Audio Engineering Society
Student Project Expo Paper
Presented at the AES 158th Convention
2025 May 22–24, Warsaw, Poland



# gemuPG - A Generative Music Poly* Grid

Aron Rocco Petritz[1,2]

[1]*University of Music and Performing Arts Graz*
[2]*University of Technology Graz*

Correspondence should be addressed to Aron R. Petritz (aronrocco@posteo.net)

**ABSTRACT**

Conventional music software, like Digital Audio Workstations (DAWs) and notation tools, adhere to established rules of traditional Western music composition, which limits creative composition in rhythmic and tonal dimensions. The new open-source music environment *gemuPG* aims to explore the capabilities of a software-centric approach to music making that is not bound to traditional approaches. The design focuses around inherent use of polymeters, polyrhythms and microtonality. The term *poly** is therefore introduced, which groups these concepts together. Additionally, generative approaches are discussed. A grid based interface rethinks audio tracks as areas on a two-dimensional grid. Generator blocks can be placed within these areas to create virtual instruments through additive synthesis. Sequencer blocks are placed on the sides of the areas to create the respective area's sequence. The length of each area's sequence therefore corresponds to its circumference. This structure enables flexible rhythmic subdivisions and layered temporal patterns across multiple independently behaving areas.

The software was written in C++ using SDL3 and ImGui. Generator blocks support simple wave shapes, as well as sample loading. Sequencer blocks offer four types that allow for microtonal sequencing. Some features like chord sequences, effect and modulator blocks, or conditionals are yet to be implemented, but would increase the two-dimensional design philosophy and especially the generative aspect greatly. Examples nonetheless demonstrate *gemuPG*'s unique affordances for minimalist, polymetric, and generative music composition.

## 1 Introduction

Traditional music software, such as DAWs and score editors, is largely constrained by the conventions of Western music theory: shared measure lengths, rhythmic subdivisions in multiples of two, and equal temperament tuning systems. These structural constraints inhibit creative exploration. *gemuPG* (**Ge**nerative **Mu**sic **P**oly* **G**rid) is a new software environment offering inherently polyrhytmic, polymetric and microtonal approaches. Here, I suggest *poly** (pronounced 'poly asterisk', or simply 'poly') as an umbrella term, which does not suit the microtonal aspect semantically, but encapsulates the concept adequately. Furthermore, I picked the term generative to describe procedural music. There are already many examples of music software referred to as 'nonlinear'. Although a lot of this nonlinear software can also be considered poly* and vice versa, the term poly* differentiates itself by focussing on forms of more granular pitch and time subdivision rather than breaking away from linear structural norms. With gemuPG I desired to create an accessible music environment that is generative and poly* at its core. Additionally it should not mimic physical hardware

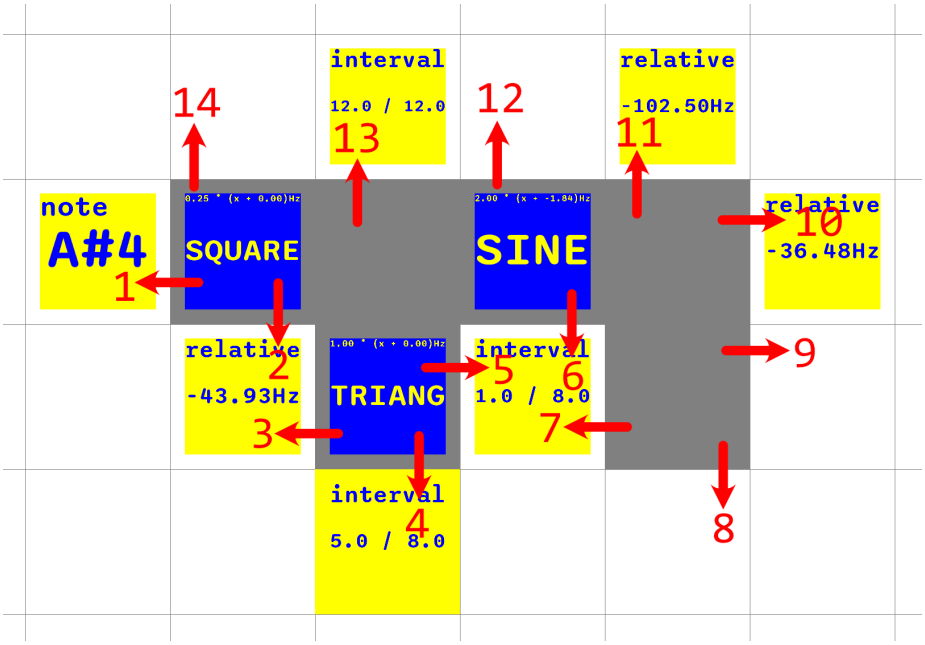


**Fig. 1:** Example area showcasing the sequence order.

but instead offer a software-centric interface, that is easy to achieve results with. The software is available under the GNU AGPLv3 license in the git repository of the Institute of Electronic Music and Acoustics: `https://git.iem.at/aronpetritz/gemupg/-/releases/Thesis`

## 2 Functionality

For the interface a grid was chosen, which effectively reimagines audio tracks as two-dimensional areas. Generator blocks can be placed within these areas to set up their virtual instrument through additive synthesis. These generator blocks offer a selection of simple wave forms or sample loading for sound generation. Additionally, each block has its own settings window that can be opened through left-clicking, where the amplitude, frequency multiplication factor and relative frequency change can be set next to the waveform. Generator blocks can also be placed outside of areas for continuous notes.

Sequencer blocks are placed around the sides of the areas, which creates a note sequence that loops at a length equivalent to the area's circumference in a counter-clockwise direction, an example of which is shown in Figure 1. The following sequencer block types are available and can also be seen in Figure 2:

- **absolute**: absolute frequency value
- **relative**: relative frequency change, based on previous pitch
- **interval**: interval as a pair of number of equal divisions of the octave (edo) and steps of given edo scale
- **note**: pitch selection from the twelve-tone scale (A = 440 Hz)

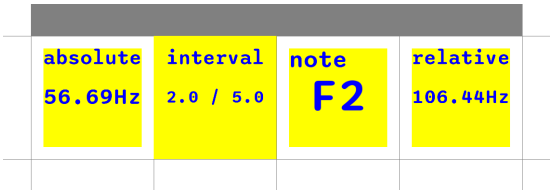


**Fig. 2:** Sequencer blocks of all types.

Notably, an area's frequency wraps around [20 Hz, 20 kHz], allowing for semi-permanently rising or falling loops, an example of which is shown in Figure 3. An overhaul of the 'note'-type sequencer that aligns more with the microtonal aspect of the

design philosophy is desired. The values of sequencer and generator blocks are randomised within reasonable ranges. Importantly, the sides of the areas are checked rather than its adjacent fields, leading to a sequencer block placed in an area's corner being stepped multiple times successively.

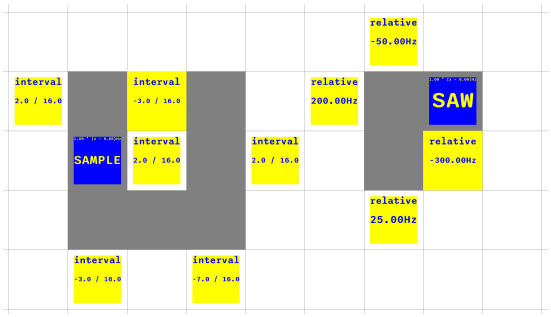


**Fig. 3:** A simple generative project.

Areas themselves offer settings windows as well, shown in Figure 4. In order to support polyrhythmic patterns a 'subdivision' setting was added here, which can rather be seen as a tempo multiplier, in relation to the global BPM setting. Furthermore, area wide amplitude controls are available, as well as glissando, attack and release controls, which function in relation to the area's sequence step length.

The user interface offers a placement type (area, generator or sequencer) selection toolbar on the left side, as well as an indicator showing the current selection in the corner above. At the bottom of the window is another toolbar for playback controls. It contains play/pause and stop buttons, as well as global tempo and volume

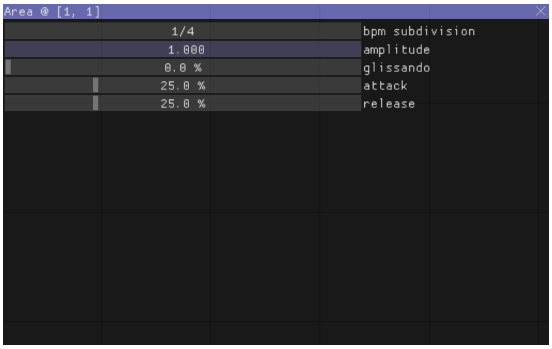


**Fig. 4:** The area settings window.

controls. Lastly, an oscilloscope which can be toggled via a keyboard shortcut is placed in the bottom right corner. All shortcuts are listed in the appendix. An empty project window showing the mentioned interface elements can be seen in Figure 5.

Copy and paste functionality for blocks was implemented, which greatly improves placement speed, especially for sequencer blocks, since it allows for presetting the sequencer type and settings. Lastly, an important addition was the implementation of save and load functionality, which allows users to save their projects in a JSON format for easy sharing and reloading of projects. This ensures that compositions can be revisited and further developed at any time.

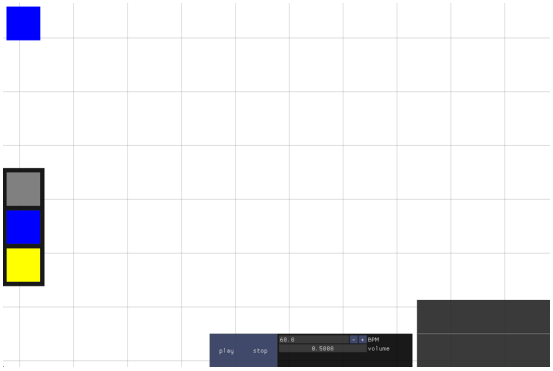

**Fig. 5:** An empty gemuPG window.

## 3 Unimplemented Features

Some planned features were omitted, as is made evident by early concept art seen in Figure 6. Namely, effect blocks, which would affect areas as a whole, parallel sequencer blocks, allowing for chords, and stereo audio with panning settings for generator blocks. Furthermore, modulator blocks, which could affect variables of up to four adjacent generator, effect, or other modulator blocks were also considered. These would allow for frequency or ring modulation synthesis. Additionally, conditional statements - possibly also implemented in the form of blocks within areas - could toggle other blocks or variables based on various factors, such as the amount of loop iterations passed, the current frequency, or even just based on a random distribution. Lastly, selectable mute groups for areas and generator blocks would allow gemuPG to be used as a versatile live performance instrument.

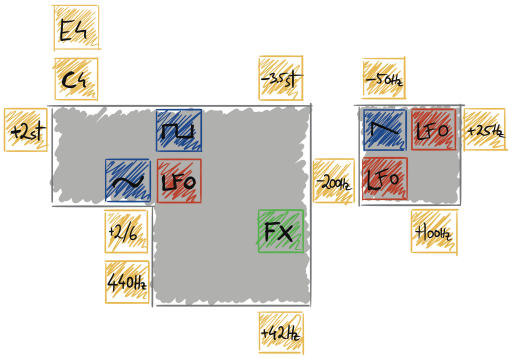


**Fig. 6:** Early concept art for gemuPG.

## 4 Instrumental Idioms

Through testing the software during development several idioms of became evident. Firstly, polymetric music was typically created unless the areas' circumferences were intentionally matched. Contrary to this, the polyrhythmic aspect, being 'hidden' in a menu, is perceived as an option rather than a natural occurrence. Furthermore, the randomisation of blocks encourages quick experimentation and a 'seeing what sticks' approach. The looping structure of gemuPG leads to a strong tendency for repetitive music, even if not fully repeating due to the underlying polymetric design. This lends itself to composition of minimal music, sound scapes and similar forms without strong musical motion. Alternatively, gemuPG can also be used as an instrument, rather than a compositional tool, where the output is recorded and combined with other tools in a DAW. Most notably however, the central role of loops limits support for any advanced musical forms. Although sufficiently large areas that allow for these forms could be declared, it would directly oppose gemuPG's usability, making traditional music software the better alternative in these cases.

## 5 Technical Issues

The software was programmed in C++ using *SDL3* and *Dear ImGui*. Unfortunately, the clock for sequence stepping is updated in the rendering loop, causing strong tempo variations on slower machines. Moreover, various audio glitches are sometimes audible when using the attack, release and glissando settings. It was concluded that this also is an issue related to the suboptimal implementation of the sequencer clock.

## 6 Summary

gemuPG offers several novel ideas that lead to new compositional approaches. The current state of the software provides a functional foundation and successfully delivers a natural poly*-centric environment for music creation. However, the absence of many mentioned features, the implementation of which would allow for greater musical complexity, limits generative composition. Future development should prioritize integrating these omissions, particularly modulator blocks and conditional statements, which would allow for more nuanced generative variation. Altogether, gemuPG can be seen as a playground for new forms of musical expression with a lot of room for improvement.

## Appendix: Controls

| Shortcut | Function |
|---:|---|
| 1 | Area Placement |
| 2 | Generator Placement |
| 3 | Sequencer Placement |
| Left Mouse Click | Place Block |
| Left Mouse Click | Edit Block |
| Right Mouse Click | Remove Block |
| Middle Mouse Drag | Pan Camera |
| Ctrl + Left Mouse Drag | Pan Camera |
| Scroll Wheel | Zoom Camera |
| Ctrl + C | Copy Block |
| Ctrl + V | Paste Block |
| Ctrl + S | Save Project |
| Ctrl + Shift + S | Save Project As... |
| Ctrl + O | Open Project |
| Ctrl + N | New Project |
| Q | Close All Settings |
| Escape | Close All Settings |
| D | Toggle Oscilloscope |
| F11 | Fullscreen |