Modularity and Generative Art

Assistant Professor Chad Eby, M.A., M.F.A.
School of Art and Visual Studies, University of Kentucky, Lexington, United States of America
https://chadeby.studio
e-mail: chad.eby@uky.edu

Abstract

Modular tools and processes often occupy significant roles in various types of generative art practices, whether implemented in hardware or software.

This paper provides a definition of modularity, a brief history of modular systems and examines a r ange of modular strategies in order to discuss what aspects may make different modular paradigms more or less conducive to generative work.

Issues of intention, connectivity, complexity, structure vs. content, and overdetermination are discussed in the context of the effectiveness of different modular paradigms for generative work.

Introduction

The aim of this paper is to introduce modular artmaking systems and make a first run at an exploration of the history and special characteristics of a particular concept of connected modularity and how its instantiation as specific modular systems—in hardware, software, and hybrid forms—may be particularly well-suited to pursuits in generative artmaking. The enormous popularity of node-graph-based modular metaphors (the abstracted software side of modular, hereafter referred to just as 'node-based') in art-making tools and systems, particularly those with significant capacity for autonomous operation, points to an at least perceived affinity between modularity and g enerative art. The evolution and typical characteristics of these systems is examined in this paper with a par ticular emphasis on m odularity in sound synthesis as represented by the Eurorack format, both physical and virtual.

Defining Modularity

Modularity is broadly defined by Melissa Schilling as "…a continuum describing the degree to which a system's components can be s eparated and recombined, and it refers both to the tightness of coupling between components and the degree to which the 'rules' of the system architecture enable (or prohibit) the mixing and matching of components." [1] Although Schilling's primary interest is in developing a general theory of how and why systems become more or less modular over time, this definition and subsequent terminology, borrowed from a v ariety of

disparate fields, is a reasonable starting point for considering modular artmaking systems.

An important consideration in a discussion of the suitability of particular modular systems for generative work involves what Schilling calls *synergistic specificity*: “The degree to which a system achieves greater functionality by its components being specific to one another.” [2] At the extreme low end of synergistic specificity, modules are utterly agnostic In terms of their preferred connections to one another. This leads to far greater freedom in recombination of modules (and implies a general and limited scope of functionality per module) but comes with a possible loss of optimal functionality at the system level. Conversely, high levels of synergistic specificity may contribute to more optimized overall system functionality, but at the cost of overly opinionated modules (more on this later) and much more severe restrictions on recombination possibilities.

In the terms of generative art, however, it is critical to interrogate the implications of what precisely is meant in this theory of modularity by “optimal.” A central tenet of making generative art with autonomous systems is that that the artist must cede some degree of autonomy to the system [3]. This giving up of autonomy, among other effects, creates a capacity for the artist to be surprised by the system’s output—something that would certainly be understood as much less than “optimal” in Schilling’s management context.

One implication of a decrease in re-combinatory possibility with an increase in synergistic specificity is that composability, and thus the condition of possibility for “type 2” [4] emergent behavior (that which does not arise from any one component in isolation, but only in the connected aggregate) may be lost. I would argue that, for the purpose of generative art, encouraging the possibility of emergent behavior in a system through more promiscuous (and less opinionated) modules is to be desired over efforts toward synergistic specificity in service to an optimal system.

The collection of modules for use in a particular modular artmaking system will often exhibit a range of different functions, purposes, or capabilities with fuzzy edges—some purposes will be unique to one particular module, and some, with varying degrees of overlap, with other modules available to the system. Modules in these types of systems are not designed primarily to be swapped-out with other identical modules (in case of failure or for production efficiencies as is the case in a manufacturing concept of modularity) but rather to offer shades of difference to a particular function or transformation in the flow of a modular system. This is expressiveness of transformation is enhanced within a system possessing modules with multiple variations of a single purpose or function, facilitating experimentation by exchanging modules with different but similar functionality or effect.

At some level, children’s building blocks, both traditional wooden sets and those snap-together plastic ones (as well as tangrams, other tiling systems, and even strung beads) might meet the criteria of generative modular system as so far expressed. But while these examples may provide a limited platform for working generatively with autonomous rules systems, they lack the feature of

components communicating with each other within a modular system.

In Schilling's definition, the word "coupling" highlights that in a modular system the individual components must be somehow connected one to another—due to the limited functionality of each individual component, a module in isolation (by design!) is seldom useful.

Connections, whether linear, branching, or recursive, imply a flow of signal or data and define the relationships between and among connected components that shape the higher-level behaviors of a given modular system.

One effect of these two characteristics taken together (a multiplicity of modules and the necessity of connection) is that spatial arrangements and connective topologies often become significant factors in the way a modular system used in artmaking both presents and functions. The "rules" or "system architecture" of an autonomous rules-based system may be external but are often encoded directly into the number and manner of possibilities for connecting modules—whether through arrangement and number of inputs and outputs, connection types or "flavors," or spatial or type constraints imposed by the scaffolding or matrix within or upon which modules in the composite system must be placed.

The generalized modular system under discussion in this paper then consists of a collection of discrete components (modules), of usually limited functionality or purpose each, connected together in some way, which may receive and/or send data and/or signals through those connections, and which are also sometimes contained in some sort of organizing and/or connecting matrix.

The Site of Modularity

Where then might we situate modularity? It seems that modularity is quite obviously a property of the module. And yet, in the field of graphic design, practitioners often work with a 'modular grid' for page layout where the property of modularity rests primarily in the organizing matrix rather than in the content placed within it (which are modular only to the extent that they are spatially constrained by the grid system). In architecture a modular plan is defined by some sort of spatial unit. This confusion of language may have come about since earlier usage of the word 'module' linked it to a measure (the modulus) rather than the thing measured today. In the case of Eurorack modular synthesizers (which we will come to presently), it is interesting that hardware modules themselves come in a number of standard widths (called 'hp' or horizontal pitch) but only a single height of three units (3u) so that they fit into the uniform rack rails that are the system's containing and organizing matrix.

A (Rather) Incomplete History of Modular Systems

Heterogenous, physical "box-and-wire" modular systems originated from the needs of early electronic lab and radio test equipment, telephony systems, and most notably from the 1950s electro-acoustic studios of *Westdeutscher Rundfunk* (WDR) in Cologne, *Groupe de Recherches Musicales* in Paris, and the *Studio di Fonologia* Musicale of Milano (the Milan Electronic Music Studio aka RAI Studio of Phonology) where standalone oscillators, function generators, filters etc., housed in large sheet metal cases, were connected together to produce early electronic sound and music. To get an idea of the sort of equipment used in those early days and how it sounded, please see

Giorgio Sancristoforo's *Berna 3* software [5].

A decade or so later across the Atlantic, these systems began to evolve toward their now more familiar form on the east and west coasts of the United States.

The first system that could be described as a modular sound synthesizer was invented by German Engineer Harald Bode while working for the Estey Organ Company in Battleboro, Vermont in 1959. [6] Called the *Audio System Synthesizer*, Bode's somewhat ungainly machine reimagined the room-sized electronic music studios as a more-or-less portable unit with arrays of input and output jacks with which one could use short wires (with plugs on each end) to connect the various oscillator, filter and amplifier modules in a multitude of easily reconfigurable states or "patches."

Directly inspired by Bode's innovation and developing more or less independently on the east and west coast throughout the late 60s and early 70s, the modular audio synthesizers produced by Robert Moog and Don Buchla further miniaturized and popularized the concept of a portable modular system. Buchla, especially, seemed to have already seen these smaller modular systems as potential platforms for generative work rather than traditionally performed instruments: the Series 100 product eschewed a standard keyboard in favor of various sliders, knobs, and touch-sensitive controls meant to trigger and modulate sequences and parameters. Today, "west-coast" synthesis remains synonymous with a more experimental, often generative, approach to sound design and music using non-traditional controllers and inputs.

It wasn't until 1996 that Doepfer released the A-100 modular system with a format that came to be known as Eurorack. This physical form-factor is now the most dominant (and de facto standard) modular audio synthesis hardware system, and arguably the paradigm most often referenced in the multitude of software node-based interfaces (including both the screw-head-literal and the abstractly metaphorical) that now exist for visual coding environments, sound design software, visual effects and compositing tools, and even 3D modeling programs.

One of the earliest of these on-screen node-based modular systems was the *GRaIL* (Graphical Input Language) software system described in a September 1969 memorandum prepared by the RAND Corporation for ARPA. [7] This novel "Experiment in Man-Machine Communications" employed a light-pen that enabled a user to sketch out algorithms as a collection of box-like modules—connected in the style of a flow-chart—directly onto the glass face of a cathode ray tube monitor. The "flow process chart" was already a familiar fixture in engineering circles, having been introduced to the American Society of Mechanical Engineers by Frank and Lillian Gilbreth in 1921 [8]. But beyond being a sophisticated drawing tool producing for flow charts, *GRaIL* was performative…it could also directly execute the algorithms sketched by the user: the visual block diagram of the structure of the code was also the code itself. This ahead-of-its-time system contained much of the DNA for the many subsequent node-based software interfaces that would follow it.

In 1985, at IRCAM (*Institut de recherche et coordination acoustique/musique*) in Paris, Miller Puckette began work on a graphical programming environment that would eventually evolve into two distinct

lines of software that have come to be known as *Pure Data* (1996) and *Max/MSP* (1997). [9] *Max/MSP*, a commercial product from Cycling '74 (part of Ableton since 2017) and *Pure Data* (often abbreviated to *PD*), a free and open-source cousin to *Max*, perhaps best embody the tangled heritage of node-based modular software systems that grew both out of performative algorithm-defining flow charts and graphical metaphors for interconnected signal generators and processors. This is most obviously manifested in the graphically differentiated communication "wires" in the applications that represent control and signal flow with distinctly different appearances (signal wires in *Max/MSP* are striped and "furry," while control wires are smooth grey vectors).

In fall of 2017, in conjunction with KnobCon, a Chicago-based modular synthesis convention, Andrew Belt released the initial beta version of *VCV Rack*, a virtual modular Eurorack synthesis platform including software emulation of both physical commercial hardware modules and entirely imaginary ones, variably transparent virtual patch cords, and an infinite simulated modular case with visible rack rails.

Though not the first virtual modular software synth (Native Instrument's *Reaktor* (originally released as Generator) has been available since 1996), the free version of *VCV Rack* (now on version 2) is maturing into both a viable alternative to hardware Eurorack, as well as a useful compliment through MIDI to control voltage and control voltage to MIDI modules and circles the story of modular interfaces firmly back toward where they began in sound synthesis.

Eric Hosick has compiled a list of over 100 "visual programming languages" with a screenshot (and an occasional video) of each interface in action. [10] Almost all of these examples qualify as some sort of modular system, and taken together, exhibit an absolutely bewildering array of graphical styles and implementations of the modular metaphor underscoring the proliferation of this way of thinking and working.

The Lure of Modularity

Simply dividing a process into smaller units—say, for analytical reasons—almost immediately suggests new generative possibilities through the selection and recombination of those units. Such was the case with Vladimir Propp, who famously proposed a series of analytical functions that could be used to describe and classify existing Russian magical fairy tales in his 1968 book Morphology of the Folk Tale. Somewhat less famously (and even unknown to some who later worked on similar systems), Propp also described a method to use his functions to generate entirely new fairy tales in what Pablo Gervás argues may be one of the earliest documented descriptions of a creative process described procedurally. [11] There seems to be something about units of story, dis-integrated from their specific narrative arcs, that stimulates a human desire to recode the modular parts into new patterns.

Long before Propp, fortune-telling methods like the I-Ching and the Tarot used modular units to procedurally generate small divinatory narratives. But roughly contemporaneous with Propp, Bode, Buchla, and Moog were developing an altogether different kind of generative modular process based on analytical tools—modular sound

synthesizers inspired, in this case, by deconstructing and re-combining features of radio test equipment originally meant for analysis. The metaphor of 'patching' modular synthesizers, in turn, has spawned generations of the previously mentioned visual programming environments, including those like *Pure Data*, *Max/MSP*, *Touch Designer* and *vvvv* that rely on modules (and their connections) to express and create algorithms; these environments in particular are often used to conceive and produce generative artwork.

Modular systems are especially useful for algorithmic generative art because they tend to be rules-based at multiple levels—both within an individual module, where parameters may be defined, exposed, and modulated, but also at the level of the whole interconnected system of nodes, since the node graph (or patch) will, taken in its entirety, itself describe a larger rules system.

The legibility of the flow of a whole system (at least in simple patches!) is often a better situation for people who have a preference for (and developed skills in) visual understanding than text-based coding would provide. Patching patterns may suggest themselves in terms of visual proximity, alignment, balance, symmetry, or rhythm that might not be apparent when working with a text-based coding language (in the case of software) or menu-diving (in the case of non- or less-modular digital hardware).

The ability to make small changes within a module, or (more often) in the connections between modules, that nonetheless results in large and potentially unexpected changes to the overall behavior of the rules system/patch can be rather seductive.

Modular systems also both encourage and constrain an artist in potentially constructive ways. They provide encouragement because they expose a combinatorial palette for composing algorithms or rules systems as a sort of "kit-of-parts" that may help express intent by hinting at what is possible within the system, as well as avoiding the need for the artist to recall with great precision all of the commands and syntax native to text-based programming environments. Node-based systems also seem to tickle that particularly *homo narrans* [12] itch to create a sort of narrative flow from small parts; call it the "joy of patching."

Constraints can be useful too, particularly when they are baked into the way modules may be connected so as to prevent, in real time, connections that would inevitably lead to undesirable outcomes rather than after-the-fact syntax or compile-time errors. Some modular systems meant to teach children coding (such as MIT's Scratch and Adafruit and Microsoft's *Make Blocks*) keep the kit-of-parts philosophy but forego the boxes-and-wires node graph interface in favor of color-coded puzzle pieces whose shapes are keyed in such a way that they can only be assembled in ways that make syntactic sense. Other systems, like Blender's node-based material editor have color-coded inlets, outlets and wires that visually distinguish data types such as RGB color data, XYZ vector data, complete shaders or single numeric values.

Many node-based systems have modules that evaluate simple logical operations such as AND, OR, NOT, and XOR (exclusive OR) and others that compare signals or data to one another or a fixed value at specific intervals for conditional operations like >, <, or ==. These modules, when connected to other

modules generating periodic, chaotic, or random values can be used to create complex behaviors from simple rules to generate sound, images, geometry, motion, video, etc., or to modulate the parameters of other modules, modulate other modulations, or even modulate themselves. This last capability is particularly useful for generative work, since various forms of feedback or recursion, mixed with other inputs and modulation, can be an extremely effective generative strategy across many forms of media and may give rise to emergent behaviors that are difficult to precisely predict.

A common source of modulation in node-based systems is the LFO, or low-frequency oscillator. These modules produce one or more wave functions or different shapes that can be used to describe rising and falling action over time spans ranging from audio range (so that they may be heard as a tone) up to much longer durations, including a tongue-firmly-in-cheek *Seriously Slow LFO* for *VCV Rack* from Frozen Wasteland that has time base settings ranging from “YEARS” to “HEAT DEATH” [13].

Other useful elements for generative strategies in a node-based environment include patchable sources of chaos (like pendulum and orbital mechanics simulations, Lorenz attractors, fractals, etc.) as well as some sources of uncertainty, such as Bernoulli gates (which shunt an input value to an A or B output based on a (selectable and modulable) probability), various colors of noise functions, and sample-and-hold modules that periodically dip into a stream of values (random or otherwise) and present what returns at an outlet.

Sequencers, a staple of audio synthesizers, are modules that emit a series of fixed values/voltages/colors when ‘banged’ or clocked. These are especially well-suited to generative serial composition strategies, especially when the sequence length is modulated by a second module, or multiple sequences are that are interleaved based on chance operations or other chaotic sequences.

An effective modular system for generative work then will include a large (but not too large!) set of simple, composable modules with multiple variations of common functions, that possess somewhat opinionated connections, and allow for conditional operations, recursion (feedback), complex modulation and chance operations.

Some Drawbacks

Node-based environments are notoriously difficult to “read” once they reach a certain level of splayed-noodle complexity. The same visual and spatial relationships that make node-based systems so powerful and legible to begin with also make maintaining anything beyond a fairly simple patch—or making sense of a patch authored by someone else—particularly challenging. Methods of leaving comments in a patch or documenting its structure do exist in many node-based systems, but often they either feel like afterthoughts, are poorly implemented, or both. Color-coding of inlets, outlets and wires (either baked into the system or as a published convention) can also mitigate visual confusion but is not a completely effective solution.

Modular systems that provide some method of sub-patching (roughly analogous to functions in other text-based programming) ameliorate the

tangle of spaghetti and give a clearer high-level view of a patch, but at the cost of low-level visibility and understanding, essentially making brand-new, functionally overloaded, components.

Indeed, if individual modules are too multifunctional (either by original design or as the result of sub-patching), it becomes easy to lose situational awareness in a patch and also disincentivizes quick substitution of similar patches for experimentation purposes (because too many collateral parameters/interior modules would be lost in the swap).

This is where text-based programming languages shine: a complex but well-commented program with rationally named variables and functions is relatively easy to read, maintain and confidently modify when compared to a complex modular system.

Back on the hardware side of modular, both audio and video synthesis modules tend to be quite expensive on their own. Generally ranging from 50 USD each for the simplest passive modules and up to 1000 USD each (and beyond) for the most sophisticated ones. And since a modular system needs an abundance of modules to be effective, for most people, a hardware Eurorack system is a significant expense.

When It Emerges from the Skronky Murk, the Krell is a Writhing, Cavorting Phantasm

In the 1956 sci-fi film, Forbidden Planet, a rescue mission to Altair IV reveals the remnants of an alien civilization, the Krell, in the form of an enormous subterranean machine and a half-million-year-old recording of a performance by Krell musicians. The soundtrack for the film, including the Krell music, was recorded by Louis and Bebe Barron using a collection of hand-made electronic instruments and tape manipulation. Even though the film's release predated Bode's experiments with modular synthesis by a few years and did not use any sort of modular patching technique, creating some version of a " Krell patch"—a generative self-playing system in modules—has become something of a rite of passage for both hard- and software modular enthusiasts after being popularized by west-coast modular synthesist Todd Barton around 2012. [14]

"Krelling" is now something of a generative modular 'hello world' exercise in that it shows that both the modular system and its patcher can perform in a generative idiom.

This fanciful ritual—intuitively recreating and extending the imaginary music of a long-dead fictional race—is achieved canonically through the use of a pair of looping amplitude envelopes (preferably ones with end-of-cycle triggers) and a chaotic or random source for pitch information.

As many variations of this patch now exist as there are people who patch it, and for many, "Krell" is as much a synonym for generative or self-playing patches as it is a specific modular configuration.

The longevity and prevalence of the practice is evidence of the generative tendencies of modular systems (it is difficult or impossible to manage to Krell on synthesizer which is not at least semi-modular) and speaks to the joy of patching.

References

[1] Schilling, Melissa A. "Toward a general modular systems theory and its application to interfirm product

modularity." *Academy of management review* 25, no. 2 (2000): 312-334.

[2] Schilling, "Toward a general modular systems theory and its application to interfirm product modularity," 312.

[3] Galanter, Philip. "Generative art theory." *A Companion to Digital Art* 1 (2016): 147-180.

[4] Hinton, Heather M. "Under-specification, composition and emergent properties." In *Proceedings of the 1997 workshop on New security paradigms*, pp. 83-93. 1998.

[5] Sancristoforo, Giorgio. “Gleetchlab Substantia Fantastic Voyage berna3 Quadrivium Bentō.” Giorgio Sancristoforo. Accessed November 8, 2022. https://www.giorgiosancristoforo.net/.

[6] Bode, Harald. "Sound synthesizer creates new musical effects." *Electronics* 34 (1961): 33-37.

[7] Ellis, Thomas O., John F. Heafner, and William L. Sibley. *The GRAIL Project: An experiment in man-machine communications*. RAND CORP SANTA MONICA CA, 1969.

[8] Krajewski, Markus. "The Structure of (Information) Infrastructure: Origins, History, and Theory of the Flow Chart." Seminar für Medienwissenschaft, Universität Basel, 2020.

[9] Puckette, Miller. “The Patcher.” In Proceedings of the 1988 International Music Conference (ICMC '88), pp 420-429, 1988.

[10] Hosick, Eric. “Interface Vision.” Blog. Accessed November 8, 2022. http://blog.interfacevision.com/design/design-visual-progarmming-languages-snapshots/.

[11] Gervás, Pablo. "Propp's Morphology of the Folk Tale as a Grammar for Generation." In *2013 Workshop on Computational Models of Narrative*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[12] Ranke, Kurt. "Kategorienprobleme der Volksprosa." (1967): 4-12.

[13] VCV Rack. “Library/Frozen Wasteland CDC Seriously Slow LFO” almostEric. Accessed November 8, 2022. https://library.vcvrack.com/FrozenWasteland/CDCSeriouslySlowLFO.

[14] Warner, Dan. *Live Wires: A History of Electronic Music*. Reaktion Books, 2017.