# Fibonacci Jungle

A generative framework for Jungle and Drum & Bass based on the Fibonacci number sequence

Martin Heinze
Berlin, Germany
papers@martsman.de

## ABSTRACT

While singular generative techniques have already become an established part of the creative process in music writing, holistic approaches to generative music production in traditional electronic dance music genres yet seem under-represented both in theory and practice. *Fibonacci Jungle* provides a simple to use generative framework for Jungle and Drum & Bass built on the Fibonacci number sequence as structural alternative to conventional meters and track build-up. This framework is implemented in Pure Data; it uses probability and randomization within a pre defined set of genre typical parameter settings (tempo, harmonics, sample selection). *Fibonacci Jungle* allows creating stand alone tracks in a Jungle and Drum & Bass aesthetics with only a few clicks and can be individually customized.

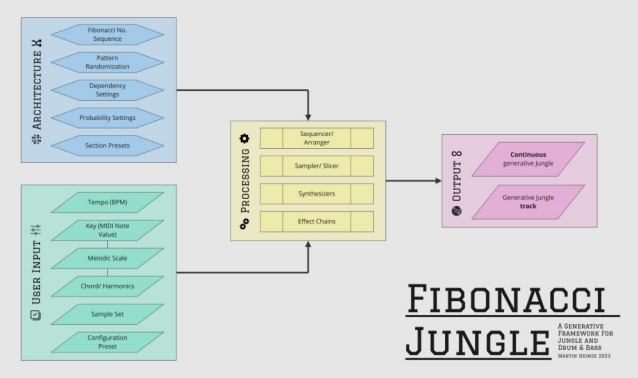


Figure 1: **A schematic depiction of the *Fibonacci Jungle* conceptual framework**

## 1 Introduction

### 1.1. Background

Jungle music, a pivotal sub-genre of electronic dance music, emerged in the early 1990s in the United Kingdom. It represents a significant cultural and sonic evolution in the history of electronic music. The genre originated as a direct descendant of the UK's rave and acid house scenes of the late 1980s. As these scenes matured, they gave birth to various sub-genres, with Jungle music being one of the most distinctive. It evolved primarily in London, but its influence quickly spread across the UK and worldwide.

The genre is characterized by several key elements, including:

- **Breakbeats**: Jungle's defining feature is its use of intricate and fast-paced breakbeats, typically sampled from classic funk and soul tracks. These complex rhythms are often syncopated and layered.
- **Sub Basslines**: Jungle music features deep, rumbling basslines that provide a powerful and driving force, contributing to its distinct sound.
- **Reggae, Ragga and Dancehall influences**: Jungle often incorporates vocal samples and elements from reggae, dancehall, and hip-hop, adding an urban flavor to the music.
- **Samples and Sound Collages**: Jungle is known for its extensive use of samples, creating a collage of sounds, including vocal snippets, movie dialogues, and environmental noises. This eclectic approach to sampling contributes to its unpredictable and immersive sonic landscape.
- **High Tempo**: Jungle typically maintains a high tempo, ranging from 150 to 180 beats per minute, adding to the general intense, energetic atmosphere of the music.

Like in other dance music genres, Jungle tracks are built on a steady 4/4 meter and typically contain a set of recognizable events or sections like breakbeat-only intro, interlude or outro, other, sonically reduced to almost silent parts where suspense is being created and drops with high intensity that serve as points of release for the built suspense.

The sequence of these sections adheres to a structured build-up format, often tailored for playback during DJ sets where two consecutive tracks are seamlessly mixed into each other. While this overall track structure establishes a familiarity and consistency for the listener, it also limits the occurrence of unexpected moments and unconventional developments within the musical piece.

## 1.2. Motivation

In order to provide for a wider range of unexpected turns and occurrences within a known and sound wise familiar musical genre, the intention of the author is to introduce an alternative pattern to the structural composition of Jungle music while maintaining the genre typical sound aesthetics and characteristics. This is being implemented by employing the Fibonacci number sequence as structuring principle.

The Fibonacci number sequence is a well-known mathematical pattern, consisting of a series of numbers where each number is the sum of the two preceding ones[1]. This sequence is named after the Italian mathematician Leonardo of Pisa, or Fibonacci, who introduced it to the Western world in his 1202 book, "Liber Abaci." The sequence starts as follows: (0,) 1, 1, 2, 3, 5, 8, 13, 21, 34 and so on. As one progresses along the sequence, the ratio between consecutive Fibonacci numbers converges to the golden ratio, approximately 1.618033988749895. This ratio is noted for its presence in various natural phenomena and its influence in art, architecture, and aesthetics.

While employing the Fibonacci number sequence as a foundational principle in music is certainly not unprecedented [1], its application within the structural context of electronic dance music remains a relatively untapped concept. To facilitate the application of this concept for music producers in the Jungle and Drum & Bass music domain, it is being implemented as ready-to-use and generative setup *Fibonacci Jungle*.

## 2. Implementation

*Fibonacci Jungle* emerges as a framework building upon the Fibonacci number sequence and reproducing distinct sound characteristics of Jungle music from the 1990s in form of a Pure Data patch including various abstractions for breakbeat slicing and playback, sampling, synthesizing techniques, note sequence (melody) generation as well as rhythmic controls. The patch setup is supplemented with sets of three breakbeats and six additional (e.g. vocals, pads, background/ atmospheric) samples.

With both a pre definable parameter configuration (e.g tempo, melodic scale, sample set) and layers of randomized parameters and probabilistically altered events triggered when continuously cycling through a limited range of the Fibonacci number sequence, *Fibonacci Jungle* generates stylistically typical, structurally unconventional Jungle or Drum & Bass tracks that potentially never turn out the same recording artefact.

### 2.1. Fibonacci number sequence as structuring principle

Fundamentally, *Fibonacci Jungle* restructures the conventional rhythmic 4/4 metering framework prevalent in electronic music. It leverages the Fibonacci number sequence to govern the duration of each musical bar, anchoring the time signatures within fractions of full note values – in the case of the patch e.g. 1/8, 2/8, 3/8, 5/8, 8/8, and beyond. This approach introduces a distinctive rhythmic framework, which, especially when combined with the characteristic syncopation of breakbeats in Jungle, along with the slicing and rearranging of these beats, leads to heightened unpredictability and added complexity.

*Fibonacci Jungle* uses the number sequence to not only introduce changing bar duration but also provide an alternative general structure for the whole musical arrangement where the sequence also defines the length of various sections within the piece and orchestrates the probabilistically altered convergence of musical elements such as breakbeats, basslines, other melodic and rhythmic elements or sample playback. Through this, *Fibonacci Jungle* devises an unconventional track arrangement distinct from the conventional build-up structures prevalent in Jungle and other electronic dance music genres.

*2.1.1 Fibonacci sequencer and arranger.* In the ***pd fibonacci-engine*** sub patch, both bar duration and general track structure are controlled via dedicated sequencers using a BPM based metro object.

Bar duration is calculated in ***pd fibonacciseq*** using 8th note values as provided by the metro object and Fibonacci sequence values up to 34. When reaching 34, the sequence is being reset and starts anew[2].

The track structure is built upon completed bar duration cycles and uses Fibonacci values up to 8[3] to map out different sections of the track in ***pd fibonacciarranger***. When reaching 8, a sections cycle is being reset and starts anew. In parallel, a second cycle is used to define a total track length and automatically stops playback if continuous mode is disabled.

Fibonacci values in any of the cycles trigger events in other sections of the patch, which can lead to e.g. the restart of a breakbeat pattern playback or tonal sequence or the activation of a track section specific configuration e.g. toggling on/off elements. The occurrence of these events are influenced by probability settings, dependencies (e.g. mutual exclusivity of audio generators) or section specific presets.

### 2.2. A generative framework for Jungle

In addition to the structural framework described above, *Fibonacci Jungle* exposes an initial layer of parameters that are being set on loading of the patch and can be altered via optional user input (main patch frontend). These parameters include a sample set selection, the track tempo in BPM, a key MIDI note value, a melodic scale and a harmonics selection in the form of a chord. On initial loading, parameters are being set via randomization within a defined range of genre typical settings

[1] https://oeis.org/A000045

[2] Ending the sequence cycle at 34 here is a conceptual decision committing to a compact presentation of *Fibonacci Jungle*.

[3] See above. The higher the value, the longer a certain segment configuration remains active, generally prolonging the track duration.

(e.g. BPM values between 140-190, covering early UK Hardcore to Jungle to Breakcore). This randomization can also be triggered manually in the main patch.

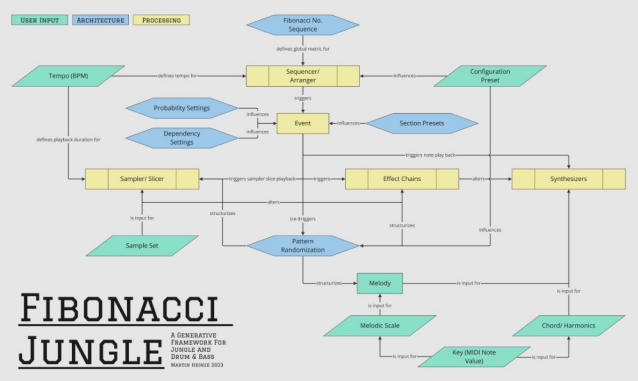


Figure 2: **A simplified depiction of *Fibonacci Jungle* components, assets and their interdependency**

A second layer of parameters is being set and altered without user input based on initial or repeating randomization. These include the playback order of breakbeat slices, tonal sequences of bassline and other soft synthesizers or play back patterns for other rhythmic elements (blips, high hats). The processing of these parameters to audio information is done within the ***pd jungle-engine*** sub patch

*2.2.1. Sample sets. Fibonacci Jungle* uses pre defined sets of three breakbeats and six additional samples. The patch is loading these from ./samples/set$ subfolders, where $ is a value from 0 to 3. Files placed in these folders need to be named break1.wav, break2.wav, break3.wav, sample1.wav, sample2.wav (...), samplex.wav in order for the patch to load them properly. All samples need to be in WAV or AIFF format and 44.1 KHz. The patch is configured to work with breakbeats of two full 4/4 bars in length, ideally with a tempo in the same range as the given global tempo[4]. The sample processing and playback is done inside the ***pd beats*** and ***pd samples*** sub patches.

*2.2.1.1 Sampler and breakbeat slicer.* The breakbeat samples are being loaded via the ***slicer.pd*** abstraction, a sampler engine primarily useful for rhythmically repetitive source material. The original tempo of the sample is calculated based on the assumption that it's a 4/4 equivalent meter (and either one, two or four bars in length). The play back tempo can be adjusted to the global tempo of the main patch. The file is split into max. 16 equal slices that can be played back in either a totally random order or as a randomly generated but fixed and repeatable pattern.

[4] Breakbeats are pitch-shifted to match the main patches' tempo in BPM but generally maintain their original sound characteristics. Technically, providing one bar and four bars of breakbeat sample material are possible to use as well, the detection would then need another user input, which is not described here.

For sample playback, the ***simpler.pd*** abstraction is being used. The abstraction allows defining play back start and end points if necessary.

*2.2.2. Other rhythmic elements.* Two components contribute to additional variety in the rhythmic section of ***pd beats***, one of which is a high hat synthesizer controlled via the random play back pattern sequencer abstraction ***herrmann.pd***, the other one being a simple blip FM synthesizer. Both work as placeholders for potentially more sophisticated additions to the patch infrastructure.

*2.2.3. Synthesizers. Fibonacci Jungle* incorporates three main tonal sound generators, one being a simple bassline synthesizer that generates sine wave based sub bass with additional LFO for Jungle typical wobble sounds, a second layer additive stab synth that can play along with the bassline, and another additive synthesizer for a typical dub chord or stab sound as derived from Reggae or Dub influences. These tonal components can be found inside the ***pd tonals*** sub patch.

*2.2.3.1 Melodic sequencing: Bassline/ stab line.* To generate a tonal sequence for these synthesizers, two components work in conjunction. The first component is the ***scaleculator.pd*** abstraction. Scaleculator generates one octave of MIDI note values based on the key note provided from the main patch and a selection of traditional western melodic scales e.g. Dorian, Phrygian, Aeolian, Mixolydian. It outputs these note values along with an index for further processing.

The second component is ***frauke.pd***. This sequencer generates a random pattern of numbers from the index value received from Scaleculator, e.g. if the index value is 9, Frauke generates a sequence of 9 values between 0-8. The complexity or variety of the sequence can be pre-defined, e.g. selecting the *Full Scale* option will use all values – convenient for melody lines – whereas the *Primitive* option will use only a fraction of values, potentially more suitable for basslines.

Upon reaching a Fibonacci number in the bar duration cycle, the subsequent value in the sequence is invoked, which refers to a specific MIDI note provided through Scaleculator. This note is then sent to the two synthesizer (***subbass.pd*** and ***minitoner.pd*** abstractions) and transformed into an audio signal.

*2.2.3.2 Harmonics generation: Dub Chord.* Where Scaleculator generates an array of note values in a melodic scale, ***chordculator.pd*** abstraction similarly creates a list of note values based on the key note value input from the main patch and a selection of commonly used harmonic chords e.g. Major, Minor, Diminished, Augmented and its variations. An initial chord is being defined in the setup section of the patch. The output list of note values from the selected chord is distributed to a set of ***dstab_mini.pd*** abstractions – compact additive synthesizers – and processed to audio.

*2.2.4. Effect chains.* Besides sound generation, *Fibonacci Jungle* incorporates a range of standard effect processing chains such as reverb, delay, filters and limiters. In favor of a more concise project description, these will not be elaborated on here.

## 2.3. Examples

The framework has been put into action creating two releases by the author, the *Fibonacci Jungle Versions*[5] and *Edouard & Leonardo*[6], the patch setup to the latter having been showcased on YouTube[7]. Both releases have been published using multiple variants of the respective track setups for different publication channels, underscoring the generative character of the recordings' source.

# 3. Conclusions and potential future developments

*Fibonacci Jungle* aims at enabling producers in the Jungle and Drum & Bass music genre to generate tracks within a framework that combines aesthetic familiarity with a substantially different approach in terms of track segmentation and build-up, thus both simplifying the process of creative music writing but also challenging established production routines by presenting alternative structure formattings off the beaten path.

Depending on the level of acceptance within the target audience, *Fibonacci Jungle* could potentially transform into a general algorithmic Jungle framework with additional sequencing schematics based on other integer sequences and also a conventional track build-up structure[8]. To cater for more sophisticated output in sound design, *Fibonacci Jungle* could serve as a fundamental structural framework compatible with a conventional DAW via MIDI or OSC to control industry standard synthesizers and sound generators, which is another option to explore.

# ACKNOWLEDGMENTS

The author would like to thank Jason Hockman for general advise and thought exchange.

# REFERENCES

[1] Shannon, A. & Klamka, Irina & van Gend, Robert. (2018). *Generalized Fibonacci Numbers and Music.* Journal of advances in mathematics. 14. 7564-7579. 10.24297/jam.v14i1.7323.

[2] Anderson, C., Eigenfeldt, A., & Pasquier, P. (2021). *The Generative Electronic Dance Music Algorithmic System (GEDMAS).* Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 9(5), 5-8. https://doi.org/10.1609/aiide.v9i5.12649

[3] Hockman, J.A. & M.E.P. Davies (2015). *Computational strategies for breakbeat classification and resequencing in hardcore, jungle and drum & bass.* 18th International Conference on Digital Audio Effects, Trondheim, Norway.

[4] Bakım, Sümeyye & Yöre, Seyit. (2020). *Investigation of applications of Fibonacci Sequence and Golden Ratio in music.* 10.35379/cusosbil.625899.

[5] https://www.martsman.de/2023/07/07/new-release-fibonacci-jungle-versions/ (accessed Nov. 4th, 2023)

[6] https://www.martsman.de/2023/08/08/more-algo-jungle-output-edouard-leonardo-bandcamp/ (accessed Nov. 4th, 2023)

[7] https://youtu.be/OwkacnSUn10?si=aHZH68emlh_oYons (accessed Nov. 4th, 2023)

[8] This concept has been exemplified in another blog post by the author: https://www.martsman.de/2023/02/08/building-a-jungle-track-in-pure-data-while-still-leaving-things-to-chance/ (accessed Nov. 4th, 2023)