Algorithmic Drum Machine with Light Dependent Timbre Control

Nikhil Bullock
n.d.bullock@se25.qmul.ac.uk
Centre for Digital Music
Queen Mary University of London
London, United Kingdom

Charalampos Saitis
c.saitis@qmul.ac.uk
Centre for Digital Music
Queen Mary University of London
London, United Kingdom

Anna Xambó Sedó
a.xambosedo@qmul.ac.uk
Centre for Digital Music
Queen Mary University of London
London, United Kingdom

Figure 1: The instrument in use during a performance (photograph by Shuoyang Zheng).

Abstract

Both the digital audio workstation (DAW) timeline and the traditional drum machine interface encourage the production of music structured around repetitive, fixed-length rhythmic patterns. To our knowledge, the production of complex, irregular rhythms is mostly out of reach when drumming virtuosity is unavailable. We present a hardware DIY drum machine instrument that uses algorithmic sequencing that can generate evolving, unpredictable rhythms that are still rooted within dance music and other rhythm-based aesthetics. The drum machine also contains light sensors that facilitate real-time timbre control, with the aim of creating an instrument that offers an engaging and enjoyable interface that helps, from the perspective of the performer, make the drum machine feel more like an instrument. We reflect on what we have learnt through building this prototype, particularly the trade-off between algorithmic complexity and immediate control and how timbre compares with direct control of rhythm as the primary real-time interaction in a percussion instrument.

Keywords

Algorithmic music, Drum machine, Timbre, DIY

1 Introduction and Related Work

Conventionally, drum machines are controlled with a step sequencer interface, the Roland TR-909 is one iconic example, the Korg Volca Beats is a more recent example. In this design, each drum sound plays on its own track, related to the other tracks only by the master tempo. The performer has to cycle through each drum sound and edit the pattern individually, making it quite challenging to make significant changes to the overall output in real-time. This interface also affords, as does the DAW timeline, thinking about temporal structure linearly, and in terms of fixed-length repeated blocks [11, 19].

The aim of this project is to build a drum machine that facilitates the creation of varying, non-linear rhythmic patterns using a minimal set of controls in order to make playing the instrument accessible and fun for people with limited or no musical experience. Attempts were made to achieve this by creating an interface that, using algorithmic sequencing techniques, combined pattern generation and instrument selection into a single set of hardware parameters, such that each parameter change would effect the rhythm as a whole.

As rhythm is the primary focus of a drum machine, timbral controls are often limited or secondary. We explored interfaces for real-time timbre control of the drum sounds, and the current prototype contains light sensors triggered using a handheld light. While light and touchless control more generally have been used extensively in instruments in NIME and elsewhere, [1, 10, 16] they have been less commonly applied to drum machines.

It was intended that the rhythmic patterns and sounds produced by the drum machine should be inspired by, and fit within the context of, the music of the subcultures within the Hardcore Continuum [17]. This was partially down to aesthetic preferences but also to foster creativity by working within a set of self-imposed constraints [6]. For example, using polyrhythms to build rhythmic complexity would result in patterns that would deviate too far from those found in these styles of music, which invariably use a 4/4 time-signature and rely on syncopation to create rhythmic complexity. While the use of polyrhythms has been an effective method for expanding the rhythmic possibilities of a drum machine [5, 7], the present instrument does not rely on

NIME '26, June 23–26, 2026, London, UK

polyrhythms to generate rhythmic complexity, but rather on the interaction between parameters and the perception of rhythm through timbral modulation. To our knowledge, this approach to the structure of rhythm is somewhat less well-explored within the context of a hardware drum machine interface.

Many drum sequencers have featured in NIME and elsewhere that deviate from the traditional drum machine paradigm, either by modifying the step sequencer interface or by using a different interface entirely. A common approach to enhancing the capabilities of the step sequencer interface has been to integrate machine learning or other probabilistic methods. Existing approaches include using a machine learning model to generate patterns on a step sequencer using data from another interface, or a model trained on existing music [21–23]. Others have used a model to interpolate between existing step sequencer patterns or modify sequences created by the user [4, 24]. In other cases the step sequencer interface has been modified to enhance accessibility [12].

While the term 'algorithmic' is often used to refer to systems that employ stochastic techniques from probability theory and AI [15], algorithmic systems of interest in this paper include those of the type that generate *algorithmic patterns*, described by Alex McLean as those where "complexity results from simple parts" [15, p.2]. The avoidance of machine learning and probabilistic techniques was based on a desire for the instrument to run on low power hardware and to reduce the conceptual distance between the performer and their generated sequences. Algorithmic sequencing of this kind includes Euclidean rhythms [20], and programming language libraries for creating and manipulating patterns through transformations such as TidalCycles [14] or the SuperCollider Pattern classes [8].

2 Implementation

2.1 Construction

The instrument houses five potentiometers and three light dependent resisters (LDR) attached to an Arduino Uno, sandwiched between two pieces of wood (as shown in Figure 2). The two pieces of wood are connected using screws and PCB standoffs. The panels and the mounting holes were cut using a laser cutter. An earlier prototype consisted of a fully enclosed box held together with glue. This design was abandoned partly due to the way it looked, but also because it would have made repairing the device (if necessary) difficult. Keeping the electronics partially exposed and using screws and PCB standoffs makes the instrument easy to repair and modify.[1]

2.2 Software

Data transfer. The pattern generation and sound synthesis is handled within SuperCollider [13]. The data from the sensors is passed to SuperCollider through an Arduino. The data from the potentiometers are used as parameters to the pattern generation algorithm, and the light sensor data are used as parameters to the synth patches).[2]

Sound synthesis. All the sounds were generated in SuperCollider using frequency modulation (FM) synthesis. FM synthesis was chosen for a number of reasons:

- It is capable of generating a wide range of timbres.
- It is suitable for synthesising (metallic) percussive sounds.
- It is computationally efficient, meaning its use aligned with the goal of creating an instrument that did not require extensive computational resources.
- It has a long history in pop and dance music.

[1]A video demonstrating the use of the instrument has been uploaded to https://nb-nik.org/posts/nime2026-drum-machine.html.
[2]All source code is available at https://codeberg.org/nb-nik/fmplex_03

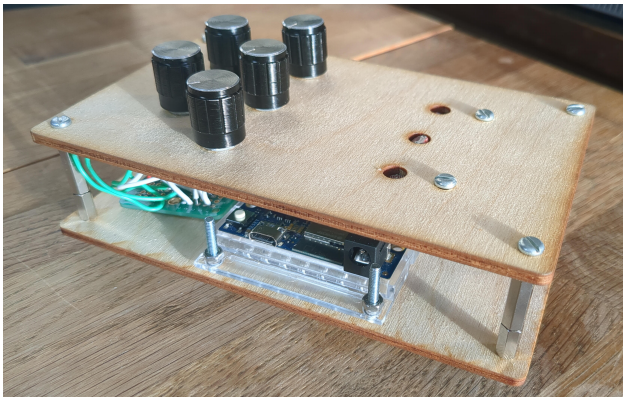

Figure 2: The instrument contains three LDRs for timbre control and five potentiometers used to control the parameters of the sequencer.

The patches were inspired by those presented by Erik Larsson [9], but many of the sounds differed in topology and/or tuning. For example, the kick drum patch (shown in Figure 3) includes an extra oscillator to enhance low frequency content. The data from the LDRs are mapped to modulator amplitude and length. The Lag unit generator, which low-passes an incoming signal to smooth out discontinuities, was necessary to soften the sudden changes in the sensor data and make the timbral modulation more musical.

```
SynthDef(\kick, {
  var freq = \freq.ir(440) * Env([4, 1], [0.5], -20).ar;
  var fm = Lag.kr(In.kr(\fm.ir)).linlin(0, 1, 0.2, 8);
  var sub = SinOsc.ar(freq) * 0.5;
  var len = Lag.kr(In.kr(\len.ir)).linlin(0, 1, 0.2, 0.8);
  var mod = SinOsc.ar(freq * 2.10) * fm;
  var sig = SinOsc.ar(freq, mod);
  sig = sub + sig * Env.perc(0.0001, len, 1, -4).ar(2);
  sig = Pan2.ar(sig) * \amp.ir(0.5);
  OffsetOut.ar(\out.ir(0), sig);
}).add;
```

Figure 3: The kick drum synthesis patch was implemented in SuperCollider 3 using two operator FM synthesis. The LDRs are mapped to modulator amplitude (lines 3 and 6) and envelope length (lines 5 and 8).

Effects. Two effects were added: a dub-style echo and a granular based delay/reverb effect. These effects were controlled with a MIDI Controller during the performance (discussed in Section 4), where they were used to help build more of a narrative arc from the raw sounds.

Algorithmic sequencing. In earlier prototypes, we tried using stochastic methods (such as Markov chains) to sequence the patterns. Through experimenting with the designs, it was found that algorithms that relied too heavily on randomness produced patterns that were too chaotic and created a sense of detachment between control and output from the resulting sounds when playing the

instrument. The current algorithm is largely inspired by the Pattern Synthesis algorithms introduced by Mark Fell [2], though it differs in many ways from any of the algorithms presented in his text.

In the algorithm, shown in Figure 4, each instrument is assigned a number from 0 to 4. The data from the potentiometers is held in an array called `pots` with values ranging from 0 to 1.

(1) For each potentiometer index `i`, take the pairwise absolute difference of the potentiometer values from `i` to the end of the array (these values will always be between 0 and 1 as the potentiometer values range from 0 to 1).
(2) For each of the values `x` calculated in step 1,
   (a) select an instrument by mapping it to the range 0 to 100 modulo the number of instruments.
   (b) Map the value to the range 1 to 32 and rest for that number of 16th notes.

```
loop {
  pots.size.do { |i|
    pots.drop(i).differentiate.collect({|x| x.abs}).do {
      |x|
      s.bind {
        switch (instSpec.map(x).mod(5).asInteger)
        { 0 } { /* hihat */ }
        { 1 } { /* rimshot */ }
        { 2 } { /* perc */ }
        { 3 } { /* cowbell */ }
        { 4 } { /* kick */ };
      };
      (restSpec.map(x) * 0.25).wait;
    };
  };
};
```

Figure 4: The pattern sequencing algorithm, implemented in SuperCollider 3 (the implementation of the instruments have been removed from this figure for clarity). The algorithm is described in detail above.

The specifics of the algorithm (e.g. the ranges to map the potentiometers to) were chosen based on the performer's analysis of the produced patterns. While simple, it was felt that the algorithm was able to produce a range of complex and unpredictable patterns while being entirely deterministic. As the patterns do not loop at a fixed length, twisting one knob causes the length of the loop as a whole to change, which makes predicting the rhythmic patterns more difficult and gives the rhythms a highly syncopated feel. The relationship between the knob positions also determines which drum sound should be played, further increasing, from the performer's perspective, the unpredictability of the patterns.

The algorithm was written through a process described by John Chowning as one of "entering, waiting, evaluating, adjusting" [3, p. 15]. This process, forced by the constraints of working with hardware, deviated significantly from the instrument designer's usual practices, where working in a software only environment makes it too easy to constantly abandon existing projects and start again. This much slower iterative cadence meant that by the time of the performance, the designer had internalised the particularities of the algorithm and how to draw the most out of the timbral controls.

3 Composition and Performance

The building of the prototypes culminated in a collaborative performance. Given that a drum machine has historically been designed to be an accompanying instrument, it was felt that a collaborative performance was a better context in which to analyse the prototype. The other performer's instrument also informed the sound design of the drum machine. The other performer was playing their digital instrument that consisted of a neural network that was trained on various recorded sounds, which generated dense ambient textures. The two instruments complemented each other well. The textures filled the space that was missing from the drums and the drums provided the textures with enough structure and definition to create a coherent and engaging performance.

This was the first time the performers played together. The performers have since continued to collaborate, recording a piece of music together featuring both instruments. However, we feel that the collaborative aspect of the performance could have been enhanced by linking the instruments over the network. In further iterations, the instruments could communicate over the network so data changes in one instrument could be used to trigger parameter changes in the other.

4 Reflections

Creating an interface that made sequencing the patterns engaging without demanding constant control was challenging. In an earlier prototype, the patterns were generated by combining binary numbers, encoded using switches, with various Boolean operations. While the generated patterns were decent, the physical interaction with the instrument left a lot to be desired.

It was originally intended that the instrument would be played by shining a lamp over it and blocking the light sensors with pieces of paper or hands. It was found through playing with the instrument that using a bicycle light to control the sensors resulted in more interesting sounds and was much more fun to play. Some audience members also remarked that it was enjoyable to watch and that it made the visual aspect of the performance align with what they were hearing, something that is often missing in electronic music performances [18].

The most interesting timbres were produced by waving the bicycle light over the instrument just after a sound was triggered, causing a distinct undulation of the brightness of the decay portion of the sound. The difficulty with anticipating the rhythmic patterns made playing with the bicycle light and triggering these more interesting modulations challenging yet engaging.

The instrument was able to produce a wide range of patterns, some of which were reminiscent of those found in UK Funky and Garage. It had some aspects of the immediacy felt when playing a traditional instrument. However, it was not always easy to conceptually map parameter changes to the resulting pattern, and it was necessary to use external effects and hardware to create a sense of narrative in the performance.

The effects were operated using a separate MIDI controller. Controlling the patterns, light sensors, and effects simultaneously was quite difficult (and at times frustrating). Future iterations could explore how to operate all these elements in a more unified interface.

5 Conclusion

This paper presented an exploration into some of the ways in which a DIY drum machine interface can be enhanced with algorithmic composition techniques and gestural control using light. Algorithmic composition was used to increase the range and complexity of the possible producible rhythms. Gestural control was incorporated in an attempt to capture some of the feeling of playing a traditional instrument. This paper also reflected on some of the challenges faced when trying to design an interface that aims to balance expressivity and accessibility.

The process of developing this instrument has generated a number of ideas for further designs and iterations. This includes adding sensors whose triggering would temporarily distort and modify the base pattern in real-time, or enabling the modification of the sequencing algorithm itself rather than only the parameters, increasing the scope of variation available.

6 Ethical Standards

The instrument was developed by the first author under the supervision of the second and third authors, and did not involve any other participants. There are no known conflicts of interests.

References

[1] Alexander Dupuis and Carlos Domínguez. 2014. Digitally Extending the Optical Soundtrack. In *International Conference on Mathematics and Computing*. https://api.semanticscholar.org/CorpusID:7117576

[2] Mark Fell. 2013. Works in sound and pattern synthesis - folio of works. (2013).

[3] Mark Fell. 2024. *Structure and synthesis: The Anatomy of Practice*. Urbanomic Media, Falmouth, England.

[4] Behzad Haki, Nicholas Evans, and Sergi Jordà. 2024. GrooveTransformer: A Generative Drum Sequencer Eurorack Module. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Utrecht, Netherlands, Article 39, 5 pages. https://doi.org/10.5281/zenodo.13904848

[5] Jiffer Harriman. 2012. Sinkapater -An Untethered Beat Sequencer. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. University of Michigan, Ann Arbor, Michigan. https://doi.org/10.5281/zenodo.1178277

[6] Robert Hasegawa. 2020. Creating with Constraints. In *The Oxford Handbook of the Creative Process in Music*. Oxford University Press. https://doi.org/10.1093/oxfordhb/9780190636197.013.17

[7] Niklas Klügel, Timo Becker, and Georg Groh. 2014. Designing Sound Collaboratively Perceptually Motivated Audio Synthesis. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. London, United Kingdom, 327–330. https://doi.org/10.5281/zenodo.1178833

[8] Ron Kuivila. 2025. Events and Patterns. In *The SuperCollider Book*, Scott Wilson, David Cottle, and Nick Collins (Eds.). The MIT Press.

[9] Erik Larsson. 2000. Synthesis of drum and percussion sounds by using frequency modulation. (2000).

[10] Zhengyang Ma, Iurii Kuzmin, Duan Ruilei, and Raul Masu. 2024. Pharosphones: interactive audience participation using light. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Utrecht, Netherlands, Article 70, 6 pages. https://doi.org/10.5281/zenodo.15016916

[11] Mark Marrington. 2010. Experiencing musical composition in the DAW: The software interface as mediator of the musical idea. In *Proceedings of the 6th Art of Record Production Conference*, Vol. 8.

[12] Lloyd May, Lateef McLeod, and Michael Mulshine. 2024. Bishop BoomBox: A Physically Accessible Drum Machine. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Utrecht, Netherlands, Article 6, 7 pages. https://doi.org/10.5281/zenodo.13904774

[13] James McCartney. 2002. Rethinking the Computer Music Language: Super Collider. *Computer Music Journal* 26, 4 (2002), 61–68. http://www.jstor.org/stable/3681770

[14] Alex McLean. 2014. Making programming languages to dance to: live coding with tidal. In *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design* (New York, NY, USA, 2014-09-03) *(FARM '14)*. Association for Computing Machinery, 63–70. https://doi.org/10.1145/2633638.2633647

[15] Alex Mclean. 2020. Algorithmic Pattern. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Birmingham City University, Birmingham, UK, 265–270. https://doi.org/10.5281/zenodo.4813352

[16] Eleonora Oreggia. 2024. Exploring the Space Between Instrument and Controller. In *Proceedings of the 21st Sound and Music Computing Conference*. Zenodo, 486–490. https://doi.org/10.5281/zenodo.14362248

[17] Simon Reynolds. 2010. The History of Our World: The Hardcore Continuum Debate. *Dancecult* 1, 2 (2010), 69–76. https://doi.org/10.12801/1947-5403.2010.01.02.05

[18] Jan C Schacher. 2012. The body in electronic music performance. In *Proceedings of the Sound and Music Computing Conference*. 194–200.

[19] Robert Strachan. 2017. *Sonic Technologies*. Bloomsbury Academic. https://doi.org/10.5040/9781501310652

[20] Godfried Toussaint. 2005. The Euclidean algorithm generates traditional musical rhythms. In *Renaissance Banff: Mathematics, Music, Art, Culture*. 47–56. https://archive.bridgesmathart.org/2005/bridges2005-47.html

[21] Richard Vogl and Peter Knees. 2017. An Intelligent Drum Machine for Electronic Dance Music Production and Performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Aalborg University Copenhagen, Copenhagen, Denmark, 251–256. https://doi.org/10.5281/zenodo.1176238

[22] Nick Warren and Anil Çamci. 2022. Latent Drummer: A New Abstraction for Modular Sequencers. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Auckland, New Zealand, Article 38. https://doi.org/10.21428/92fbeb44.ed873363

[23] Xiaowan Yi and Mathieu Barthet. 2025. The Drum Machine of Tao. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Canberra, Australia, Article 79, 4 pages. https://doi.org/10.5281/zenodo.15698954

[24] Çağrı Erdem and Carsten Griwodz. 2024. dB: A Web-based Drummer Bot for Finger-Tapping. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Utrecht, Netherlands, Article 12, 11 pages. https://doi.org/10.5281/zenodo.13904788