# Algorithmic Awakening

Workflow Documentation for the 2nd HKUST AI Film Festival

AI Music Section - New Boundaries, New Life



| | |
|---|---|
| Work Title | Algorithmic Awakening |
| Creator / Team | Shuaishuai |
| Category | AI Music Section |
| Duration / Format | 04:20 - WAV, 48 kHz, 16-bit, stereo |
| Documentation Type | Corrected single PDF supplementary material for the shortlisted work |

This document specifically describes the creation process of Algorithmic Awakening. It replaces earlier supplementary material that was prepared for another work.

# 1. Project Overview

Algorithmic Awakening is a computational music work that treats sound as an invisible film. The piece imagines an autonomous system moving from cold data processing toward a condition resembling consciousness. Its musical language begins with discrete, staccato pulses and gradually expands into denser drones, glitch textures, and mechanized rhythmic motion.

The central creative method is Computational Serialism: a custom twelve-tone sequence generation system is used as the structural source for pitch organization, while the final audio result is shaped through human arrangement, sound design, mixing, and mastering. The intention is not to hide behind automation, but to expose the creative boundary between algorithmic logic and human musical judgment.

## Short Synopsis

*The work explores the blurred boundary between carbon-based creativity and silicon-based logic. A system of isolated code-like pulses gradually learns to breathe: rigid serial patterns fold into organic drones, glitch fragments become rhythmic memory, and mathematical order begins to sound like an emerging life form. Through a hybrid workflow of custom algorithmic generation and human post-production, Algorithmic Awakening reflects on how AI can serve as a compositional partner rather than a replacement for human authorship.*

## Theme Alignment

- New Boundaries: the work crosses the boundary between composition, software design, algorithmic music, and cinematic sound design.
- New Life: the sonic narrative moves from binary regularity to emergent, organic musical behavior.
- Human-AI collaboration: computational tools generate structured material, while human decisions determine dramaturgy, texture, pacing, and final presentation.

# 2. Workflow Overview




The detailed workflow table appears on the next page to keep the process transparent and easy to read.

# Detailed Workflow Table

| Stage | Main Action | Human Decision | Output |
|---|---|---|---|
| 1. Concept | Define the sonic narrative: data, awakening, mechanical motion, and organic emergence. | Set the piece as an invisible sci-fi film rather than a conventional song. | Narrative and aesthetic direction. |
| 2. Algorithmic Source | Use my own code to generate twelve-tone rows, serial matrix transformations, and random sequence variants. | Set the constraints, inspect the generated material, and keep only fragments that serve the form. | Twelve-tone matrix, random sequence pools, pitch-class materials. |
| 3. Sonification | Map abstract numbers and generated sequences to pitch, rhythm, dynamics, arpeggiator behavior, and automation parameters. | Select registration, density, speed, loop length, and layering strategy. | MIDI/audio sketches and evolving texture blocks. |
| 4. Sound Design | Use Studio One arpeggiator, synthesizers, modulation, and processing to create glitch, pulse, drone, and cybernetic textures. | Program arpeggio behavior, shape synth timbres, adjust filters/envelopes, and build transitions manually. | Studio One project session with arranged audio/MIDI layers. |
| 5. Post-production | Mix and master the work for headphone/installation playback. | Control dynamics, harsh frequencies, and final loudness. | Final WAV master and cover image. |

# 3. AI and Computational Tools Disclosure

The project uses AI and computational processes at the levels of ideation, material generation, and documentation, while the musical form and final sound are manually curated. The following disclosure summarizes the tools and their roles.

| Tool / System | Role in the Project | Degree of Human Control |
|---|---|---|
| Custom code-based twelve-tone generator | Own code generates serial pitch-class matrices and transformation families such as P, R, I, and RI. | High: the creator defines the row logic, selects materials, and decides how to map them into sound. |
| Custom random sequence generator | Own code creates controlled random sequence pools for additional ordering, density, and textural variants. | High: stochastic output is filtered, selected, and edited by the creator rather than used directly as a finished composition. |
| Studio One arpeggiator | Turns selected pitch materials and generated sequences into pulsed, looped, and accelerating arpeggiated motion. | Full manual control: arpeggiator rate, gate, range, layering, and placement are programmed by the creator. |
| Studio One synthesizers and effects | Used to design synthetic timbres, glitch surfaces, drones, filter movement, modulation, spatial depth, and mix processing. | Full manual control: timbre, envelopes, effects, automation, arrangement, mixing, and export are shaped by the creator. |
| AI text assistant | Used for concept refinement, English wording, prompt framing, and workflow documentation drafting. | High: text is reviewed, edited, and structured by the creator. |

Note on originality: the submitted work is presented as an original human-AI / human-computational collaboration. The final selection of musical material, arrangement, sound design, mix, and master were made by the creator. No unauthorized third-party likeness or intentionally copyrighted sample material is included.

# 4. Core Algorithmic Design

The technical core is a custom code-based serial matrix system. Rather than generating tonal melodies through a general-purpose model, the project begins with a pitch-class row and applies strict operations from twelve-tone theory. This creates a rule-based musical grammar that is computationally transparent and musically constrained. In addition, a self-written random sequence generator was used to create candidate ordering, density, and textural variants. These random outputs were not treated as finished composition; they were filtered, selected, and re-shaped according to musical form, register, density, and dramatic pacing.

| Operation | Musical Meaning | Computational Logic |
|---|---|---|
| Transposition Tn | Move the entire row by n semitones. | $f(x) = (x + n) \bmod 12$ |
| Inversion I | Invert interval direction around a reference pitch. | $f(x_i) = (2x_0 - x_i) \bmod 12$ |
| Retrograde R | Reverse the order of the row. | $P = [x_{11}, x_{10}, \ldots, x_1, x_0]$ |
| Retrograde-Inversion RI | Apply inversion and then reverse the result. | $RI = R(I(P))$ |

# 5. Sonification and Musical Mapping

The matrix itself is only numerical material. To become music, the values are mapped into sound parameters. The mapping process is deliberately cinematic: pitch becomes character movement, rhythm becomes data flow, density becomes system load, and timbre becomes the surface of a machine learning to speak.

| Parameter | Mapping Strategy | Artistic Function |
|---|---|---|
| Pitch | Pitch-class values 0-11 are mapped to MIDI notes and registral layers. | Produces a non-tonal, high-tension harmonic field. |
| Rhythm / Arpeggiation | Generated sequences are distributed across different speeds, onset densities, and Studio One arpeggiator patterns. | Creates the feeling of pulses, code loops, and accelerating computation. |
| Dynamics | Layer density and automation shape crescendos, swells, and sudden interruptions. | Suggests the system moving between order and overload. |
| Timbre | Studio One synthesizers, filter/envelope editing, stepped or digital wave textures, glitch processing, and drone layers are used. | Balances mechanical coldness with an emerging organic surface. |
| Form | Sparse pulses gradually expand into larger textures, then stabilize in a final awakened state. | Turns an abstract process into a clear listening narrative. |

# 6. Key Prompt Excerpts

The following excerpts represent the prompt language used during AI-assisted ideation and sound-world planning. They were used as creative direction, not as a substitute for final musical editing.

| Prompt Excerpt | Purpose |
|---|---|
| "Create a cold, data-driven cinematic sound world where isolated code pulses slowly evolve into organic drones." | Defines the macro-narrative from rigid computation to emergence. |
| "Use glitch textures, minimalist rhythmic drive, and non-tonal serial pitch behavior; avoid conventional pop-song harmony." | Prevents the material from drifting into tonal or generic song language. |
| "Imagine an AI system waking up inside a neon digital highway: sharp attacks first, then breathing layers and widening space." | Connects the visual/film cue with musical pacing, texture, and spatialization. |
| "Keep the final sound suitable for headphone presentation: clear motion, controlled harshness, and no extreme dynamic spikes." | Supports exhibition safety and listener comfort. |

# 7. Major Human Edits

The final work was not accepted directly as raw algorithmic output. The generated materials were treated as source material, then selected, edited, layered, and produced manually.

| Area | Human Edit / Decision | Reason |
|---|---|---|
| Material Selection | Chose the most effective serial fragments and random-sequence variants, discarding patterns that sounded static or arbitrary. | Maintains musical direction and avoids pure randomness. |
| Registration | Reassigned pitch materials into low drones, mid-register pulses, and high glitch elements. | Creates depth and avoids a single flat register. |
| Rhythmic Editing | Rebalanced note density, spacing between pulse groups, and Studio One arpeggiator behavior. | Clarifies the sense of mechanical movement and acceleration. |
| Texture Layering | Combined twelve-tone fragments, random-sequence material, staccato pulses, arpeggiated motion, drones, and noise-like digital textures. | Builds a cinematic progression from data to life. |
| Sound Design | Designed synth timbres inside Studio One; adjusted arpeggiator settings, filters, envelopes, modulation, effects, and automation. | Gives the piece a recognizable machine-oriented sonic identity. |
| Mixing / Mastering | Controlled frequency masking, high-frequency harshness, and dynamic peaks. | Makes the piece suitable for public listening and headphone exhibition. |

# 8. Iteration Notes

| Version | Problem Found | Revision Made |
|---|---|---|
| Draft 1 - Code / Sequence Sketch | The twelve-tone and random sequence outputs were coherent but too rigid or sparse when heard directly. | Filtered generated material, added registral separation, sustained layers, and more varied onset density. |
| Draft 2 - Studio One Texture Build | The arpeggiated and synthetic layers became too dense, with harsh upper frequencies. | Adjusted arpeggiator density, reduced high-frequency accumulation, and clarified mid/low layers. |
| Draft 3 - Cinematic Form | The piece needed a stronger sense of awakening and final arrival. | Reshaped the buildup and extended drone expansion before the final stabilization. |
| Final Master | Needed exhibition-ready balance and clear technical transparency. | Rendered WAV master, checked duration/format, prepared corrected workflow documentation. |

# 9. Production Time and Deliverables

| Task | Estimated Time | Main Output |
|---|---|---|
| Concept and aesthetic planning | 2-3 hours | Narrative, theme, prompt direction |
| Custom code modeling and sequence generation | 5-7 hours | Twelve-tone matrix, random sequence pools, serial materials |
| Mapping and Studio One arrangement | 6-8 hours | Structured MIDI/audio sections, arpeggiated layers |
| Studio One sound design and editing | 5-6 hours | Glitch, pulse, drone, arpeggiator, and synthetic layers |
| Mixing, mastering, and export | 3-4 hours | Final 04:20 WAV master |
| Documentation and cover preparation | 2-3 hours | Workflow PDF and high-resolution cover image |

## Submitted / Prepared Files

- Final audio master: Algorithmic Awakening, WAV, 04:20, 48 kHz, 16-bit, stereo.
- Workflow documentation: corrected single PDF tailored specifically to Algorithmic Awakening.
- Cover image: high-resolution cybernetic waveform artwork used as the visual identity of the work.
- Process assets available upon request: Studio One session screenshots, stems, MIDI/sequence exports, custom-code screenshots/notes, and algorithmic generation records.

# 10. Rights and Authorship Statement

I confirm that I am the creator / authorized submitter of Algorithmic Awakening and have the right to submit the music, cover image, and accompanying documentation for festival review, screening, promotion, archive, and academic discussion under the festival rules. The submitted work is original and was developed through a hybrid process of custom code-based algorithmic composition, AI-assisted ideation/documentation, and human musical production in Studio One.

This supplementary document is prepared to provide transparent information about the creation process, including workflow overview, AI/computational tools, prompt excerpts, major human edits, iteration notes, and production outputs.

Creator / Team: Shuaishuai

Work Title: Algorithmic Awakening

Category: AI Music Section

# Figures and Code Appendix

Supplement to “Algorithmic Awakening – Workflow Documentation”

This appendix provides visual and technical supporting material for the shortlisted work **Algorithmic Awakening**. It is intended to accompany the workflow documentation and to make the computational and production process more transparent. The appendix includes:

- **Figure 1**: the self-developed twelve-tone / modernist sequence generator interface;
- **Figure 2**: the actual Studio One production workspace used in the project;
- **Code Excerpts 1–3**: representative source-code fragments showing sequence generation, voice-leading logic, and style-parameter definitions.

| | |
|---|---|
| Project | Algorithmic Awakening |
| Category | HKUST AI Film Festival · AI Music Section |
| Appendix purpose | Visual evidence of tools, interface design, and representative code logic |
| Production environment | Custom Python code + Studio One |
| Included materials | Generator interface, actual DAW workspace, and code excerpts |

Note: the production interface shown in Figure 2 uses the latest actual Studio One project screenshot provided by the creator.

# Figure 1 · Self-Developed Twelve-Tone Generator Interface

The interface presents the twelve-tone matrix, tonal-bias options, style presets, parameter controls, instrument-library browsing, and matrix-analysis areas used to support material generation.




Figure 1. Custom software interface for serial / modernist material generation.

# Figure 2 · Actual Studio One Project Workspace

This figure shows the actual Studio One project workspace used during production. It includes the arrangement timeline, color-coded track organization, mixer section, plug-in routing, and processing environment used for editing, sound design, and final production.

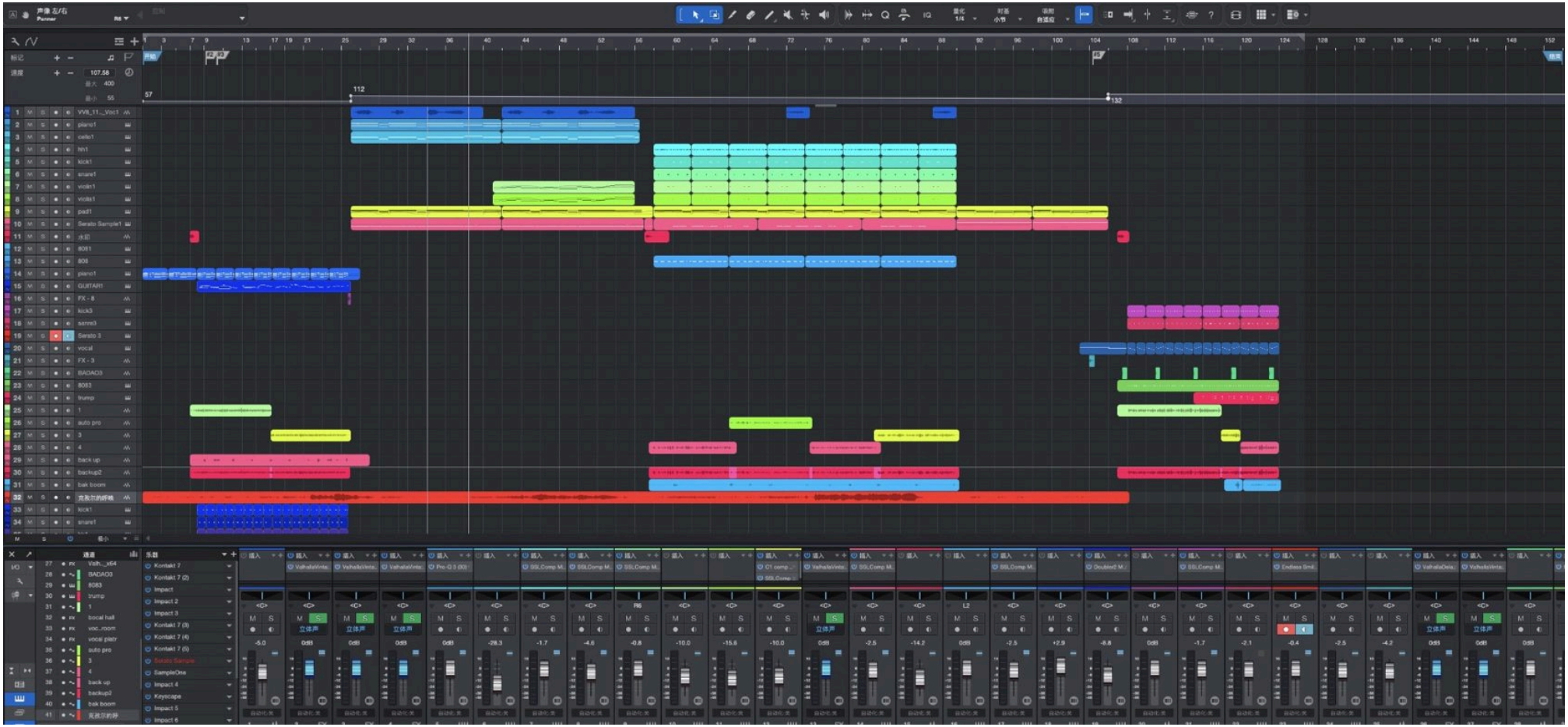


Figure 2. Actual Studio One production session used for arrangement, mixing, and sound design.

## Code Excerpt 1 · Random Twelve-Tone Row Generation

This function generates twelve-tone rows with different bias models (atonal, pentatonic, diatonic, whole-tone, or octatonic), while preserving a computationally transparent rule-based structure.

```
def generate_random_row(self, tonality='atonal'):
    """Generate a random twelve-tone row with tonal bias."""
    self.reset_seed()

    if tonality == 'atonal':
        row = random.sample(range(12), k=12)

    elif tonality == 'pentatonic':
        pentatonic = [0, 2, 4, 7, 9]
        remaining = [1, 3, 5, 6, 8, 10, 11]
        row = self._blend_sources(pentatonic, remaining, bias=0.75)

    elif tonality == 'diatonic':
        diatonic = [0, 2, 4, 5, 7, 9, 11]
        remaining = [1, 3, 6, 8, 10]
        row = self._blend_sources(diatonic, remaining, bias=0.70)

    elif tonality == 'whole_tone':
        whole_tone1 = [0, 2, 4, 6, 8, 10]
        whole_tone2 = [1, 3, 5, 7, 9, 11]
        base, extra = (whole_tone1, whole_tone2) if random.random() < 0.5 else (whole_tone2, whole_tone1)
        row = self._blend_sources(base, extra, bias=0.60)

    elif tonality == 'octatonic':
        octatonic = [0, 1, 3, 4, 6, 7, 9, 10]
        remaining = [2, 5, 8, 11]
        row = self._blend_sources(octatonic, remaining, bias=0.65)

    else:
        row = random.sample(range(12), k=12)

    return row
```

Interpretation: instead of using unconstrained automatic melody generation, the system organizes pitch material through explicit collections and weighted blending, keeping the process reviewable and reproducible.

## Code Excerpts 2–3 · Voice Leading and Style Parameter System

The following excerpts demonstrate basic voice-leading generation and the style-parameter dictionary used to configure compositional behavior.

### Code Excerpt 2 · create_voice_leading()

```
def create_voice_leading(self, chords, num_voices=4):
    """Create voice leading from a chord progression."""
    voices = [[] for _ in range(num_voices)]

    for chord in chords:
        root = chord['root']
        chord_type = chord['type']

        if chord_type == 'major':
            chord_notes = [root, (root + 4) % 12, (root + 7) % 12]
        elif chord_type == 'minor':
            chord_notes = [root, (root + 3) % 12, (root + 7) % 12]
        elif chord_type == 'maj7':
            chord_notes = [root, (root + 4) % 12, (root + 7) % 12, (root + 11) % 12]
        elif chord_type == 'min7':
            chord_notes = [root, (root + 3) % 12, (root + 7) % 12, (root + 10) % 12]
        elif chord_type == 'dom7':
            chord_notes = [root, (root + 4) % 12, (root + 7) % 12, (root + 10) % 12]
        else:
            chord_notes = [root, (root + 4) % 12, (root + 7) % 12]

        for i in range(num_voices):
            note = chord_notes[i % len(chord_notes)] + 12 * (3 + i)
            voices[i].append(note)

    return voices
```

### Code Excerpt 3 · Style Parameter Dictionary

```
self.styles = {
    'serial': {
        'name': 'Serialism',
        'description': 'Strict twelve-tone writing',
        'parameters': {
            'strict_serialism': True,
            'allow_repetition': False,
            'development_techniques': ['transposition', 'inver
            'texture': 'contrapuntal'
        }
    },
    'spectral': {
        'name': 'Spectralism',
        'description': 'Spectral-based composition',
        'parameters': {
            'spectral_based': True,
            'microtonal': False,
            'texture': 'clustered',
            'dynamic_evolution': True
        }
    },
    'minimalist': {
        'name': 'Minimalism',
        'description': 'Repetition and gradual process',
        'parameters': {
            'repetition': True,
            'gradual_change': True,
            'pattern_based': True,
            'texture': 'ostinato'
        }
    }
}
```

These excerpts are representative fragments included to clarify the computational logic behind the work. The full project codebase remains modular and larger than the sample shown here.

# Software Copyright Registration Evidence

This appendix adds the software copyright registration certificate for the custom twelve-tone generation tool used in *Algorithmic Awakening*. The original certificate is shown below as issued; the official source language is Chinese.

| | |
|---|---|
| Software name | Twelve-Tone Cube Generator Software |
| Abbreviation | Twelve-Tone Cube |
| Version | V1.0 |
| Copyright owner | Dai Xudong |
| Method of rights acquisition | Original acquisition |
| Scope of rights | All rights |
| Registration number | 2026SR0486941 |
| Certificate number | Software Copyright Registration No. 17701222 |
| Registration date | March 25, 2026 |
| Issuing authority shown on certificate | National Copyright Administration of the People’s Republic of China / China Copyright Protection Center |



中华人民共和国国家版权局
计算机软件著作权登记证书

证书号： 软著登字第17701222号

软 件 名 称： 十二音魔方生成软件
[简称：十二音魔方]
V1.0

著 作 权 人： 戴旭东

权利取得方式： 原始取得

权 利 范 围： 全部权利

登 记 号： 2026SR0486941

根据《计算机软件保护条例》和《计算机软件著作权登记办法》的规定，经中国版权保护中心审核，对以上事项予以登记。

中华人民共和国国家版权局
计算机软件著作权登记专用章

2026年03月25日

Original certificate image included as supporting evidence. The English table above is a concise translation of the key registration fields.