Reshaping Time - Exploring grid interfaces for ansiorhythmic patterns

Adam Tindale
OCAD University
100 McCaul Street
Toronto, Canada
atindale@ocadu.ca

Colin Clark
Institute for Research and Development on Inclusion and Society
20-850 King Street West
Oshawa, Canada
colin@colinclark.org

ABSTRACT

Grid layouts are popular in music controllers. While they are excellent for many tasks, they usually require a predefinition of a single subdivision to be represented. In this paper we explore techniques for visually representing rhythmic phrases that have multiple subdivisions and tuplet groupings within a grid interface. The paper proposes an ansiorhythmic (dissimilar pattern) grid notation system for expressing rhythms that may vary their length, subdivision or phase within a grid structure that has historically been limited to one subdivision per sequence. A proof-of-concept Max/MSP demonstrates a tactile interface for a polyrhythmic, polymetric, and polyphasic sequencer.

Author Keywords

Musical notation, grid controllers, tactile interfaces, music sequencer, step sequencer

CCS Concepts

•**Applied computing** → **Sound and music computing;** Performing arts;

1. INTRODUCTION

Rhythm can be imagined as the events that occur in relation to a pulse. A pulse can be divided into any number of subdivisions and variations of patterns. Grid layouts are popular in music controllers - they make it very easy to turn on and off events in a subdivision in order to create patterns, they leverage the tactility of a physical interface. While grids are excellent for many tasks, but sequencers utilizing grids often only provide a single subdivision to be represented. In this paper we explore techniques for visually representing rhythmic phrases that have multiple subdivisions and tuplet groupings within a grid interface, and explore its use in performance. The paper proposes an ansiorhythmic grid notation system for expressing rhythms that may vary their length, subdivision, or phase within a grid structure. The goal of the project is to provide a method to express polyrhythmic, polymetric, and polyphasic patterns that can be implemented with either software or hardware.

While there is no shortage of software and hardware interfaces available that are able to sequence events for musical uses, in many of them it is not a primary function to be able to define rhythms with arbitrary tuplet groupings. Many sequencers focus on rhythms that are a power of 2 divisor of the beat. Many also offer rhythms that are a power of 3 divisor of the beat, but most do not offer a way to easily move between these subdivisions, if they offer this capability at all. Very few offer a divisor of 5 or 7 or 9.

With many grid-based interfaces the choice of subdivision has to made when the sequence is initialized, and some offer the ability to change, but that usually means that the divisions for the whole track are changed, rather than one beat. There are interfaces that overcome this limitation but most often these require some menu diving, complex uses of slices and chains, or other workarounds rather than providing changing subdivisions as a primary interaction modality.

In Bjork's *Hunter* from the album Homogenic the drum pattern was created by Mark Bell on a Roland 909. The majority of the track utilizes a sixteenth note pattern that creates variation with a variety of accent levels and distributing notes between the bass drum and snare drum sound. Embellishments are added throughout the track that are seemingly impossible without modern sequencer features such such a micro-timings. Bell is able to create ruffs by cloning the main pattern and breaking it into sections, changing the subdivision of a section to thirty-second notes and inserting extra notes to perform the ruff, and then chaining these patterns back together on the fly. A similar tactic is employed to perform the quarter-note triplet that appears but to realize this figure Bell creates a pattern that changes the tempo of the machine[1]. Bell's performance demonstrates knowledge and control of both the Roland 909 and the ability of varying subdivisions to effect the flow of a piece of music to magnificent effect.

If a performer is able to move easily between subdivisions would they utilize them more often, and could it help them develop their own voice? Gander proposes that learning more about polyrhythms provides performers with tools they need to expand their musical vocabulary and develop an identifiable ideolect. "In my approach, a gridded framework of metric precision is a necessary first step to fluid integration of complex rhythmic language, and I develop this area of vocabulary in reference to a digitally fixed temporal source ... prior to any relaxation of this constraint".[7] While Gander is referring to drumset performers, much of the same logic applies to electronic musicians.

[1] `https://www.youtube.com/watch?v=Ix4U9eVDJk0`

NIME'24, 4–6 September, Utrecht, The Netherlands.

2. BACKGROUND

Sequencer instruments allow performers to express patterns through creation, chaining, modification, and other transformations of the component materials. Sequencer instruments use some form of linear representation of time, often with an added dimension to represent pitch or intensity[3]. This project shares the aims of many sequencer projects by creating an interface to express a multitude of rhythms[1, 8, 12]. The circle provides the potential to express the infinite possibilities of rhythm within a cycle [10, 9]. The circle is an "interface that can generate a wide variety of rhythmic configurations, including ones that evade isochronous divisions of a cycle." [4] Conceptualizing a cycle as a circle goes back hundreds of years to Safi al-Din Urmavi who wrote the kitāb al-Adwār in the 13th century and in it he details necklace notation, where events can be placed around a circle [2].

Unfortunately, grids are finite spaces that are unable to represent infinite possibilities. The Monome[6] grid interface was a watershed moment in sequencer interfaces, providing inspiration for both music making, and development of idiomatic sequencer creation, stemming from the open-source nature of the project and its clear focus on sharing of ideas as well as music[14].

Grid-based tactile hardware MIDI controllers are widely available commercially and usually includes visual feedback in the form of lighting the individual buttons, such as the Monome, Ableton Push, MPC, Beatstep (and Pro), Launchpad. Most of these interfaces also include buttons around the grid. Grid controllers are especially amenable to changes because they represent a simple and adaptable interface. Instead of proposing a new piece of hardware this project proposes a new method of using grids to express rhythmic patterns that are difficult or impossible to achieve with other solutions. As Bill Buxton says: "Devices, then, are chosen for their range of applicability." [5]

3. EXAMPLES OF GRID NOTATIONS

Ansiorhythmic Grid Notation works by allowing the user to freely define regions on the grid that correspond to rhythmic sequences. One dimension of the grid represents the number of beats and the other dimension represents the comprising tuplet of the beat. While these orientations are flexible, for the purposes of this paper the columns or Y dimension will represent the beats proceeding from bottom to top, and the rows or X dimension will represent the tuplets comprising each beat oriented from left to right. For example, a rhythm comprising of a quarter note, then two eighth notes, and then five quintuplets, making eight notes spaced over three beats would be expressed as a single cell for the first beat, two cells for the second beat, and five cells on the third beat (see Figure 1).

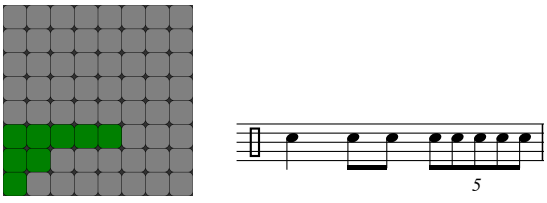

Figure 1: A pattern mixing tuplets with the grid representation shown on the left and the western notation shown on the right.

The initial purpose of the ansiometric grid notation system was to express complex rhythms, upon explorations there was also an opportunity to express simple rhythms using less space on the grid. For example, the four-on-the-floor bass drum pattern of quarter notes, typically in four/four common time, can be expressed as a set of four cells in a region, similar to venerable line piece in Tetris. Given that the pattern may be made up of a single repeating item, this shape can be further reduced a single repeating cell (see Figure 2).

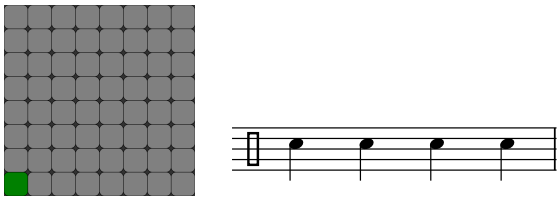

Figure 2: A four on the floor pattern with the grid representation shown on the left and the western notation shown on the right.

Since repeating patterns can be expressed using only the number of cells required, rather than selecting the impulses in a predefined set of locations, there can be a great deal of space left over in a grid. The patterns are expressed relative to themselves, so their location within the grid is not tied to their representation inside of a system. This allows for patterns to be placed anywhere on the grid, and also for many patterns to be placed on the same grid. It is this property of the notation that allows for both polymetric (patterns of different or varying lengths) and polyrhthmic (patterns emerging from multiple voices playing patterns with different subdivisions) to be expressed simultaneously utilizing the primary mode of input (See Figure 3).

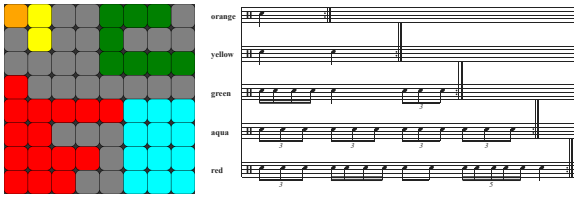

Figure 3: Multiple patterns with varying lengths and tuplet divisions (each colour denotes a different pattern) with the grid representation shown on the left and the western notation shown on the right.

3.1 Creating Patterns

The creation of a sequence requires the user to touch one or more cells on the grid which creates a region consisting of the enclosed cells. The creation of the region occurs when any touch from the grid is released. The lower left cell of the region is assigned to the origin and the start of the sequence it represents. Should a user touch the lower left cell and the cell four up and four across from there, a region of 16 cells with four rows and columns is created. This represents a sequence of four groups of sixteenth notes, should the beat duration be set to a quarter note. A single cell region, like the one in Figure 2, is created by touching one cell and releasing. A pattern like the one in Figure 1 is created by touching three places on the grid: the origin of the bottom left, the right hand edge of the group of two, and the right hand edge of the group of five.

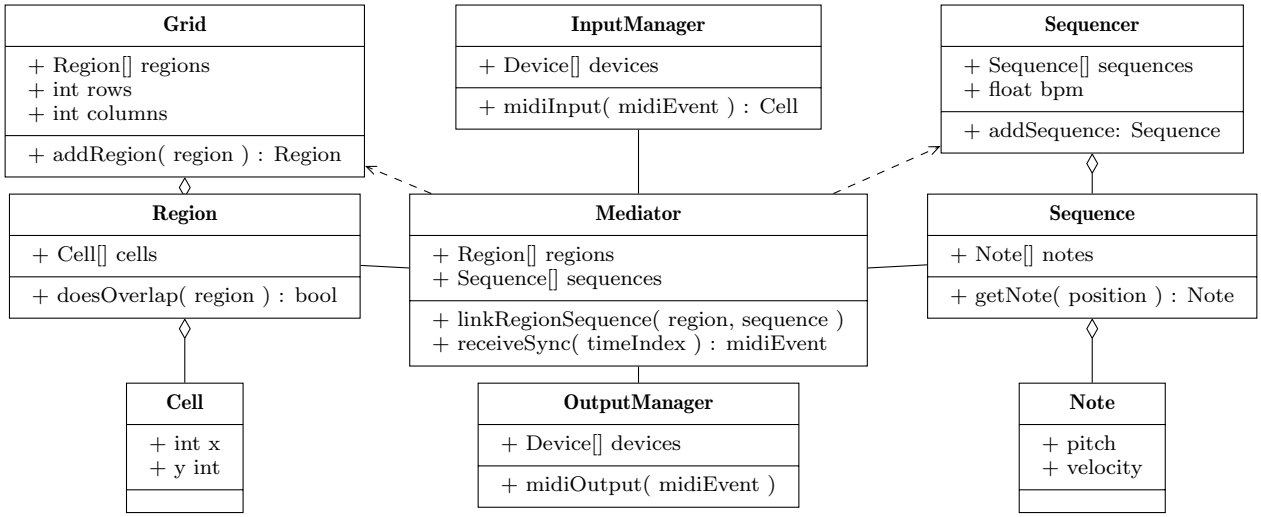

Figure 4: Class relationship diagram of the software architecture.

3.2 Modifying Patterns

If a region that is created overlaps with an existing region then the new region will not be added to the grid. The exception to this rule is the case of modifying an existing region. To modify a region after it has been created, the user can touch the left hand side of a region, which corresponds to the cell that aligns with the beat divisions of the correlated sequence. If a new region aligns with the beat divisions then it will modify the region instead of creating a new one or being deleted. For example, another way to create the pattern in Figure 1 would be to create a region of three quarter notes, then to create a region of two that was aligned with the second beat, and a region of five on the third beat. This region can conversely be modified back by repeating the original three quarter note gesture and the cells will be freed.

3.3 Adjusting Pattern Phase

The ability to adjust the phase of the sequences allows for methods of expression, such as displacement patterns on the drumset, or phase music such as the compositions of Steve Reich. Each region has its own phase, and with multiple running sequences it is possible to express polyphasic patterns in addition to the polymetic and polyrhythmic patterns already explored.

By default sequences begin at their origin and proceed to their endpoints. The starting point may be adjusted by entering a mode where a touch within a region indicates that the cell will now represent the starting phase of the sequence. For example, by touching the third note in the pattern in Figure 1 then the sequence will start from the third eighth note, or halfway through the bar.

This functionality requires the application to signal a change of input mode to the Mediator class in Javascript. Input modes are available by a drop-down menu in Max/MSP or using a physical key on hardware. While this choice is sub-optimal, it was chosen to ensure that all cells in the grid are available for pattern expression.

4. IMPLEMENTATION

The proof-of-concept ansiorhythmic grid notation software is implemented in two layers: a Javascript layer of classes to manage state, and a Max/MSP layer providing a simple polyphonic step sequencer. The rationale for the software to

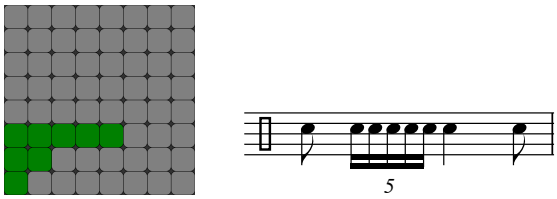

Figure 5: A mixed tuplet pattern that has been phase shifted to the third note with the grid representation shown on the left and the western notation shown on the right.

be implemented in separate layers is that the Javascript allows for an Object-Oriented approach of the translation between visual and temporal representation. Max/MSP was chosen because it has the ability to host Javascript, web applications, and manage hardware connections. There are three main categories of classes: grid representation classes, sequencer representation classes, and intermediary classes that are responsible for the mediation, representation, and synchronization between the visual notation and the sequences (see Figure 4).

The grid object contains regions and the region class contains cells. Each class has helper functions that query whether an input parameter overlaps with its own representation. The region class contains logic for situations where a region may overlap in such a way that it augment its shape with the region being compared. Likewise, the sequencer object contains sequences and the sequence class contains notes. The sequence class contains methods for expressing shape array notation as indices for the sequencer and managing the indices with the corresponding note objects.

Grid, region, and cell objects work to represent the visual notation and logic. Sequencer, sequence, and note objects represent the musical events. The InputManager and OutputManager classes manage the communication and data representation between the application and the hardware, or software, inputs and outputs. The Mediator class is primarily responsible for bridging the visual and musical representations, but is also responsible for handling the transformations from gestural inputs to new shapes for creating or manipulating patterns. As the Sequencer class runs the Mediator class receives synchronization messages when individual notes are triggered that are resolved with the corresponding cells in the grid representation. When a Cell

is changed from active to non-active the Mediator class informs the OutputManager and all software and hardware connected is synchronized.

4.1 Sequencer

The Max/MSP patch created for this project implements a polyphonic step sequencer using a phasor signal, modelling the implementation presented in [13]. A central phasor is used to keep track of the main beat clock, which is then duplicated into a signal per sequence using the *mc* objects (See Figure 6). Each signal undergoes a transformation of rate to correspond to its subdivision using the *rate* object. However, the implementation utilizes the newer *what* object with Javascript code providing a list of event locations corresponding to each event. The end result is still a series of impulses and indices which are fed back to the Javascript code for synchronization. The Max/MSP patch hosts a P5.js sketch that can be used as both the interface and the visual display of the sequence, in addition to synchonizing with an Ableton Push controller, if it is connected at the time.

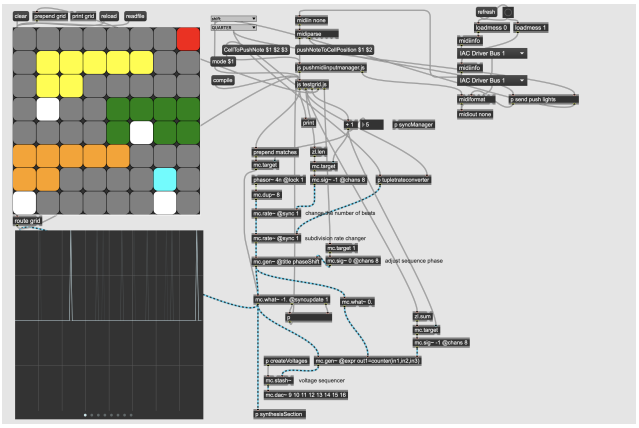

Figure 6: A screenshot of the Max/MSP displaying the signal flow, and live representation of the grid state.

5. SUMMARY AND FUTURE WORK

Grid interfaces are excellent at providing tactile interfaces for expressing rhythms in a live running sequencer. The ansiorhythmic grid notation system presented in this paper demonstrates a tactic to allow performers to express complex rhythms as a first order operation (ie. without "menu-diving") without sacrificing the immediacy of the grid to express and edit sequences quickly. Furthermore, this paper has explored opportunities to compress the notation of isometric rhythms into smaller portions of the grid, allowing for a grid to represent many sequences simultaneously.

A proof-of-concept Max/MSP patch implements a flexible polyphonic signal-based sequencer that can easily express the complex and varying rhythms, while managing inputs from hardware and software inputs, and synchronizing visual output. The current implementation provides the ability to switch modes, but many more modes can be added, such as sequence chaining, while also exploring techniques like gestures to switch modes[11].

There are many rhythms which are difficult or impossible to express with this notation, such as nested tuplets and micro-timings. Nested tuplets provide an incredible opportunity for expression, and the programming language NestUp[15] provides a brilliant and compact notation for their expression. While tuplets are easily expressed in the ansiorhythmic grid notation, rhythms inside of a tuplet could be imagined as a type of branching in an orthogonal direction, thought introduces potential collisions. While these options are viable tactics, the loss of coherence in the notation indicates one of the many potential limits of the notation system.

This project will continue to ruminate on rhythmic expression with grids. We offer this tool and its thinking to the reader to help them explore their own ideas to express rhythms.

A github repository is provided at:
`https://github.com/drart/gridquencer`

6. REFERENCES

[1] R. Adeney and A. R. Brown. Improvising with grid music systems. In *Improvise: The Australasian Computer Music Conference*, pages 102–110, 2009.

[2] S. al-Dīn al Urmawī. *Kitāb al-Adwār [Book of Cycles]*. Mawṣil, A.H. 1103 (1692), Bagdad, 1695.

[3] R. Arar and A. Kapur. A history of sequencers: Interfaces for organizing pattern-based music. In *Sound and Music Computing Conference, Stockholm, Sweden*, 2013.

[4] S. Barton. Circularity in rhythmic representation and composition. In *Proceedings of the New Interfaces for Musical Expression Conference*, pages 505–508, 2020.

[5] W. A. Buxton. There's more to interaction than meets the eye: some issues in manual input. *Human-Computer Interaction*, pages 122–137, 1987.

[6] b. crabtree and k. cain. monome. https://monome.org, 2006-present.

[7] A. J. Gander. *Developing a Polyrhythmic Idiolect*. PhD thesis, The University of Sydney, 2017.

[8] S. J. Hunt. Exploring polyrhythms, polymeters, and polytempi with the universal grid sequencer framework. In *Proceedings of the 15th International Audio Mostly Conference*, AM '20, page 101–106, New York, NY, USA, 2020. Association for Computing Machinery.

[9] V. Malmberg et al. Iris: A circular polyrhythmic music sequencer. Master's thesis, Aalto University, 2010.

[10] A. J. Milne. Xronomorph: Investigating paths through rhythmic space. *New Directions in Music and Human-Computer Interaction*, pages 95–113, 2019.

[11] B. Rossmy, S. Unger, and A. Wiethoff. Touchgrid–combining touch interaction with musical grid interfaces. In *Proceedings of the New Interfaces for Musical Expression Conference*, 2021.

[12] B. Rossmy and A. Wiethoff. Musical grid interfaces: Past, present, and future directions. In *Proceedings of the New Interfaces for Musical Expression Conference*, 2021.

[13] G. Taylor. *Step by step: adventures in sequencing with Max/MSP*. Cycling '74, 2018.

[14] O. Vallis, J. Hochenbaum, and A. Kapur. A shift towards iterative and open-source design for musical interfaces. In *Proceedings of the New Interfaces for Musical Expression Conference*, pages 1–6, 2010.

[15] A. Van Gils and S. Tarakajian. Nestup. https://nestup.cutelab.nyc, 2020.