώματα (πλανήτες, ήλιος, σελήνη) κινούνται με **μαθηματικές αναλογίες**
- Αυτές οι αναλογίες αντιστοιχούν σε **μουσικά διαστήματα**
- Το σύμπαν σχηματίζει μια **κοσμική αρμονία**

Musica Universalis

Πυθαγόρας και Ακουσματική Μουσική

Η ιδέα του «ήχου χωρίς ορατή πηγή»

Στη σχολή του Πυθαγόρα:

- Οι μαθητές άκουγαν τον δάσκαλο **πίσω από ένα παραπέτασμα**
- Δεν έβλεπαν τον ομιλητή
- Η γνώση έπρεπε να κρίνεται **μόνο από αυτό που ακούγεται**

Οι μαθητές αυτοί ονομάζονταν **Ακουσματικοί.**

Πυθαγόρας και Ακουσματική Μουσική

Ακουσματική Μουσική - Acousmatic Music

Η έννοια επανεμφανίζεται τον 20ό αιώνα από τον **Pierre Schaeffer** (Πιέρ Σεφέρ, 1910–1995).

Στο πλαίσιο της Musique Concrète:

- Οι ήχοι αναπαράγονται από **ηχεία**
- Δεν υπάρχει ορατός εκτελεστής
- Ο ακροατής επικεντρώνεται **στον ίδιο τον ήχο**

Reduced Listening

Ο Schaeffer εισάγει την έννοια:

écoute réduite (μειωμένη ακρόαση)

Ο ακροατής δεν σκέφτεται:

- τι προκαλεί τον ήχο

Αλλά αφήνεται:

- υφή
- φάσμα
- διάρκεια
- κίνηση

Acousmatic Music

Όταν **αφαιρείται η πηγή**:

Ο ήχος γίνεται:

Αντικείμενο

Η μουσική γίνεται:

Αυτόνομο Καλλιτεχνικό Έργο.

Schaeffer and Cage

Schaeffer → αφαιρεί την *πηγή* → αλλάζει τον τρόπο *ακρόασης*

Cage → αφαιρεί την *πρόθεση* → αλλάζει τον τρόπο *δημιουργίας*

John Cage Parenthesis

John Cage

- Αμερικανός συνθέτης (20ός αιώνας)
- Πρωτοπόρος της Avant Garde

Βασικές Ιδέες

- **Chance / Τυχαιότητα**
  Χρήση διαδικασιών (π.χ. I Ching) αντί προσωπικών επιλογών
- **Indeterminacy (Απροσδιοριστία)**
  Το αποτέλεσμα δεν είναι πλήρως καθορισμένο
- **Απομάκρυνση του ελέγχου, Αφαίρεση του Εγώ**
  Ο συνθέτης σχεδιάζει το σύστημα, όχι κάθε ήχο
- **Όλα μπορούν να είναι μουσική**
  Περιβάλλον, θόρυβος, σιωπή

John Cage Parenthesis

Ο John Cage δεν έγραφε κώδικα, αλλά χρησιμοποιούσε **αλγόριθμους χωρίς υπολογιστή**.

John Cage Parenthesis

Δημιουργεί συστήματα → μια διαδικασία που παράγει νότες με τυχαιότητα

Θέτει constraints (περιορισμούς) → μόνο νότες από πεντατονική κλίμακα

Αποφασίζει το πλαίσιο → κομμάτι διάρκειας 5 λεπτών για πιάνο

Το **Εγώ** δεν εξαφανίζεται, απλά μετακινείται:

Από τον Συνθέτη → στο Σύστημα

John Cage Parenthesis

Ο **Schaeffer** αποσυνδέει **αιτία**

Ο **Cage** αποσυνδέει **έλεγχο**

John Cage Parenthesis

Και οι δύο μετατοπίζουν το κέντρο της μουσικής εμπειρίας, αλλά με διαφορετικό τρόπο:

- **Schaeffer**:
  από την προέλευση του ήχου → στην ίδια την ηχητική του υπόσταση

- **Cage**:
  από την προσωπική πρόθεση → στην εξέλιξη του γεγονότος

Ο **Schaeffer** αποκόπτει τον ήχο από την αιτία του.

Ο **Cage** αποσύρει τον δημιουργό από τον έλεγχο του αποτελέσματος.

John Cage Parenthesis

1. Music of Changes (1951)

- Χρησιμοποιεί το **I Ching**
- Κάθε παράμετρος (pitch, duration, dynamics) αποφασίζεται με ρίψεις ζαριών

Αλγοριθμικά:

```
for each event:
   pitch = random_table()
   duration = random_table()
   dynamic = random_table()
```

Δεν υπάρχει καμία “έμπνευση” εκείνη τη στιγμή, μόνο **εκτέλεση διαδικασίας.**

Η Αποδόμηση του 20ου Αιώνα

Παραδοσιακή Μουσική

Pierre Schaeffer
Αφαιρεί: Την Πηγή (Ορατότητα).
Διαδικασία: Écoute Réduite. Ο ήχος αποκόπτεται από την αιτία του.
Αποτέλεσμα: Ακουσματική Μουσική (Ο ήχος ως αυτόνομο αντικείμενο).

John Cage
Αφαιρεί: Την Πρόθεση (Έλεγχος / Εγώ).
Διαδικασία: Indeterminacy & Τυχαιότητα (π.χ. I Ching).
Αποτέλεσμα: Η σύνθεση ως απρόβλεπτο γεγονός. Το κέντρο βάρους μετατοπίζεται στο σύστημα.

Γενεαλογία της Αλγοριθμικής Σκέψης

Athanasius Kircher (1650)
Arca Musarithmica. Μηχανικό σύστημα παραγωγής πολυφωνίας μέσω κωδικοποιημένων καρτών.

Arnold Schoenberg (20ός αι.)
Σειραϊσμός. Μια σειρά 12 φθόγγων παράγει 479 εκατομμύρια μαθηματικές μεταθέσεις.

Πυθαγόρας (500 π.Χ.)
Μαθηματικές αναλογίες & Musica Universalis. Ο αριθμός ως θεμέλιο της κοσμικής αρμονίας.

J.S. Bach (18ος αι.)
Αλγοριθμικές μεταθέσεις. Το όνομα B-A-C-H ως μαθηματικό μοτίβο.

Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Συνέχεια

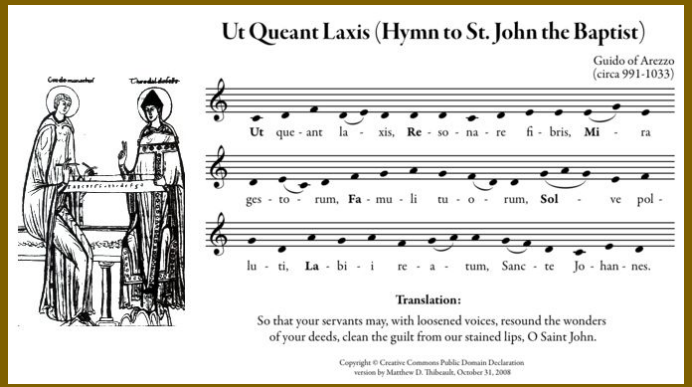
.

.

.

Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Το μνημονικό σύστημα σύνθεσης και η τυποποίηση της μουσικής σημειογραφίας του **Guido d'Arezzo (Γκουίντο ντ' Αρέτσο)** (11ος αι.) διαχώρισαν για πρώτη φορά τον ρόλο του συνθέτη από εκείνον του εκτελεστή.



Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Η περίοδος του **Μπαρόκ** είδε την εμφάνιση πραγματικών συστημάτων σύνθεσης, με πιο χαρακτηριστικό παράδειγμα την **Arca Musarithmica** του Athanasius Kircher (1650), ένα κουτί καρτών που κωδικοποιεί κανόνες μέτρου, ρυθμού και τονικότητας και επιτρέπει την μηχανική παραγωγή πολυφωνικής μουσικής ακολουθώντας συγκεκριμένες οδηγίες.

Παράλληλα, ο **J.S. Bach** χρησιμοποίησε αλγοριθμικές διαδικασίες με το μοτίβο B–A–C–H, όπου το όνομά του μετατρέπεται σε μελωδικό υλικό μέσω κανόνων, όπως ακούγεται στη Fugue XV από το **The Well-Tempered Clavier (Book 1)**.

Arca Musarithmica

‘Mechanical music-making is nothing more than a particular system invented by us whereby anyone, even the ἀμουσος [unmusical] may, through various applications of compositional instruments compose melodies according to a desired style.’

Athanasius Kircher, Musurgia Universalis (1650), Book VIII, page 185

https://www.arca1650.info/prepared.html

B.A.C.H

Προέρχεται από τη **γερμανική ονοματολογία των νοτών** και χρησιμοποιήθηκε ως μουσική «υπογραφή» από τον Bach.

Είσοδος: όνομα "BACH"

Κανόνας: Μετατροπή γραμμάτων σε νότες με βάση το γερμανικό σύστημα.

Έξοδος: [Bb, A, C, B]

Μετά ο συνθέτης μπορεί να το επεξεργαστεί:

- Μεταφορά σε άλλη τονικότητα (transpose)
- Αναστροφή (inversion)
- Ανάποδα (retrograde)
- Χρήση ως θέμα σε φούγκα

Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Στις αρχές του 20ού αιώνα, η αλγοριθμική σκέψη τυποποιήθηκε μέσω του **Σειραϊσμού.**

Ο **Arnold Schoenberg** και οι μαθητές του **Anton Webern** και **Alban Berg** ανέπτυξαν την τεχνική των δώδεκα φθόγγων, όπου μία μόνο σειρά μπορούσε να παράγει 479.001.600 μεταθέσεις…

$$(12 \times 11 \times 10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1)$$

Σειραϊσμός

Ο συνθέτης δημιουργεί μια **σειρά από τις 12 νότες της χρωματικής κλίμακας**, χωρίς να επαναλαμβάνεται καμία πριν ακουστούν όλες.

Παράδειγμα σειράς:

Ντο – Φα – Μι – Σολ ♭ – Ρε – Λα ♭ – Σι – Μι ♭ – Σολ – Ρε ♭ – Λα – Σι ♭

Αυτή η σειρά λέγεται **tone row** ή **σειρά 12 φθόγγων**.

Μετά όλο το κομμάτι χτίζεται από αυτή τη σειρά.

Σειραϊσμός

Οι βασικοί μετασχηματισμοί

Η σειρά μπορεί να εμφανιστεί με διαφορετικούς τρόπους:

1. **Prime (P)** – η αρχική μορφή
2. **Retrograde (R)** – ανάποδα
3. **Inversion (I)** – αντιστροφή των διαστημάτων
4. **Retrograde Inversion (RI)** – ανάποδη αντιστροφή

Αυτό δημιουργεί ένα **σύστημα κανόνων**, σχεδόν σαν μαθηματικό παιχνίδι.

Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Ένα υλικό παράδειγμα προ-ψηφιακής αλγοριθμικής λογικής συναντάμε στα *Studies for Player Piano* (1940s–90s) του **Conlon Nancarrow**, όπου ο συνθετικός αλγόριθμος χαρασσόταν κυριολεκτικά σε ρολά πιάνου, καθορίζοντας τέμπο και ρυθμούς πέρα από τις ανθρώπινες εκτελεστικές δυνατότητες.

Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Αυτές οι ιδέες επεκτάθηκαν περαιτέρω από τον Iannis Xenakis, ο οποίος εισήγαγε τη στοχαστική μουσική εφαρμόζοντας τη θεωρία πιθανοτήτων σε ηχητικές μάζες σε έργα όπως το Metastaseis & *Pithoprakta* (1953–56).

Ξενάκης - Pithoprakta

Η βασική ιδέα

Αντί να ελέγχει κάθε νότα ξεχωριστά, ο Ξενάκης ελέγχει **στατιστικές συμπεριφορές μεγάλων ομάδων ήχων**.

Δηλαδή:

- όχι «μελωδία»
- αλλά **μαζική κίνηση ήχων**.

Κάτι σαν **σμήνος από μόρια ή πουλιά**.

Ξενάκης - Pithoprakta

Η μαθηματική βάση

Στα *Pithoprakta* χρησιμοποιείται η **κίνηση Brown (Brownian motion)**, ένα μοντέλο από τη φυσική που περιγράφει την τυχαία κίνηση σωματιδίων.

Ο Ξενάκης φαντάστηκε κάθε **όργανο της ορχήστρας σαν ένα σωματίδιο** που κινείται στο τονικό ύψος (pitch).

Πώς λειτουργεί η «αλγοριθμική» διαδικασία

Για κάθε όργανο καθορίζεται:

- αρχικό ύψος
- πιθανότητα μετακίνησης προς τα πάνω ή κάτω
- μέγεθος διαστήματος
- διάρκεια

Κάθε νότα μετακινείται **τυχαία αλλά μέσα σε συγκεκριμένους κανόνες**.

Ξενάκης - Pithoprakta

Άλλες διαδικασίες που χρησιμοποιεί

Στατιστικές κατανομές

Π.χ. **Gaussian distribution** για την κατανομή των τονικών υψών.

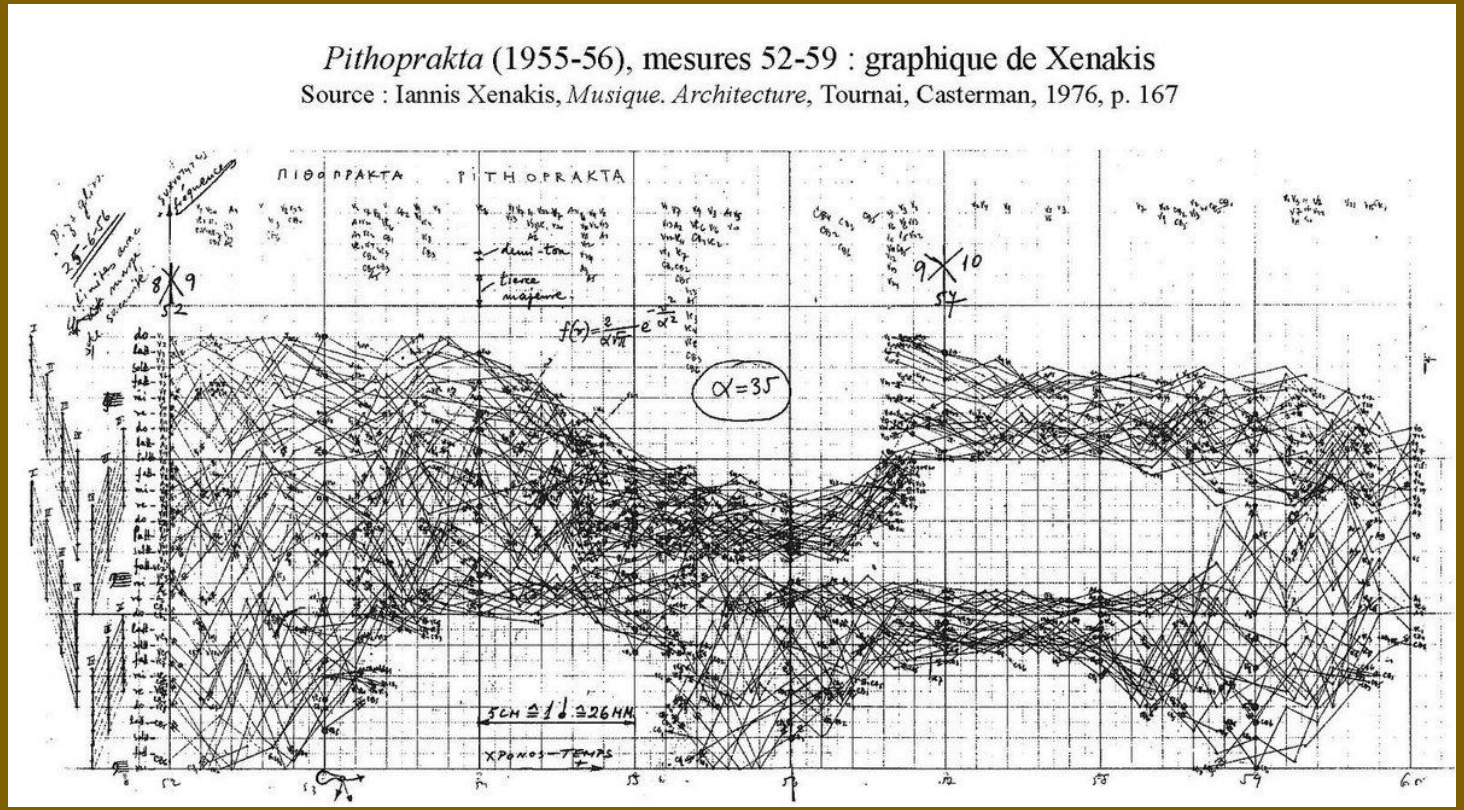
Πιθανότητες

Καθορίζουν:

- πυκνότητα γεγονότων
- διάρκεια
- δυναμικές.

Ξενάκης - Pithoprakta

Pithoprakta (1955-56), mesures 52-59 : graphique de Xenakis
Source : Iannis Xenakis, *Musique. Architecture*, Tournai, Casterman, 1976, p. 167



Ξενάκης - Pithoprakta

Η βασική ιδέα του Ξενάκη ήταν:

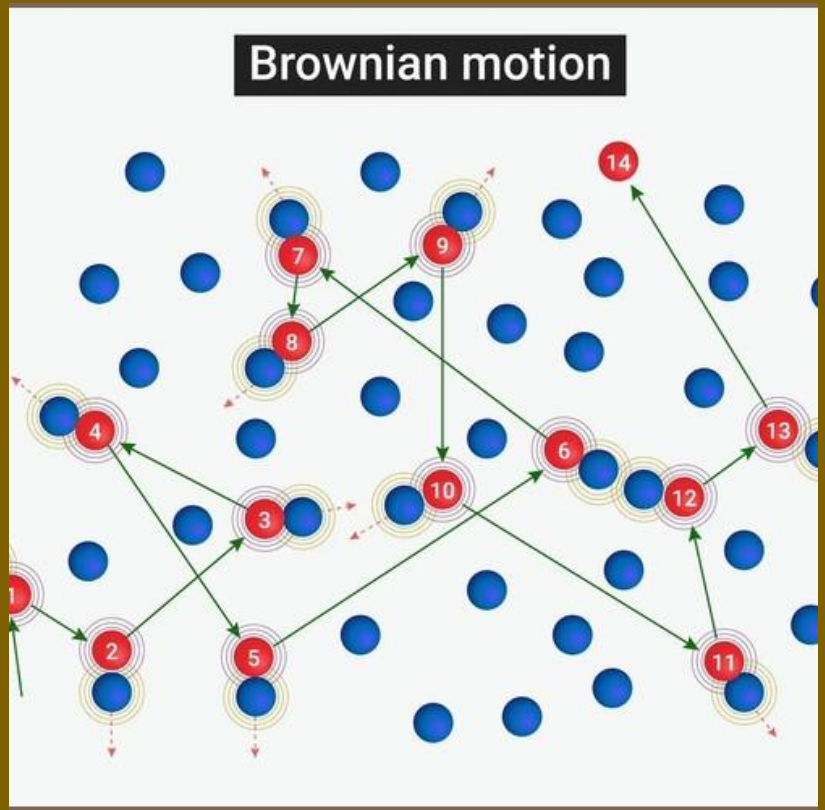
Να συνθέτει τη συμπεριφορά μιας ηχητικής μάζας, όχι τις μεμονωμένες νότες.

Brownian motion και Gaussian distribution

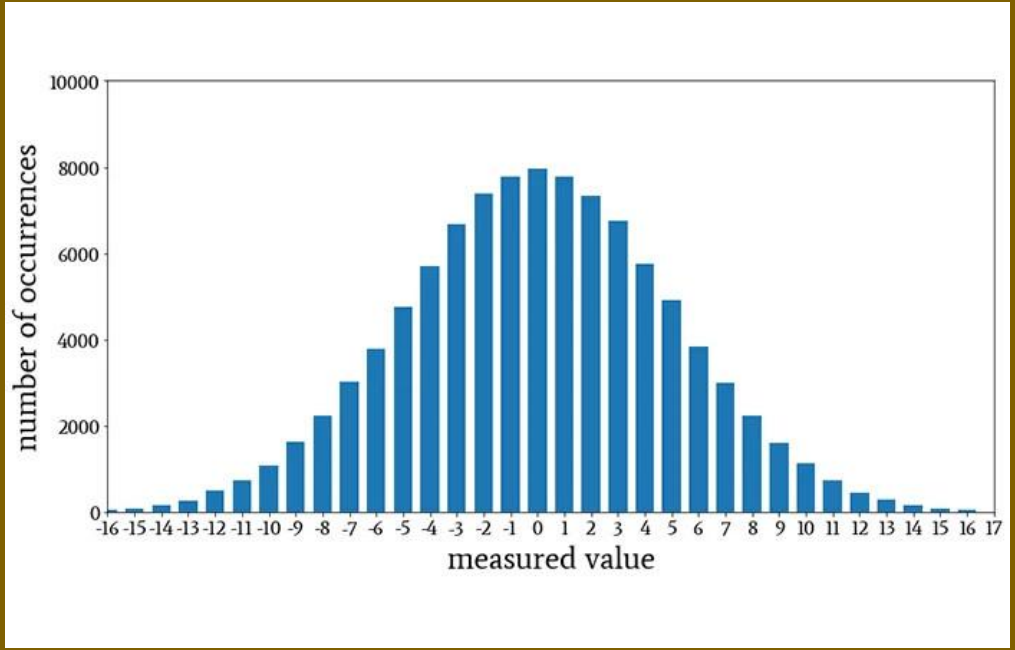
Η **Brownian motion** περιγράφει την **τυχαία κίνηση σωματιδίων** μέσα σε ένα μέσο. Στην αλγοριθμική μουσική αυτή η ιδέα χρησιμοποιείται ώστε μια **νότα** να μετακινείται **σταδιακά και απρόβλεπτα στο ύψος**, δημιουργώντας οργανικές καμπύλες αντί για σταθερές μελωδίες.

Η **Gaussian distribution** (κανονική κατανομή) χρησιμοποιείται για να καθορίσει **πώς κατανέμονται στατιστικά οι τιμές** - π.χ. ποια τονικά ύψη εμφανίζονται πιο **συχνά και ποια σπανιότερα γύρω από ένα κεντρικό τονικό σημείο.**

Brownian motion



Gaussian distribution



Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Στις 9 Αυγούστου 1956, το ***Illiac Suite*** των **Lejaren Hiller and Leonard Isaacson** έκανε πρεμιέρα στο University of Illinois.

Η *Illiac Suite* για κουαρτέτο εγχόρδων αποτελεί ένα σημείο ορόσημο στην αλγοριθμική σύνθεση.

Illiac Suite

Οι Hiller και Isaacson προγραμμάτισαν τον υπολογιστή **ILLIAC I** ώστε να παράγει το μουσικό score αλγοριθμικά, χρησιμοποιώντας αλυσίδες Markov και άλλες στοχαστικές διαδικασίες.

Το έργο τους, που δημοσιεύθηκε στο *Experimental Music* (1959), εγκαινίασε μια νέα εποχή που αυτοματοποιεί τη δημιουργία μουσικών δομών.

Η κρίσιμη μετατόπιση είναι ότι **ο υπολογιστής γίνεται ο βασικός παραγωγός μουσικού υλικού.**

Ο αλγόριθμος εκτελείται αυτόνομα, με την ανθρώπινη παρέμβαση να περιορίζεται κυρίως στον σχεδιασμό κανόνων και στην εκ των υστέρων επιλογή.

Monte Carlo Method - Illiac Suite

Η **Monte Carlo αλγοριθμική μέθοδος** είναι μια τεχνική που χρησιμοποιεί **τυχαίους αριθμούς και πιθανότητες** για να λύσει προβλήματα ή να δημιουργήσει δομές.

Το όνομα προέρχεται από το **Monte Carlo**, διάσημο για τα καζίνο - επειδή η μέθοδος βασίζεται στην **τυχαιότητα**.

Βασική ιδέα

Αν ένα πρόβλημα είναι πολύ περίπλοκο να λυθεί ακριβώς, τότε:

1. δημιουργείς **πολλές τυχαίες δοκιμές**
2. παρατηρείς τα αποτελέσματα
3. υπολογίζεις **στατιστικά το αποτέλεσμα**

Όσο περισσότερες δοκιμές κάνεις, τόσο πιο κοντά έρχεσαι στην πραγματική λύση.

Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Ο **Ξενάκης** ήταν από τους πρώτους συνθέτες που ενσωμάτωσαν άμεσα τους υπολογιστές στη συνθετική πρακτική.

Ανέπτυξε ένα στοχαστικό πρόγραμμα σε FORTRAN για τον κεντρικό υπολογιστή IBM-7090 στο Παρίσι και το χρησιμοποίησε για πολύπλοκους υπολογισμούς σε έργα όπως το ***Amorsima–Morsima*** (1962).

Την ίδια περίοδο, ο **Gottfried Michael Koenig** (Μίχαελ Καίνιχ) ανέπτυξε τα *Project 1* και *Project 2*, πρωτοποριακά προγράμματα υπολογιστικής σύνθεσης, τα οποία παρήγαν αλγοριθμικά δομικά πλαίσια που στη συνέχεια υλοποιούνταν μέσω ερμηνευτικών οδηγιών.

Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Αξίζει να σημειωθεί ότι μια σημαντική κατεύθυνση αλγοριθμικής σκέψης αναπτύχθηκε παράλληλα στον **Φασματισμό** (**Spectralism**), όπως για παράδειγμα στα έργα του **Tristan Murail** (Τριστάν Μουράιλ) και **Jonathan Harvey.**

Το *Désintégrations* (1982) του Tristan Murail εφαρμόζει μια μορφή αλγοριθμικής ανάλυσης και επανασύνθεσης του οργανικού ηχοχρώματος βασισμένη σε υπολογιστικά μοντέλα ήχου.

Επίσης εισάγει την έννοια της **αποσύνθεσης (“désintégration”) του ήχου** σε βασικά φασματικά στοιχεία και της **ανασύνθεσής τους σε νέα μορφή**.

Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Το *Mortuos Plango, Vivos Voco* (1980) του Jonathan Harvey αποτελεί επίσης ορόσημο αυτής της προσέγγισης, χρησιμοποιώντας πρώιμα υπολογιστικά εργαλεία (MusicV & Chant) για να προσομοιώσει τα φάσματα μιας καμπάνας καθεδρικού ναού και της φωνής ενός παιδιού μέσω αυστηρά ελεγχόμενων συχνοτήτων και μετασχηματισμών, μετατρέποντας φυσικές πηγές ήχου σε αυστηρά δομημένες συνθετικές μορφές.

https://youtu.be/0T-H-fVIHE0?si=HMvRkvWRZ57r0X96

Ο Harvey αναφέρει ότι ήθελε οι **καμπάνες να "μιλήσουν" στη φωνή** και ο υπολογιστής είναι το μέσο για να υπολογίσει **συμμετρίες, συντονισμούς και σχέσεις**, κάτι που θυμίζει τη λογική του Ξενάκη αλλά σε **φασματικό επίπεδο**, όχι στοχαστικό pitch-level.

Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Ο **Curtis Roads** είναι ένας από τους πιο σημαντικούς θεωρητικούς και συνθέτες στον χώρο της **ηλεκτρονικής και αλγοριθμικής μουσικής**, ειδικά γνωστός για την ενασχόλησή του με **microsound και granular synthesis.**

Granular / Microsound Σύνθεση

- Εισήγαγε έννοιες όπως η **microsound**: ήχοι που είναι τόσο σύντομοι που λειτουργούν περισσότερο σαν υφή παρά σαν παραδοσιακές νότες.
- Αναλύει πώς μπορούμε να συνθέσουμε ήχους σε μικρο-χρονική κλίμακα και να τους χτίσουμε σε πιο μεγάλα ηχητικά μοτίβα.

Στοχαστική και Αλγοριθμική Προσέγγιση

- Χρησιμοποιεί **τυχαία ή πιθανοκρατικά στοιχεία** στον σχεδιασμό των ήχων και της μορφής.
- Συνδυάζει μαθηματικά, στατιστική και υπολογιστικά μοντέλα με καλλιτεχνική σύνθεση.

Βιβλία και θεωρία

- Το πιο γνωστό του βιβλίο: *"**The Computer Music Tutorial"**,* που θεωρείται εγκυκλοπαίδεια για την ηλεκτρονική και υπολογιστική μουσική.
- Άλλα σημαντικά έργα: ***"Microsound"***, όπου αναλύει λεπτομερώς granular synthesis και temporal structuring.

Curtis Roads

“Point Line Cloud” (1986)

Landmark έργο που χρησιμοποιεί **πυκνές υφές ήχων**, όπου ήχοι μοιάζουν με “σύννεφα” από σημεία (points - grains).

Αλγοριθμική προσέγγιση:

- Το πρόγραμμα καθορίζει ένα **σύνολο πιθανοκρατικών σημείων** στον χρόνο και στο φάσμα συχνοτήτων.
- Κάθε σημείο αντιστοιχεί σε ένα μικρο-ήχο (grain).
- Ο Roads χρησιμοποιεί **στοχαστικά φίλτρα** για να δημιουργήσει την κίνηση και την εξέλιξη της υφής.

https://youtu.be/jOzOZvoyUJ4?si=6njhgahZ90HEjgfC

Curtis Roads

Στοχαστικά Φίλτρα

Probability Density Functions (PDFs)

Spectral Filtering (Στοχαστικό Φιλτράρισμα Φάσματος)

Επιλογή ποια frequencies “επιτρέπονται”.

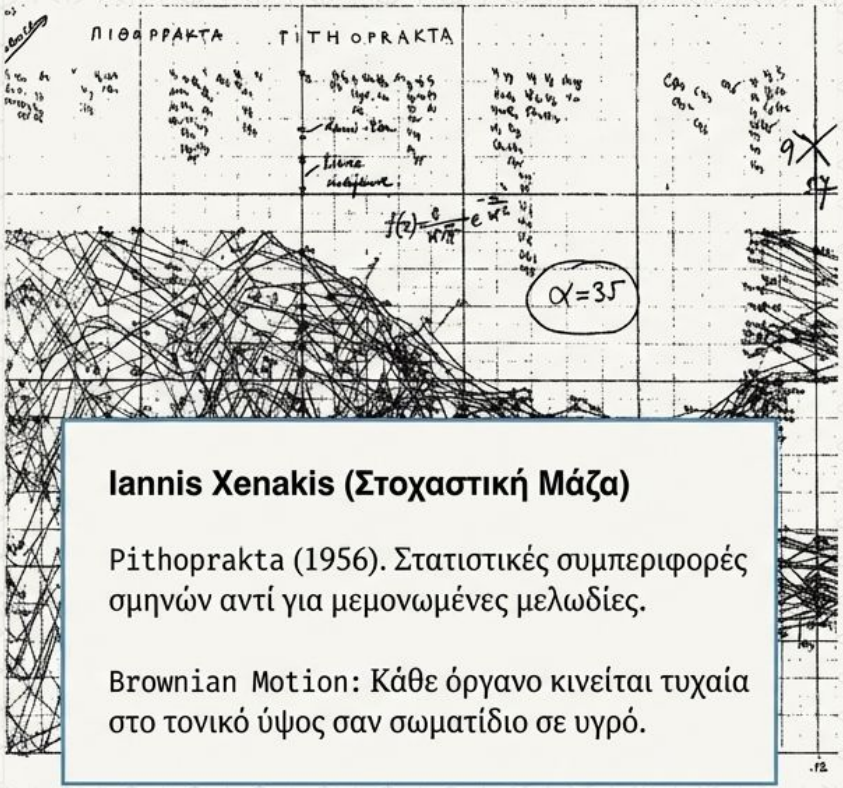
- Π.χ.:
  - 70% grains → mid frequencies
  - 20% → highs
  - 10% → lows

Spatial Stochasticity

Τοποθέτηση σε multi-speaker space.

- Κάθε grain:
  - random pan
  - αλλά με bias (π.χ. προς τα δεξιά)

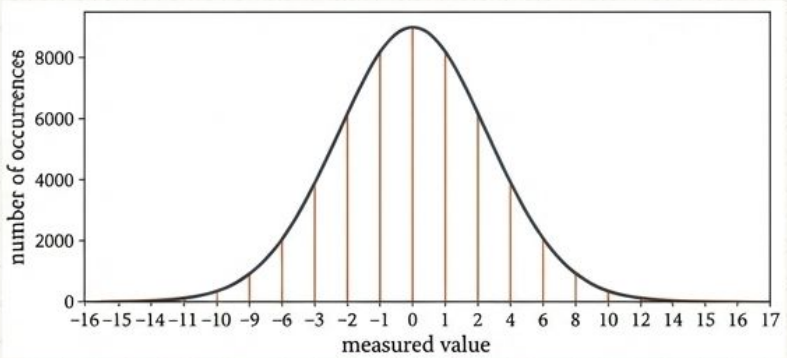
Ήχος ως Μάζα vs. Μικροσωματίδιο



Iannis Xenakis (Στοχαστική Μάζα)

Pithoprakta (1956). Στατιστικές συμπεριφορές σμηνών αντί για μεμονωμένες μελωδίες.

Brownian Motion: Κάθε όργανο κινείται τυχαία στο τονικό ύψος σαν σωματίδιο σε υγρό.



Curtis Roads (Microsound)

Point Line Cloud (1986). Ήχοι διάρκειας 1-50ms που λειτουργούν ως υφή.

Στοχαστικά Φίλτρα (Gaussian): Καθορίζουν πού πέφτουν οι ήχοι στο φάσμα, δημιουργώντας μουσική βαρύτητα.

Ιστορική αναδρομή αλγοριθμικής σύνθεσης

Ο Αλγόριθμος ως Ερμηνευτής (2010s–σήμερα)

Η εμφάνιση του **Deep Learning** σηματοδοτεί μια θεμελιώδη **ρήξη** στην αλγοριθμική πρακτική του ήχου.

Τα σύγχρονα συστήματα μηχανικής μάθησης δεν εκτελούν απλά προκαθορισμένους κανόνες, αλλά **μαθαίνουν μοτίβα από μεγάλα σύνολα δεδομένων** ώστε να αναγνωρίζουν, να κατηγοριοποιούν και να μεταφράζουν ήχο.

Πρόκειται για ποιοτική μετατόπιση: ο αλγόριθμος πλέον αποτελεί μια μορφή διακριτής, μη ανθρώπινης αντίληψης, όπου **το ηχητικό περιβάλλον δεν αντιμετωπίζεται πλέον ως υλικό προς μετασχηματισμό αλλά ως δεδομένα προς στατιστική ερμηνεία.**

Αλγόριθμος ως Ερμηνευτής

Νέο είδος “στοχαστικότητας”

Πριν:

- **randomness = probability**

Τώρα:

- **randomness = sampling από latent space**

Δεν είναι απλά τυχαίο.
Είναι πιθανό με βάση τα datasets που εκπαιδευτηκε.

Μερος 2

Μεθοδολογία

Μεθοδολογία

Algorithmic Idea

↓

AI Assisted Coding

↓

Development in VS Code

↓

Browser Testing

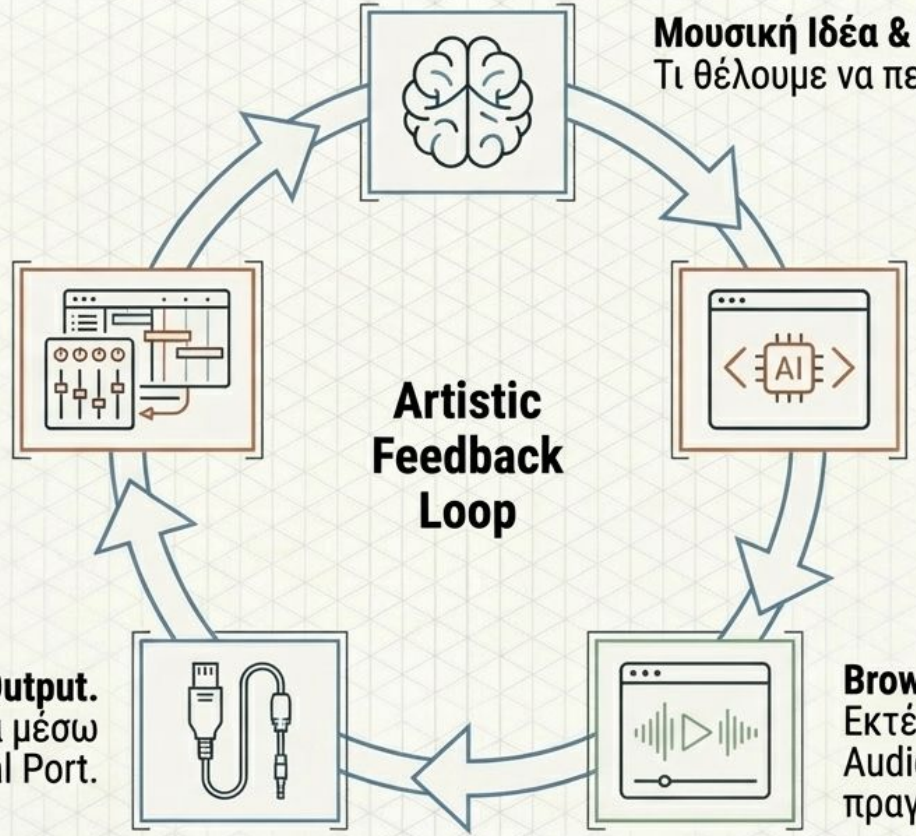
↓

MIDI Output

↓

Control of DAW Instrument

Το Σύγχρονο Σύστημα: Από την Ιδέα στο DAW



Μουσική Ιδέα & Κανόνες.
Τι θέλουμε να πετύχουμε;

AI Assisted Coding.
Fira Cod Coding.
Το AI ως Generator, ο συνθέτης ως Director.

Browser Testing.
Εκτέλεση του Web Audio/MIDI σε πραγματικό χρόνο.

MIDI Output.
Επικοινωνία μέσω Virtual Port.

DAW Instrument.
Ακουστική αξιολόγηση και επανασχεδιασμός.

Artistic Feedback Loop

Μεθοδολογία

Η διαδικασία ξεκινά από μια **μουσική ιδέα που μπορεί να μετατραπεί σε αλγόριθμο**.

Παραδείγματα:

- Markov chains → μελωδίες με πιθανότητες
- Euclidean rhythms → ισορροπημένοι ρυθμοί
- Perlin noise → οργανικές μελωδικές κινήσεις
- Phase shifting → σταδιακή μετατόπιση ρυθμών

Η βασική ερώτηση είναι:

Ποια μουσική διαδικασία μπορεί να περιγραφεί με κανόνες;

Ο αλγόριθμος γίνεται το γεννητικό σύστημα της μουσικής.

Μεθοδολογία

Η υλοποίηση του αλγορίθμου γίνεται με **AI-assisted programming**.

Χρησιμοποιούνται εργαλεία όπως:

- OpenAI Codex
- Claude
- Chat-based AI tools

Η χρήση AI βοηθά σε:

- γρήγορη μετατροπή ιδεών σε κώδικα
- debugging
- πειραματισμό με παραλλαγές αλγορίθμων

Ο καλλιτέχνης λειτουργεί ως:

Designer του συστήματος και επιμελητής του αποτελέσματος.

Μεθοδολογία

Development Environment

Η ανάπτυξη γίνεται στο:

π.χ. **Visual Studio** (editor)

Εκεί γράφεται ο κώδικας:

- JavaScript (algorithm)
- HTML (interface)
- CSS (visual feedback)

Η δομή ενός project είναι συνήθως:

project folder
index.html
script.js
style.css

Το περιβάλλον αυτό επιτρέπει:

- γρήγορη επεξεργασία κώδικα
- debugging
- άμεση δοκιμή στον browser

Μεθοδολογία

Browser Testing

Ο κώδικας εκτελείται στον browser μέσω:

- **JavaScript**
- **Web Audio API**
- **Web MIDI API**

Ο browser λειτουργεί σαν:

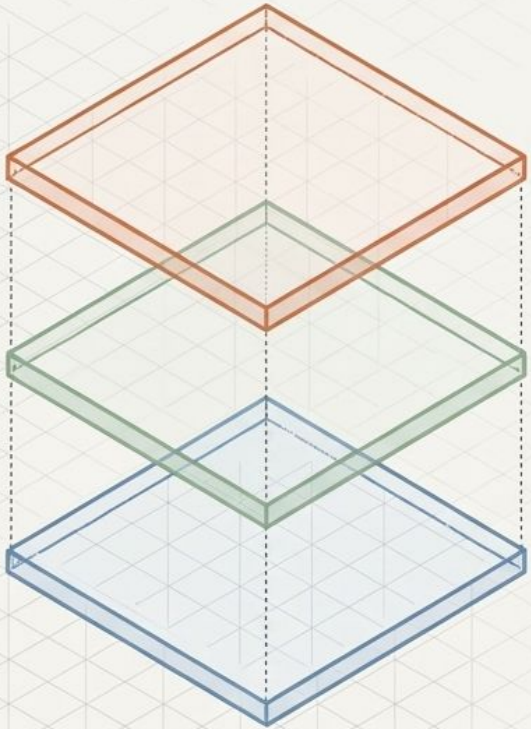
Real-time generative music engine.

Κατά το testing ελέγχουμε:

- αν ο αλγόριθμος παράγει σωστά patterns
- αν το interface ανταποκρίνεται (sliders, buttons)
- αν το timing είναι σωστό

Η διαδικασία είναι **πειραματική και επαναληπτική**.

Η Αρχιτεκτονική του Browser ως Synthesizer



Control Layer (HTML/CSS)

Sliders, buttons, visual feedback. Η διαδραστικότητα του χρήστη.

Logic Layer (JavaScript)

Ο αλγόριθμος. Markov chains, Euclidean rhythms, chaos logic.

Engine Layer (Web Audio / Web MIDI API)

Δίκτυο από nodes (Oscillators → Filters → Gain → Speakers/MIDI).

Περιορισμοί: Το timing και το latency του browser (jitter) και η απουσία πρόσβασης σε low-level hardware.

Μεθοδολογία

MIDI Communication

Η μουσική πληροφορία στέλνεται επιλεκτικά σε εξωτερικά εργαλεία μέσω MIDI.

Με αυτόν τον τρόπο ο αλγόριθμος δεν παράγει απλά ήχο, αλλά **ελέγχει εξωτερικά όργανα**.

Χρησιμοποιείται το:

Web MIDI API

Ο browser μπορεί να στείλει:

Note On
Note Off
Velocity
Control Changes

Μεθοδολογία

Connection to DAW

Το MIDI στέλνεται στο **Ableton Live η σε άλλο DAW η σε εξωτερικό MIDI device**.

Η σύνδεση γίνεται μέσω:

macOS IAC Driver

Workflow:

Browser Algorithm
↓
Web MIDI Output
↓
IAC Driver
↓
Ableton MIDI Track
↓
Instrument / Synth

Μεθοδολογία

Artistic Feedback Loop

Η διαδικασία είναι **κυκλική**.

Idea
↓
Code
↓
Listen
↓
Modify Algorithm
↓
New Musical Result

Η τελική μουσική μορφή προκύπτει από:

- αλγοριθμικό σχεδιασμό
- ακουστική αξιολόγηση
- συνεχή επανασχεδιασμό του συστήματος

MIDI Alternative

OSC Open Sound Control

Παραδείγματα ιδεών σε λογικές διαδικασίες.

Feedback Composition

Λογική διαδικασία

1. Παίξε μια νότα.
2. Θυμήσου τις τελευταίες 5 νότες που παίχτηκαν.
3. Αν κάποια νότα επαναλαμβάνεται πολύ συχνά:
    - άλλαξε κλίμακα ή αρμονία.
4. Αν οι νότες είναι πολύ διαφορετικές:
    - επιβράδυνε τον ρυθμό.

Τι ελέγχει

- μορφή
- πυκνότητα
- εξέλιξη μουσικού υλικού

Παραδείγματα ιδεών σε λογικές διαδικασίες.

Probability Fields

Λογική διαδικασία

1. Κάθε νότα έχει μια πιθανότητα να παιχτεί.
2. Μετά από κάθε νότα:
    - αύξησε την πιθανότητα κοντινών νοτών.
    - μείωσε την πιθανότητα μακρινών νοτών.
3. Διάλεξε την επόμενη νότα σύμφωνα με αυτές τις πιθανότητες.

Τι ελέγχει

- μελωδική κατεύθυνση
- μουσική «βαρύτητα» γύρω από συγκεκριμένες νότες

Παραδείγματα ιδεών σε λογικές διαδικασίες.

Phase Shifting (τύπου Steve Reich)

Λογική διαδικασία

1. Δημιούργησε ένα μικρό ρυθμικό μοτίβο.
2. Παίξε το ίδιο μοτίβο σε δύο φωνές.
3. Κάθε λίγα δευτερόλεπτα:
    - η δεύτερη φωνή γίνεται ελάχιστα πιο γρήγορη.
4. Τα μοτίβα αρχίζουν να μετατοπίζονται μεταξύ τους.

Τι ελέγχει

- ρυθμικές μετατοπίσεις
- πολυφωνία
- evolving patterns

Παραδείγματα ιδεών σε λογικές διαδικασίες.

Fractal Music

Λογική διαδικασία

1. Δημιούργησε ένα μικρό μοτίβο (π.χ. 3 νότες).
2. Αντέγραψε το μοτίβο πολλές φορές.
3. Κάθε αντίγραφο:
    - είναι λίγο πιο γρήγορο
    - είναι λίγο πιο ψηλό
4. Συνέχισε τη διαδικασία σε πολλές «κλίμακες».

Τι ελέγχει

- δομή
- μορφή
- αυτοομοιότητα στη μουσική

Παραδείγματα ιδεών σε λογικές διαδικασίες.

Constraint Based Composition

Λογική διαδικασία

1. Ορίσε περιορισμούς:
   - μόνο νότες της κλίμακας
   - άλματα μέχρι 5 νότες
   - όχι πάνω από 3 επαναλήψεις
2. Δημιούργησε νότες τυχαία.
3. Αν μια νότα παραβιάζει κανόνα:
   - διάλεξε άλλη.

Τι ελέγχει

- μουσικό στυλ
- αρμονία
- μελωδική λογική

Παραδείγματα ιδεών σε λογικές διαδικασίες.

Energy Based Music

Λογική διαδικασία

1. Υπολόγισε την «ενέργεια» της μουσικής:

- πόσες νότες παίζονται
- πόσο δυνατά
- πόσο γρήγορα

2. Αν η ενέργεια είναι υψηλή:
    - πρόσθεσε παύσεις.
3. Αν η ενέργεια είναι χαμηλή:
    - πρόσθεσε περισσότερες νότες.

Τι ελέγχει

- δυναμική
- ένταση
- δραματουργία

Παραδείγματα ιδεών σε λογικές διαδικασίες.

Soundscape Driven Music (AI Classification) (Segments)

Λογική διαδικασία

1. Άκου τον περιβάλλοντα ήχο.
2. Αναγνώρισε τι υπάρχει:

- πουλιά
- ομιλία
- αυτοκίνητα
- νερό

3. Κάθε κατηγορία ελέγχει διαφορετικό όργανο.

Τι ελέγχει

- instrumentation
- μουσική δομή
- αλληλεπίδραση με το περιβάλλον

Feedback Composition example

Ενα σύστημα που ακούει τον εαυτό του και αλλάζει συμπεριφορά με βάση το πρόσφατο παρελθόν.

Βασικός μηχανισμός

1. Παίζεται μια νότα.
2. Η νότα αποθηκεύεται σε μια **μνήμη 5 νοτών**.
3. Το σύστημα αναλύει τη μνήμη.
4. Με βάση τα αποτελέσματα αλλάζει τη συμπεριφορά.

Παράδειγμα Feedback Composition

Ψευδοκώδικας

```
memory = last 5 notes

play(note)

add note to memory

if same_note_frequency(memory) > threshold:

   change_scale()

if pitch_variance(memory) > threshold:

   slow_tempo()
```

Feedback Composition example

Ανάλυση των κανόνων

1. Αν μια νότα επαναλαμβάνεται συχνά

Το σύστημα αντιδρά για να αποφύγει στασιμότητα.

Πιθανές αντιδράσεις:

- αλλαγή κλίμακας
- μεταφορά τονικότητας
- αλλαγή αρμονικού κέντρου
- εισαγωγή νέου interval

Feedback Composition example

Αν οι νότες είναι πολύ διαφορετικές

Δηλαδή μεγάλη απόσταση μεταξύ pitches.

Παράδειγμα:

C F# Eb A Bb

Το σύστημα αντιδρά:

- μειώνει tempo
- αυξάνει διάρκεια νοτών
- αφαιρεί νότες

Javascript

Η **JavaScript** είναι πρακτική γλώσσα για αλγοριθμική μουσική επειδή λειτουργεί **απευθείας μέσα στον browser** χωρίς εγκατάσταση ειδικού λογισμικού.

Αυτό σημαίνει ότι ένα μουσικό εργαλείο μπορεί να τρέξει σε:

- laptop
- κινητό
- tablet

Javascript - **Πλεονεκτήματα**

1. Άμεσος ήχος στον browser

Με το **Web Audio API** μπορούμε να:

- δημιουργήσουμε synthesizers
- επεξεργαστούμε ήχο
- αναλύσουμε ήχο σε πραγματικό χρόνο

2. Σύνδεση με μουσικά εργαλεία

Με το **Web MIDI API** μπορεί να:

- στείλει MIDI νότες
- ελέγξει synthesizers
- συνδεθεί με DAWs όπως το **Ableton Live**

3. Διαδραστικά Interfaces

Συνεργάζεται άμεσα με **HTML και CSS**, επιτρέποντας τη δημιουργία:

- sliders
- sequencers
- visual feedback
- interactive music systems

Javascript - **Περιορισμοί**

Timing και Latency

Οι browsers δεν είναι σχεδιασμένοι κυρίως για **ακριβές μουσικό timing**.

Αυτό σημαίνει ότι:

- το timing μπορεί να έχει μικρές αποκλίσεις
- δύσκολα επιτυγχάνεται sample-accurate συγχρονισμός
- σε πολύ γρήγορους ρυθμούς μπορεί να εμφανιστεί jitter

Αν και το **Web Audio API** βελτιώνει το timing, εξακολουθεί να μην είναι τόσο ακριβές όσο σε native audio environments.

Javascript - Περιορισμοί

Περιορισμένη απόδοση σε βαριά επεξεργασία

Η JavaScript εκτελείται μέσα στον browser και όχι απευθείας στο σύστημα.

Αυτό σημαίνει ότι:

- μεγάλα DSP processes μπορεί να είναι βαριά
- σύνθετοι αλγόριθμοι ή machine learning μπορεί να επιβαρύνουν την CPU
- το performance εξαρτάται από τον browser και τη συσκευή

Περιορισμένη πρόσβαση στο σύστημα

Η JavaScript λειτουργεί σε ένα **ασφαλές περιβάλλον (sandbox)**.

Άρα δεν έχει πλήρη πρόσβαση σε:

- hardware
- drivers
- χαμηλού επιπέδου audio συστήματα

Τα Εργαλεία μας

Visual Studio

https://code.visualstudio.com/download

DeepSeek/Kimi/Claude/OpenAI Codex

Google Chrome

Web Technologies

HTML → δημιουργεί το interface

CSS → στυλ HTML

JavaScript → υλοποιεί τον αλγόριθμο

Web Audio API → δημιουργεί και επεξεργάζεται ήχο

Σύνδεση JavaScript με HTML Controls 1

<input type="range" id="tempoSlider" min="60" max="180" value="120">

<!-- δημιουργεί ένα slider από 60 έως 180 BPM -->

let tempo = 120;

// μεταβλητή που κρατά την τιμή του tempo

Σύνδεση JavaScript με HTML Controls 2

```
let slider = document.getElementById("tempoSlider");

// βρίσκουμε το slider μέσα στο HTML

slider.oninput = function(){

// κάθε φορά που ο χρήστης μετακινεί το slider

 tempo = slider.value;

  // παίρνουμε τη νέα τιμή από το slider

 console.log("Tempo:", tempo);

  // εμφανίζουμε την τιμή για έλεγχο

}
```

Μακροσκοπικά

User Interaction

(sliders, buttons)

↓

HTML Interface

↓

JavaScript Algorithm

↓

Web Audio Engine

↓

Sound Output

(speakers / headphones)

Web Audio API

Το **Web Audio API** είναι ένα σύστημα της JavaScript που επιτρέπει στον browser να:

- δημιουργεί ήχο
- επεξεργάζεται ήχο
- αναλύει ήχο
- ελέγχει ήχο σε πραγματικό χρόνο

Με απλά λόγια:

Browser = Synthesizer + Audio Processor

Χρησιμοποιείται για:

- web synthesizers
- generative music
- interactive installations
- games
- sound art

Web Audio API

Το Web Audio API λειτουργεί σαν **δίκτυο από nodes**.

Sound Source
↓
Audio Processing
↓
Audio Output

Παράδειγμα:

Oscillator → Filter → Gain → Speakers

Κάθε στοιχείο λέγεται **Audio Node**.

Web Audio API

Δημιουργία Audio Context

Το πρώτο βήμα είναι να δημιουργήσουμε το **audio engine**.

```
let audioContext = new AudioContext();
// δημιουργεί το βασικό περιβάλλον ήχου του browser
```

Web Audio API

Δημιουργία Ήχου (Oscillator)

Ένας απλός τρόπος να δημιουργήσουμε ήχο είναι με έναν oscillator.

```
let osc = audioContext.createOscillator();
// δημιουργούμε μια πηγή ήχου

osc.frequency.value = 440;
// ορίζουμε τη συχνότητα (440Hz = νότα Α)

osc.connect(audioContext.destination);
// στέλνουμε τον ήχο στα ηχεία

osc.start();
// ξεκινά ο ήχος
```

Web Audio API

Έλεγχος Έντασης (Gain Node)

Η ένταση ελέγχεται με ένα **Gain Node**.

```
let gain = audioContext.createGain();
// δημιουργούμε node ελέγχου έντασης

gain.gain.value = 0.2;
// ορίζουμε ένταση (0–1)

osc.connect(gain);
// συνδέουμε oscillator → gain

gain.connect(audioContext.destination);
// gain → speakers
```

Web Audio API

Φίλτρα Ήχου

Παράδειγμα φίλτρου:

```
let filter = audioContext.createBiquadFilter();
// δημιουργούμε φίλτρο

filter.type = "lowpass";
// αφήνει να περάσουν μόνο χαμηλές συχνότητες

filter.frequency.value = 800;
// cutoff frequency
```

Web Audio API

Ανάλυση Ήχου

Το Web Audio API μπορεί να **αναλύει ήχο σε πραγματικό χρόνο**.

```
let analyser = audioContext.createAnalyser();
// δημιουργούμε analyser node

let data = new Uint8Array(analyser.frequencyBinCount);
// πίνακας για δεδομένα συχνοτήτων

analyser.getByteFrequencyData(data);
// γεμίζουμε τον πίνακα με πληροφορίες συχνοτήτων
```

Δημιουργία Virtual MIDI Port στο macOS

Το macOS έχει ενσωματωμένο εργαλείο (Windows LoopMIDI app).

Άνοιξε:

Audio MIDI Setup

Βήματα:

1. Applications → Utilities → Audio MIDI Setup
2. Menu → **Window** → **Show MIDI Studio**
3. Διπλό click στο **IAC Driver**
4. Ενεργοποίησε:

Device is online

5. Δημιούργησε ένα port π.χ.

WebMIDI

Αυτό μετά γίνεται το **virtual MIDI cable** μεταξύ browser και Ableton.

Δημιουργία Virtual MIDI Port στο macOS

Ρύθμιση στο Ableton Live

Άνοιξε:

Ableton Live

Settings → **Link / Tempo / MIDI**

Στην περιοχή **MIDI Ports**

Βρες:

IAC Driver → WebMIDI

Ενεργοποίησε:

Track = ON
Remote = ON

Δημιουργία Virtual MIDI Port στο macOS

Ενεργοποίηση Web MIDI

Η JavaScript πρέπει να ζητήσει πρόσβαση στο MIDI.

```
navigator.requestMIDIAccess().then(startMIDI);
// ζητάμε πρόσβαση στα MIDI devices του συστήματος
```

Δημιουργία Virtual MIDI Port στο macOS

Επιλογή MIDI Output

```
function startMIDI(midi){

 let outputs = midi.outputs.values();
 // παίρνουμε όλα τα διαθέσιμα MIDI outputs

 let output = outputs.next().value;
 // επιλέγουμε το πρώτο (συνήθως το IAC port)

 sendNote(output);
 // στέλνουμε μια δοκιμαστική νότα

}
```

Δημιουργία Virtual MIDI Port στο macOS

Αποστολή MIDI Note

```
function sendNote(output){

 let noteOn = [0x90, 60, 100];
 // 0x90 = note on
 // 60 = MIDI note (Middle C)
 // 100 = velocity

 output.send(noteOn);
 // στέλνουμε το μήνυμα στο Ableton

}
```

Javascript Functions

Τα functions στη JavaScript είναι μπλοκ κώδικα που τα γράφεις μία φορά και τα καλούμε για να εκτελέσουν μια συγκεκριμένη δουλειά.

```
function greet(name) {
  return "Hello " + name;
}

greet("Dimitri"); // "Hello Dimitri"
```

Function = “πάρε κάτι, κάνε κάτι, δώσε κάτι πίσω”

Μετατροπή μουσικών ιδεών σε λογικές διαδικασίες με κώδικα

Random Selection Algorithm

Διαλέγουμε ένα τυχαίο κομμάτι ήχου από μια λίστα.

```
let sounds = ["sound1.wav", "sound2.wav", "sound3.wav"];   // λίστα με διαθέσιμα ηχητικά αρχεία
let randomIndex = Math.floor(Math.random() * sounds.length);  // δημιουργούμε έναν τυχαίο αριθμό από 0 μέχρι το πλήθος των ήχων
let chosenSound = sounds[randomIndex];   // παίρνουμε τον ήχο που βρίσκεται σε αυτή τη θέση στη λίστα
playSound(chosenSound);   // παίζουμε τον επιλεγμένο ήχο
```

Λογική διαδικασία

1. Φτιάχνουμε μια λίστα με ήχους
2. Παράγουμε έναν τυχαίο αριθμό
3. Επιλέγουμε τον ήχο στη θέση αυτού του αριθμού
4. Τον αναπαράγουμε

Μετατροπή μουσικών ιδεών σε λογικές διαδικασίες με κώδικα

Frequency Analysis Algorithm

Αναλύουμε έναν ήχο για να δούμε **πόσο έντονες είναι οι συχνότητές του**.

let analyser = audioContext.createAnalyser(); // δημιουργούμε ένα εργαλείο που μπορεί να αναλύσει τον ήχο
let dataArray = new Uint8Array(analyser.frequencyBinCount); // δημιουργούμε έναν πίνακα που θα αποθηκεύσει τις τιμές των συχνοτήτων
analyser.getByteFrequencyData(dataArray); // γεμίζουμε τον πίνακα με τις πραγματικές τιμές συχνοτήτων από τον ήχο
let average = dataArray.reduce((a,b)=>a+b) / dataArray.length; // υπολογίζουμε τον μέσο όρο της έντασης όλων των συχνοτήτων
console.log(average); // εμφανίζουμε το αποτέλεσμα για να ξέρουμε πόσο "έντονος" είναι ο ήχος

Λογική διαδικασία

1. Παίρνουμε δεδομένα από τον ήχο
2. Μετράμε τις συχνότητες
3. Υπολογίζουμε τον μέσο όρο τους
4. Χρησιμοποιούμε την πληροφορία για επιλογές στον αλγόριθμο

Μετατροπή μουσικών ιδεών σε λογικές διαδικασίες με κώδικα

Rule-Based Selection

Επιλέγουμε ήχο **ανάλογα με το αποτέλεσμα της ανάλυσης**.

```
if(average > 150){   // αν ο ήχος έχει συχνότητες μεγαλύτερες από…
  playSound("loud_segment.wav");   // παίζουμε ένα δυνατό ηχητικό κομμάτι
} else {   // αν ο ήχος είναι πιο ήπιος
  playSound("soft_segment.wav");   // παίζουμε ένα πιο ήσυχο κομμάτι
}
```

Λογική διαδικασία

1. Μετράμε τον ήχο
2. Ελέγχουμε μια συνθήκη
3. Αν ισχύει → παίζουμε συγκεκριμένο ήχο
4. Αν όχι → παίζουμε διαφορετικό

Μετατροπή μουσικών ιδεών σε λογικές διαδικασίες με κώδικα

Segment Selection Algorithm

Παίρνουμε **μικρά κομμάτια από πολλά αρχεία** και φτιάχνουμε νέο track.

```
let segments = [];   // δημιουργούμε μια άδεια λίστα για να αποθηκεύσουμε κομμάτια ήχου

for(let i=0; i<10; i++){   // επαναλαμβάνουμε τη διαδικασία 10 φορές
let randomSound = sounds[Math.floor(Math.random()*sounds.length)];   // διαλέγουμε τυχαίο αρχείο ήχου
let segment = cutRandomSegment(randomSound);   // κόβουμε ένα μικρό τυχαίο κομμάτι από αυτό
segments.push(segment);   // προσθέτουμε το κομμάτι στη λίστα
}

playSequence(segments);   // παίζουμε όλα τα κομμάτια διαδοχικά σαν νέο track
```

Λογική διαδικασία

1. Διαλέγουμε τυχαίο αρχείο
2. Κόβουμε μικρό τμήμα
3. Το αποθηκεύουμε
4. Επαναλαμβάνουμε
5. Παίζουμε όλα τα τμήματα μαζί

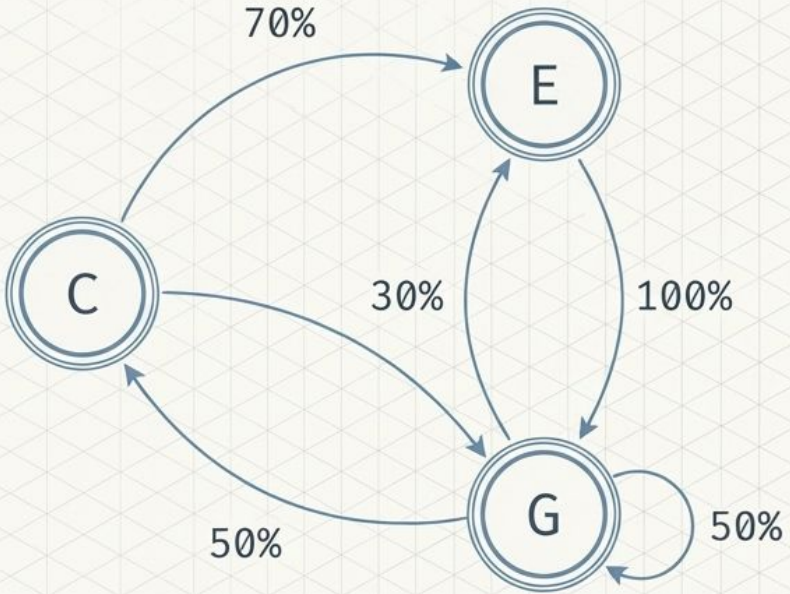
Markov Chains

Είναι μαθηματικά μοντέλα που **περιγράφουν τη μετάβαση από μία κατάσταση σε άλλη**, όπου **η πιθανότητα της επόμενης κατάστασης εξαρτάται *αποκλειστικά* από την τρέχουσα κατάσταση** και όχι από το ιστορικό (1st order).

Βασικά Χαρακτηριστικά:

- **Καταστάσεις (States):** Μια σειρά διακριτών καταστάσεων (π.χ. Ηλιόλουστος, Βροχερός).
- **Πιθανότητες Μετάβασης (Transition Probabilities):** Οι πιθανότητες μετάβασης από μία κατάσταση σε μια άλλη.
- **Ιδιότητα Markov:** Το μέλλον εξαρτάται μόνο από το παρόν.
- **Πίνακας Μετάβασης (Transition Matrix):** Ένας πίνακας που αποθηκεύει τις πιθανότητες μετάβασης, όπου το άθροισμα των πιθανοτήτων κάθε σειράς ισούται με 1.

Οπτική Γκαλερί: Markov Chains (Πιθανότητες Μετάβασης)



Concept

Στοχαστική διαδικασία με «έλλειψη μνήμης». Η επόμενη κατάσταση εξαρτάται μόνο από την τρέχουσα.

Musical Application

Δημιουργία μελωδιών που δεν είναι εντελώς τυχαίες. Αναλύοντας ένα υπάρχον κομμάτι, εξάγουμε πιθανότητες μετάβασης νοτών. Παράγει νέο υλικό που διατηρεί τη μουσική συνέχειά και το στυλ του πρωτοτύπου.

Μετατροπή μουσικών ιδεών σε λογικές διαδικασίες με κώδικα

Markov Chain Melody

Η επόμενη νότα εξαρτάται από την **προηγούμενη νότα**.
Έτσι δημιουργείται μουσική με **πιθανότητες μετάβασης**.

```
let transitions = {                  // πίνακας που λέει ποιες νότες μπορούν να ακολουθήσουν άλλες
 C: ["E","G","D"],                   // μετά το C μπορούμε να πάμε σε E, G ή D
 D: ["F","A"],                      // μετά το D μπορούμε να πάμε σε F ή A
 E: ["G","C"],                      // μετά το E μπορούμε να πάμε σε G ή C
 G: ["C","D"]                       // μετά το G μπορούμε να πάμε σε C ή D
};

let currentNote = "C";               // ξεκινάμε από τη νότα C

function nextNote(){                 // συνάρτηση που βρίσκει την επόμενη νότα
 let possible = transitions[currentNote];     // βρίσκουμε τις πιθανές επόμενες νότες
 let randomIndex = Math.floor(Math.random()*possible.length);  // επιλέγουμε τυχαία μία από αυτές
 currentNote = possible[randomIndex];         // η επιλεγμένη νότα γίνεται η νέα τρέχουσα νότα
 playNote(currentNote);                       // παίζουμε τη νέα νότα
}
```

Λογική διαδικασία

1. Ορίζουμε πιθανές μεταβάσεις μεταξύ νοτών
2. Ξεκινάμε από μία νότα
3. Επιλέγουμε τυχαία την επόμενη από τις πιθανές
4. Επαναλαμβάνουμε

Euclidean Rhythm

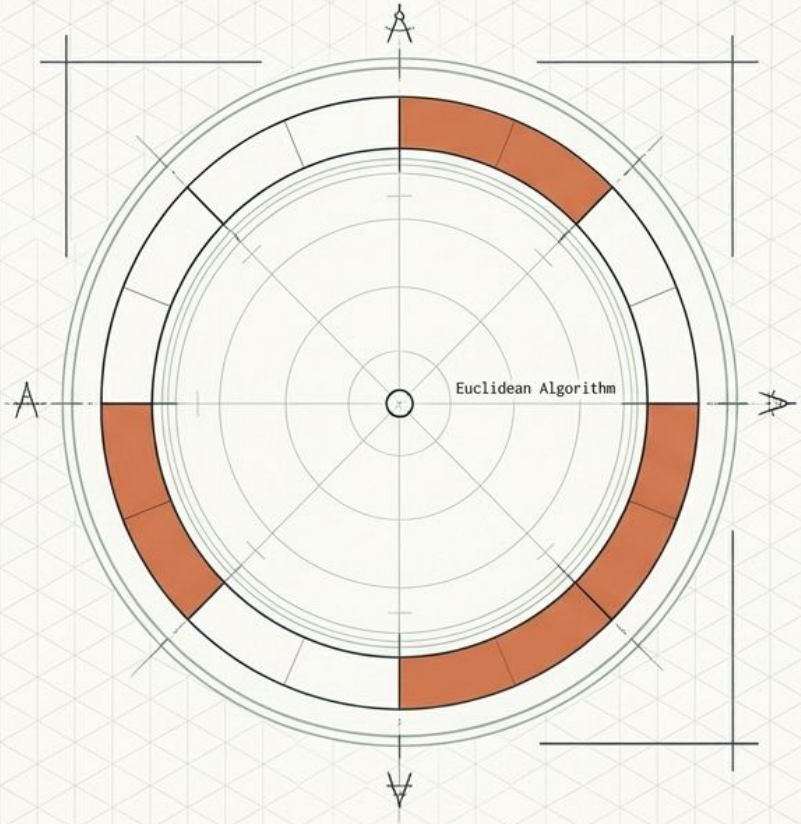
Τα **Euclidean Rhythms** είναι χρήσιμα στη μουσική επειδή επιτρέπουν την **ομοιόμορφη κατανομή ενός αριθμού χτυπημάτων μέσα σε έναν ρυθμικό κύκλο**, δημιουργώντας φυσικά και ισορροπημένα patterns. **Βασίζονται στον «Ευκλείδειο αλγόριθμο» για την εύρεση του μέγιστου κοινού διαιρέτη.**

Ο αλγόριθμος βρίσκει τον πιο **δίκαιο** τρόπο να τοποθετήσει π.χ. 3, 5 ή 7 χτυπήματα μέσα σε 8, 12 ή 16 βήματα.

- Χρησιμοποιούν 3 παραμέτρους:
- **Συνολικά βήματα** (**steps**),
- **Αριθμός χτύπων** (**pulses**)και
- **Offset** (μετατόπιση).
- **Ισόποση Κατανομή:** Αν δεν διαιρούνται τέλεια (π.χ. 3 χτύποι σε 8 βήματα), ο αλγόριθμος κατανέμει τα χτυπήματα όσο πιο ομοιόμορφα γίνεται, δημιουργώντας ενδιαφέροντες, φυσικούς ρυθμούς.

Επειδή είναι απλός αλλά πολύ ευέλικτος, χρησιμοποιείται συχνά σε **generative music systems, drum machines και algorithmic composition**, επιτρέποντας τη δημιουργία σύνθετων ρυθμών με λίγες μόνο παραμέτρους.

Οπτική Γκαλερί: Euclidean Rhythms (Μαθηματική Κατανομή)



Concept

Βασισμένο στον Ευκλείδειο αλγόριθμο (μέγιστος κοινός διαιρέτης) για την απολύτως ομοιόμορφη κατανομή χτυπημάτων.

Musical Application

Ο πιο δίκαιος τρόπος να τοποθετηθούν π.χ. 3, 5, ή 7 χτυπήματα σε 8 ή 16 βήματα. Δημιουργεί οργανικούς, πολύπλοκους πολυρυθμούς με ελάχιστες παραμέτρους (Steps, Pulses, Offset).

Μετατροπή μουσικών ιδεών σε λογικές διαδικασίες με κώδικα

Euclidean Rhythm

Παράδειγμα: 3 χτυπήματα σε 8 βήματα → `x..x..x.`

```
function euclidean(steps, pulses){     // steps = συνολικά βήματα, pulses = πόσα χτυπήματα
 let pattern = [];                     // λίστα που θα κρατήσει το ρυθμό
 for(let i=0; i<steps; i++){           // επαναλαμβάνουμε για όλα τα βήματα
   if((i * pulses) % steps < pulses){  // μαθηματικός τρόπος για ομοιόμορφη κατανομή
     pattern.push(1);                  // 1 σημαίνει χτύπημα
   } else {
     pattern.push(0);                  // 0 σημαίνει παύση
   }
 }
 return pattern;                       // επιστρέφουμε το τελικό μοτίβο
}
```

Διαδικασία

1. Ορίζουμε συνολικά βήματα
2. Ορίζουμε πόσα χτυπήματα θέλουμε
3. Ο αλγόριθμος τα μοιράζει ομοιόμορφα
4. Παίρνουμε ένα ρυθμικό μοτίβο

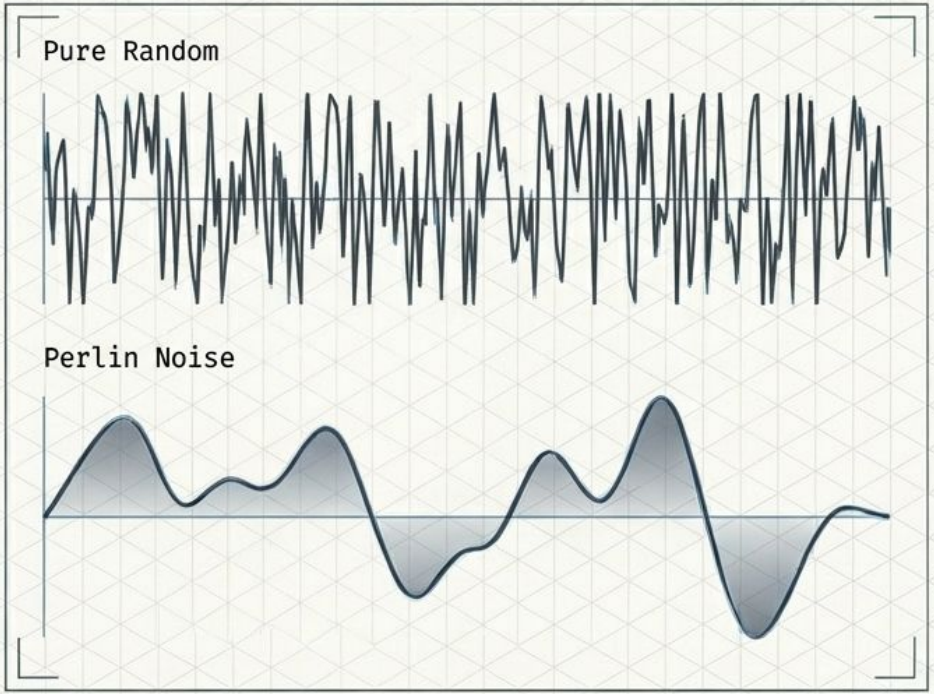
Perlin Noise

Το **Perlin Noise** είναι χρήσιμο στη μουσική επειδή δημιουργεί **ομαλές, συνεχείς τυχαίες μεταβολές**, αντί για απότομα και εντελώς απρόβλεπτα αποτελέσματα όπως το απλό random.

Αυτό το κάνει ιδανικό για generative συστήματα όπου θέλουμε οι αλλαγές σε παραμέτρους όπως **μελωδία, pitch, φίλτρα ή ένταση** να εξελίσσονται με φυσικό τρόπο στον χρόνο.

Το Perlin noise μπορεί να χρησιμοποιηθεί για να παράγει μελωδίες που κινούνται σταδιακά μέσα σε μια κλίμακα ή για να δημιουργεί αργές μεταβολές στον ήχο, δίνοντας την αίσθηση οργανικής εξέλιξης αντί για χαοτική τυχαιότητα.

Οπτική Γκαλερί: Perlin Noise (Οργανική Εξέλιξη)



Concept

Παραγωγή ομαλών, συνεχών τυχαίων μεταβολών στο χρόνο, σε αντίθεση με τα απότομα άλματα της απόλυτης τυχαιότητας.

Musical Application

Ιδανικό για LFOs, αργές μεταβολές φίλτρων (cutoff), pitch drifts, και εξελισσόμενα ηχοχρώματα. Δίνει την αίσθηση ενός συστήματος που "αναπνέει" φυσικά.

Μετατροπή μουσικών ιδεών σε λογικές διαδικασίες με κώδικα

Perlin Noise Melody

Χρησιμοποιούμε **smooth random values** για να δημιουργήσουμε φυσική μελωδία. Σε αντίθεση με το random, οι αλλαγές είναι **ομαλές**.

```
let t = 0;                    // μεταβλητή χρόνου που κινείται αργά
function nextNote(){
 let noiseValue = noise(t);     // παίρνουμε μια τιμή Perlin noise μεταξύ 0 και 1
 let scale = ["C","D","E","G","A"];   // μουσική κλίμακα
 let index = Math.floor(noiseValue * scale.length);   // μετατρέπουμε την τιμή σε θέση στη κλίμακα
 let note = scale[index];       // επιλέγουμε τη νότα
 playNote(note);               // παίζουμε τη νότα
 t += 0.1;                     // μετακινούμαστε λίγο στον "θόρυβο" ώστε η επόμενη τιμή να είναι παρόμοια
}
```

Διαδικασία

1. Παίρνουμε μια ομαλή τυχαία τιμή
2. Τη μετατρέπουμε σε θέση μέσα σε μια κλίμακα
3. Παίζουμε τη νότα
4. Προχωράμε λίγο στον χρόνο για την επόμενη τιμή

Lorenz Attractor

Είναι ένα **σύνολο χαοτικών λύσεων σε ένα τρισδιάστατο μαθηματικό σύστημα διαφορικών εξισώσεων**, που αναπτύχθηκε από τον μετεωρολόγο Edward Lorenz το 1963.

Μοντελοποιεί την ατμοσφαιρική μεταφορά, παρουσιάζοντας μια μορφή που θυμίζει πεταλούδα και αποτελεί **κλασικό παράδειγμα ντετερμινιστικού χάους, όπου μικρές αλλαγές στις αρχικές συνθήκες οδηγούν σε τεράστιες αποκλίσεις.**

- **Χαοτική Συμπεριφορά:** Αν και το σύστημα είναι ντετερμινιστικό (καθορισμένο), η μακροπρόθεσμη συμπεριφορά του είναι απρόβλεπτη.
- **Ευαισθησία στις Αρχικές Συνθήκες:** Δύο σχεδόν πανομοιότυπα σημεία εκκίνησης θα αποκλίνουν εκθετικά γρήγορα, ακολουθώντας εντελώς διαφορετικές διαδρομές.
- **Περιγράφεται απο 3ς παραμέτρους** (σύστημα τριών διαφορικών εξισώσεων), **σ, ρ, β.**

Μουσική Χρήση: 3 παραμέτρους → pitch, pan, volume

Serendipity - Assemblage - Segments

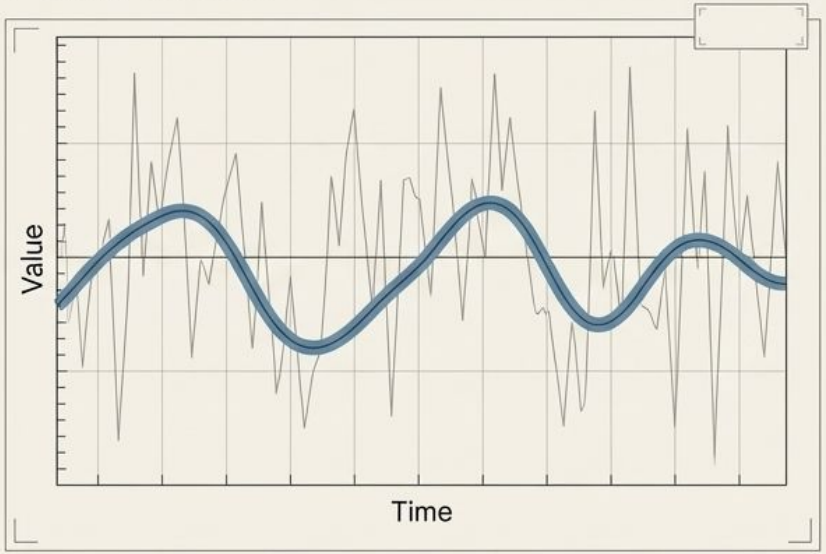
https://ihearcolors.online/serendipity

https://ihearcolors.online/assem/assemblage

https://ihearcolors.online/segments

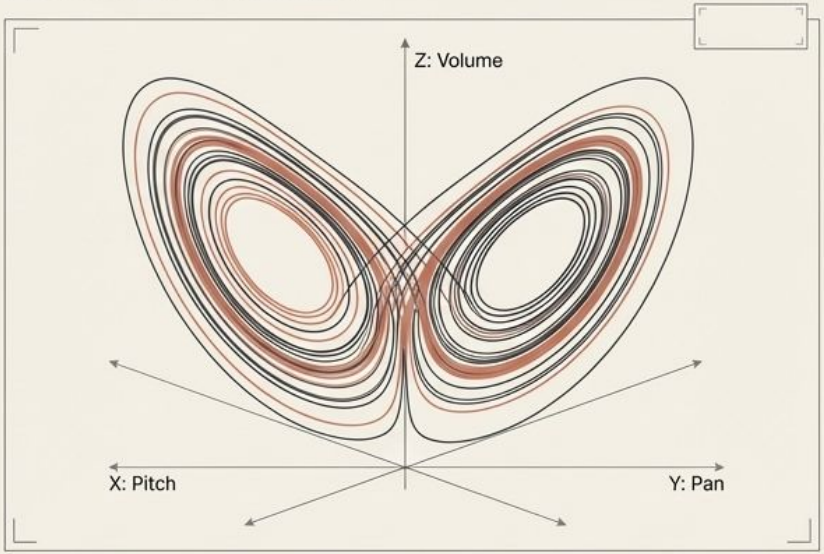
Translating Math to Modulation

Perlin Noise



Organic Evolution
Generates continuous, smooth random values. Ideal for natural-sounding filter sweeps, gentle pitch drifts, and LFOs.

Lorenz Attractor



Deterministic Chaos
A 3D mathematical system (1963) where microscopic changes create massive divergences. Its 3 parameters perfectly map to Pitch, Pan, and Volume for unpredictable but structured modulation.

Μέρος 3

Πρακτικά

Tips

Διαδικασία

- Διαμόρφωσε μια μουσική ιδέα που θες να πετύχεις και το εργαλείο που θα σε βοηθήσει.
- Περιέγραψε την ιδέα σε ένα AI Assistance περιβάλλον με κάθε λεπτομέρεια.
- Σε αυτό το αρχικό στάδιο η σωστή διαμόρφωση της ιδέας παίζει σημαντικό ρόλο.
- Κάνε copy - paste τον κώδικα σε ένα editor όπως το Visual Studio.
- Save as… HTML αρχείο.
- Ανοιξε το αρχείο στον browser.
- Παρατήρησε το αποτέλεσμα και επέστρεψε στο AI Assistance περιβάλλον για διορθώσεις.

Τι να προσέξεις 1

Μην αφήσεις το AI να κάνει τα πάντα

Αν το app σου είναι 100% AI-driven:

- δεν έχεις control
- δεν έχεις consistency
- δεν έχεις ταυτότητα

Κράτα:

- **AI = generator**
- **Εσύ = Director (rules, constraints, UI)**

Τι να προσέξεις 2

Αν ξεκινήσεις με **θολές ιδέες**, το AI θα σου δώσει **θολά αποτελέσματα**.

Ένα καλό input δεν είναι “θέλω ένα μουσικό app με AI”.

Αντίθετα, είναι κάτι που **περιορίζει, κατευθύνει και ορίζει εμπειρία**.

Τι να προσέξεις 3

Control layer (πολύ σημαντικό)

Το UI είναι άμεσα συνδεδεμένο με το audio engine.

Αυτή η σύνδεση χρειάζεται (τις περισσότερες περιπτώσεις) να είναι ξεκάθαρη.

Σκέψου το UI **πριν** το Audio Engine.

Ο χρήστης θα βλέπει αυτό.

Ανάλογα το app, μπορείς να ελέγξεις?

- density
- energy
- randomness
- structure

Τι να προσέξεις 4

Σε ποιους απευθύνεσαι;

Είναι εύκολο να παρασυρθείς με **προσωπικές συνήθειες** και διαδικασίες μουσικής παραγωγής.

Αν είναι ένα εργαλείο **μόνο για σενα,** δεν υπάρχει πρόβλημα.

Απευθύνεσαι σε άλλους users;

Τότε χρειάζεται να βγεις από τον εαυτό σου και να σκεφτείς σαν αυτούς.

Μπορεί να λειτουργήσει με τις ελάχιστες επεξηγήσεις?

Ο χρήστης δεν νοιάζεται ότι είναι AI generated.

Νοιάζεται:

- “τι νιώθω όταν το χρησιμοποιώ;”
- “έχει flow;”
- “με εμπνέει;”

Παράδειγμα καλού input

“Θέλω ένα mobile web app όπου ο χρήστης κινείται σε εξωτερικό χώρο και η απόσταση από ένα σημείο GPS καθορίζει το pitch σε ενα granular engine (το engine βασισμένο σε Asynchronous GS από τον Curtis Roads με όλες τις βασικές παραμέτρους για έλεγχο των clouds).

Η ταχύτητα της κίνησης να ελέγχει το grain density. Το grain size να βασίζεται σε Gaussian distribution. Το position στο buffer να ορίζεται από Brownian motion. Το granulation θέλω να γίνεται σε αρχεία ήχου που εισάγει ο χρήστης. Για window function στα grains χρησιμοποίησε Hann window με επιλογή και για Blackman.

Θέλω file input στο UI για .wav μέχρι 48Khz 24Bit.

Το UI θέλω να είναι οργανωμένο ανά section, να έχει σκοτεινά χρώματα και neon γραμματοσειρά και τα clouds να είναι animated ανάλογα την ταχύτητα και χρωματισμένα ανάλογα μπάντες συχνοτήτων (όσο πιο ψηλές οι συχνότητες, τόσο πιο φωτεινά clouds). θέλω επίσης την επιλογή ηχογράφησης και αναπαραγωγής μέχρι 5 λεπτα σε mp3.

Σκέψου για πιθανούς περιορισμούς στο audio engine, ώστε το app να τρέχει σε web χωρίς προβλήματα και memory issues. Το σύστημα πρέπει να είναι playable live και να έχει χαμηλό latency.

Τέλος θέλω ξεχωριστά αρχεία .html, .css, .js ”

Ιδέες:

Markov Melody MIDI Generator

Cellular Automaton Rhythm

Pentatonic MIDI with Stochastic Density Control

Fractal Pattern Mutation with Sine Waves

?

?

?