UNIVERSITY OF CALIFORNIA,
IRVINE

Sinew & *Ecosystem One*: two works driven by a self-regulating performance system

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Integrated Composition, Improvisation, and Technology

by

Kevin Patrick Anthony

Dissertation Committee:
Professor Mari Kimura, Chair
Assistant Professor Lukas Ligeti
Assistant Professor Theresa J Tanenbaum
Professor Vincent Olivieri

2020

DEDICATION

To

my dear wife and young children

in recognition of their untold sacrifices

and their unceasing love and support

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ADC	Analog-to-digital converter
ADSR	Attack, Decay, Sustain, Release
DAC	Digital-to-analog converter
DSP	Digital signal processing
GLSL	OpenGL Shading Language
LEM	Live electronic music
OS	Operating system
OSC	Open sound control
UCI	University of California, Irvine
xMPL	Experimental Media Performance Lab

VITA

Kevin Patrick Anthony is a music composer, electronicist, vocalist, artist, designer, and a pursuer of creative technologies. Currently completing a PhD at the University of California, Irvine (2020) in Integrated Composition, Improvisation, and Technology, he has a passion for maintaining interdisciplinary environments which promote collaborative creativity. He has received degrees from Brigham Young University Idaho (BMA, 2015) and Brigham Young University (MA in Music Composition, 2017). Kevin is an Assistant Professor of Commercial Music and Composition & Theory at Brigham Young University, Provo, beginning Fall, 2020.

ABSTRACT OF THE THESIS

Musical works and texts historically demonstrate a desire to reconcile composition with improvisation. This text aims to progress the narrative of reconciliation. It outlines the polarized relationship between composition and improvisation. This paper presents tools that aid in their reconciliation. These tools stem from both theoretical and technological contexts. They provide a foundation for a creative work that gives equal weight to its improvisational and compositional components, while fostering a sense of shared purpose. This paper also introduces a new model for interactive performance systems called the *self-regulating performance system*. Two substantial compositions by the author are outlined and presented; *Ecosystem One* and *Sinew.* These works are implementations of the newly presented model.

"Sinew & Ecosystem One: two works driven by a self-regulating performance system" is a dissertation submitted to the University of California, Irvine in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Integrated Composition, Improvisation, and Technology by Kevin Patrick Anthony, with Mari Kimura as committee chair.

Chapter 1.

Introduction

The fundamental natures of composition and improvisation as they relate to this text are inherently in conflict with one another, often at the expense of improvisational freedoms, composer authorship, and a sense of shared purpose. The debate surrounding the relationship between composition and improvisation will likely never "resolve." As long as individual artists maintain their creative freedoms—something I wholeheartedly advocate—there will always exist disparate definitions of both terms. This text offers insight into a theoretical and technological reconciliation of composition and improvisation. To facilitate effective discussion, and to quell circular arguments, I define these terms within this highly specific context. Note that I present these definitions with the understanding that they should remain fluid and part of a living discussion.

This paper presents my research and creative work to address the following research questions:

1. What are the conflicting characteristics of composition and improvisation?
2. What contexts allow for dually maximized composer authorship and improvisational freedom?
3. What theoretical and philosophical concepts are necessary for the reconciliation of composition and improvisation?
4. What technological concepts and tools are necessary for the reconciliation of composition and improvisation?

Musical works and texts historically demonstrate a desire to reconcile composition with improvisation. I aim to progress the narrative of reconciliation through my research and creative work. In this text, I outline the polarized relationship between composition and improvisation.

Following this is an exploration of how this relationship affects a sense of shared purpose between performers, concert attendees, and the composer. The purpose of discussing these issues is to preface the presentation of my solution. In this text, I describe tools that aid in the reconciliation of composition and improvisation. The tools stem from both theoretical and technological contexts. They provide a foundation for a creative work that gives equal weight to its improvisational and compositional components, while fostering a sense of shared purpose.

As a practical implementation of reconciliation, I present a performance system model called the *self-regulating performance system*. The ideas encapsulated by this model are twofold, in that they include both theoretical and technological points. Enactive paradigms and concepts of emergence inform the theoretical points. The theoretical points then inform the model's technological points. I provide details of the model's most prominent characteristics, as well as descriptions of more nuanced attributes that evade categorization.

Following this, I will present two substantial compositions as the creative activity portion of this research. They are *Ecosystem One* and *Sinew.* Both works are practical examples of a self-regulating performance system. These works demonstrate how improvisational freedom and composer authorship can be maximized using certain technologies. Though this is not a tutorial setting, I have presented the works in such a way that will introduce relevant frameworks to those who are interested. I also present specific tools and code-blocks that the reader can use to mimic the processes found in this paper.

I primarily present my creative works from a technological perspective. However, they culminate as answers to this paper's dominant research questions. My first composition I present is *Ecosystem One*. This is the first of my works which I consider to have successfully maximized both improvisational freedom and composer authorship. *Sinew* is the second work that I present.

It is similar to *Ecosystem One*, but is a long-form work. Commentary on the COVID-19 pandemic's impact on *Sinew*—which resulted in a redesign—prepends a discussion of the work.

The starting point for discussing these works will surround their major operational logics. With these logics established and defined, I explain how the works are able to simultaneously maximize improvisational freedom, composer authorship, and a sense of shared purpose. Concepts derived from enactive cognition and emergence provide the analytical lens for how the systems that drive these works qualify as being self-regulating.

Chapter 2.

Research and Literature

2.1. Improvisational Freedom

Existing literature offers various definitions of improvisation, though none appear to be entirely unchallenged or above reproach.[1] While the debate surrounding the disambiguation of improvisation has merit, it does not progress the purposes of this text. For this reason, I focus the discussion on *improvisational freedom*, which is a ubiquitous characteristic of improvisation. Improvisational freedom refers to the notion of performing music without needing to satisfy explicitly stated expectations.

As both a performer and a concert attendee, I have greatly enjoyed the spontaneous music that emerges from the apparent lack of expectation. As a composer, I have enjoyed framing improvisations throughout my works, which I use to explore spontaneous creativity. I add improvisational elements in the hope that certain musical serendipities might occur. However, as I continue to experiment with composition and improvisation in the world of live electronic music, I find myself less satisfied with certain phenomena.

One phenomenon is the extremely varied levels of purposefulness from performance to performance. Improvisers who are familiar with one another's tendencies appear to create meaningful, resonant moments—the opposite also seems to hold true. This, however, is not a constant rule, and is often contradicted. Too many factors appear to be in play to determine

[1] Mazierska, "Improvisation in Electronic Music—The Case of Vienna Electronica"; Bailey, *Improvisation*; Blum, "Representations of Music Making"; Alperson, "A Topography of Improvisation."

which ones contribute to musical evocation of meaning and purpose. How, then, does one ensure a sense of purpose is shared between the performers, the composer, and concert attendees?

Another phenomenon is the lack of authorship I feel when setting improvisation within a composition. Others performing my work while I am not a participant exacerbates this lessened sense of authorship. A seminal definition for free improvisation from Derek Bailey suggests that it is, "established only by the sonic-musical identity of the person or persons playing it."[2] This definition gives no place for compositional authorship outside of the improviser and their past influences.

I take a different approach to understanding improvisation in relation to composition. Rather than label a work—or portion of a work—as improvised or composed, it is useful instead to evaluate its level of improvisational freedom. I evaluate improvisational freedom by considering the magnitude of expectation from external structures. Additionally, there is a notable difference between *perceived* improvisational freedom and *actual* improvisational freedom. Evaluating *actual* improvisational freedom is akin to deeper discussions of agency and unrelated to the purposes of this text.

By evaluating *perceived* improvisational freedom, we enable practical observations that can lead to experimentation. For instance, consider a recital with a single instrumentalist performing spontaneous musical gestures. Many hidden structures provide the expectations driving the improviser's actions. Socially established expectations, like beginning the performance at a designated time, are one example. The performer's past musical experiences are another example. There are, no doubt, infinite hidden influences at play. As the musical performance has no explicit instructions from obvious external sources—e.g., a score—we

[2] Bailey, *Improvisation*.

perceive a high level of improvisational freedom. Here, improvisation is no longer in binary opposition to composition. Instead, it is a component of music with varying levels of presence. By this approach, any structures, cues, prompts, or timelines that originate from a *perceived* source reduce the level of improvisational freedom. In other words, explicit external structures diminish improvisational freedom.

2.1.1. John Zorn's Cobra (1984)

John Zorn's *Cobra* is a "game piece" for an improvising ensemble. The piece requires a "prompter" that signals the beginning of a new event using cardboard cue cards with large letters, numbers, and other symbols. Each performer is given a sheet describing the function of the cards. The sheet also contains descriptions of hand gestures and their functionality. Cobra does not have a physical score. Rather Zorn himself verbally relays the instructions to performing ensembles. During a performance, the ensemble takes charge of the piece by rigorously dictating which symbols they would like the prompter to display.[3]

Cobra explicitly defines several external structures that confine performer choice. Note that confinement of choice within a performance is not qualitatively good or bad—it is, in fact, very desirable in many performance settings. This confinement within *Cobra*'s structure reduces the notion of perceived improvisational freedom. Zorn composed his game pieces, including *Cobra*, in an effort to free the improviser. While discussing the motivation for writing his game pieces, Zorn stated:

> "For a composer to give an improviser a piece of music which said, 'play these melodies - then improvise - then play with this guy…', to me, that was defeating the purpose of what these people had developed, which was a very particular way

[3] Van Der Schyff, "The Free Improvisation Game."

> of relating to their instruments and to each other. And I was interested in those relationships."[4]

Zorn hopes that giving performers a game-like scenario will enable them to more easily express their personal idioms as they are given more power over structure and interaction. His logic initially makes sense, but it does not acknowledge a vital factor: his own desire for authorship. Zorn wants improvisers to feel free and unfettered during a live performance, which is at odds with a composer's paradigm of maintaining an authorial relationship with the composition. This is apparent in the rules that govern *Cobra*. Players are bound by the composer's structures. *Cobra* requires performers to be intensely aware of many discrete events that could occur at any given moment from a wide range of sources. Players can only participate in the piece if they pay attention to a long list of tasks and responsibilities. They must mentally traverse the rules established by the composer in order to take action.

This work has been described by improvisers as either exciting, confusing, stressful, or fun to perform.[5] Frantic attempts to subvert, communicate, and collaborate with fellow performers create an environment of ordered chaos. Each performance is meant to be relatively unstable because the score is mainly a framework for improvisers to follow their own creative inclinations. However, the work remains dominated by obvious external structures. Recall that the magnitude of obvious expectation from external structures indicates the level of perceived improvisational freedom. *Cobra*'s high level of expectation imposed upon performers diminishes the presence of perceived improvisational freedom.

---

[4] Bailey, *Improvisation*.

[5] Van Der Schyff, "The Free Improvisation Game."

2.2. Composer Authorship

Michel Foucault describes literary authorship as the "relationship that holds between an author and a text, the manner in which a text apparently points to this figure who is outside and precedes it."[6] De Benedictis translates this to a musical context as, "the relationship between a composer and a performance."[7] Within my work, as the perceived improvisational freedom increases, my sense of authorship decreases. This leads me to question the nature of authorship, and to explore its apparently inverse relationship with improvisational freedom.

I am not alone in experiencing the dissatisfaction of authorial decline within a work. Luciano Berio expresses similar sentiments when commenting on performances of his work *Sequenza I* (1958) for solo flute. The score employs proportional notation, having no time signature or bar lines. Years after its publication, Berio said he intended the work's flexibility for rhythm and tempo to give the performer:

> "…the freedom – psychological rather than musical – to adapt the piece here and there to his technical stature. But instead, this notation has allowed many players [...] to perpetrate adaptations that were little short of piratical. In fact, I hope to rewrite *Sequenza I* in rhythmic notation: maybe it will be less 'open' and more authoritarian, but at least it will be reliable."[8]

Berio makes clear his desire to explore the "psychological" freedoms associated with improvisation. However, he is ultimately displeased with the results. His republication of the work in 1998 contains strict rhythmic notation. By removing the primary improvisational element of the work, he solidifies his role as author and composer.

---

[6] Foucault, "What Is an Author?"

[7] De Benedictis, "Authorship and Performance Tradition in the Age of Technology."

[8] Berio et al., *Two Interviews*.

The first publication of *Sequenza I* suffers from a clash of desires—one for improvisational freedom and another for composer authorship. As discussed in 2.1 Improvisational Freedom on page 4, improvisational freedom is diminished by explicit external structures.

2.2.1. Agostino Di Scipio's Two Pieces of Listening and Surveillance (2013)

Agostino Di Scipio's work *Two Pieces of Listening and Surveillance* demonstrates how improvisational freedom and composer authorship have trouble existing simultaneously. It is a piece for flute and electronics, and is a slow and dramatic live performance.[9] It begins silently with a single flute in the center of the stage. Attached to it is a contact microphone that is connected to a computer which processes the flute's audio signal. As the piece progresses, complex sonorities audiate from speakers placed slightly offstage. The sonorities gradually come and go, without any visible human performer controlling them and leaving time for large gaps of silence. The music, for the audience member, appears to arise from nothing. In reality, a meticulously designed performance system processes the audio. A self-regulating feedback loop inflates the resonant qualities of the flute and its complex relationship to the room's acoustic qualities.

Di Scipio likens the performance to Lucier's *I am Sitting in a Room* (1970) but with interactive elements introduced. Eventually, a careful performer takes to the stage, gently lifting the flute to chest-level. As designated in the score for *Two Pieces of Listening and Surveillance*, the performer proceeds to hold down the flute's keys in a given order until all of the keys are pressed. If resulting sonorities from the feedback system become unruly or explode in an

[9] Di Scipio, "Dwelling in a Field of Sonic Relationships."

"emergency situation," the performer is instructed to execute "security procedures." This includes violently slamming down the keys or blowing into the flute's mouthpiece, keyholes, and foot joint. As they do so, the system is dampened and the explosive feedback diminishes.

In *Two Pieces of Listening and Surveillance*, the composer's level of authorship decreases mostly when the flutist satisfies the improvisatory instructions for "emergency situations." This is when the piece strays from the composer's specified timeline. As the sonic properties of the work resolve, the timeline is resumed and composer authorship increases. The two elements struggle to exist side by side as they are attempting to occupy the same locale within the work. The performer's improvisational inclinations cannot fully govern performance behavior while satisfying the expectations of an explicit score.

2.3. A Sense of Shared Purpose

The phrase "shared purpose," simply means a common understanding of cause and effect. Alternatively, it is a common understanding of function. These definitions are admittedly reductive, but intentionally so as they are likely the most productive. I seek to experiment with ways that a composer might maximize the chances of cultivating a sense of shared purpose.

Two Pieces of Listening and Surveillance (see 2.2.1) demonstrates how transparent functionality might be embedded within a score. The work encourages a common understanding of cause and effect when the performer interacts with the system. As they follow the score, occasionally improvising, their actions reveal semi-predictable properties of the system. The performer, composer, and concert attendee all glimpse portions of the system's functionality and intuitively learn its nature. This common understanding fosters a sense of shared purpose through highly transparent functionality. The slow revealing of cause and effect is a core component of the score's timeline fully intended by the composer.

Cause and effect throughout a performance of *Cobra* (see 2.1.1) is relatively opaque and obscured. Its functionality appears to be more difficult to locate for both performer and attendee. The lack of apparent cause and effect add to the whimsical, chaotic nature of the work. While this does not equate to a lack of shared purpose, it does not encourage it. Perhaps the work encourages a sense of common understanding by way of this general chaos. However, the piece does not seem to consider the revealing of functionality as a structural element. Instead, the persistent ambiguity of cause and effect serves as the motivator for performer behavior.

2.3.1. George Lewis' Voyager

George Lewis' *Voyager* lays the groundwork for experimentation with embedding composer-intended human response, which is a useful mechanism for fostering a sense of shared purpose. *Voyager* is a "non-hierarchical, interactive musical environment that favors improvisation. When performing with *Voyager*, improvisers engage in dialogue with a computer-driven, interactive 'virtual improvising orchestra.'"[10] Lewis' system invites the performer and listener to discover purpose and function within its performances.

The research behind *Voyager* reflects on human tendencies and behaviors. This is a fascinating topic to focus on with how juxtaposed it is to an autonomously performing digital system. Music produced by *Voyager* exists because of the cyclic exchange of information between the system and the performer. There appears to be a stochastic-dominance throughout the system. Lewis did not build the machine to be predictable. He intentionally hides action and consequence. He has intentionally built a cold system that forces performers and concert attendees to look inward for purpose and meaning.

[10] Lewis, "Too Many Notes."

Voyager's natural and organic performance behaviors encourage a sense of familiarity and purpose. This is especially apparent as performers and concert attendees frequently anthropomorphize the interactive system. In a panel discussion following a 2012 performance with Vijay Iyer and George Lewis, Iyer talks of "reading [*Voyager*'s] mood." [11] A commentator discusses the system's behavior in terms of "personality," saying that it is timid, inspired, or "in a funk." When discussing how the performers adapted to a particularly timid portion of the system's performance, Lewis said,

> "It's hard to explain how you can feel that a computer is in a particular mood, but that's just the way it is. You get to know something for a while and you can just tell the thing's not going to go a certain way."[12]

Voyager establishes a precedent for composers to embed intentional human responses within performance systems. It demonstrates how the use of archetypical gesture generates a sense of purpose and meaning within a performance system. This concept is useful when experimenting with designing desirable human reactions to a work, particularly when attempting to embed within it a sense of shared purpose.

2.4. Repulsion

Having contextualized improvisational freedom, composer authorship, and a sense of shared purpose, we can better summarize which of their qualities are repellent from one another. Improvisational freedom diminishes composer authorship and does not inherently generate shared purpose. Composer authorship in the form of explicit external structures restricts improvisational freedoms and can bolster a sense of shared purpose.

---

[11] *George Lewis and Vijay Iyer in Concert*.

[12] *George Lewis and Vijay Iyer in Concert*.

Two Pieces of Listening and Surveillance encourages a sense of shared purpose through the interactions of the performer. Di Scipio carefully displays cause and effect, which fosters a common understanding between the performer, composer, and the concert attendees. This clear communication of meaning is the result of strong external structures, at the expense of more improvisational freedoms. Here, mechanisms generate a strong sense of purpose and repel freedom—the mechanisms being explicitly stated external structures.

Cobra's explicit structure creates strict expectations for the performer to satisfy. This requires the performer to undergo anticipatory behavior, which is at odds with the ideal scenario of reactive performance. Anticipation comes from the need to satisfy an "established authority."[13] Thus, explicit external structure diminishes improvisational freedom.

Reconciling improvisational freedom, composer authorship, and a sense of shared purpose is a difficult task. The scenarios previously explored in this text illustrate a precedent for maximizing the presence of all three elements. Common practice currently offers only fragmented solutions, as will be explored in sections 2.5 and 2.6. I have found through much experimentation that successful reconciliation of all three elements requires alternative methods, both theoretical and technological.

2.5. Theoretical Reconciliation of Improvisation and Composition

Composition and improvisation might be reconciled theoretically through the philosophy of enactive cognition, as well as concepts involving emergence. These ideas enable a valid relocation of composer authorship that satisfies the nature of both improvisation and a sense of

[13] Spolin, *Improvisation for the Theater*.

shared purpose. They also establish a foundation for certain technologies to aid in the maximizing of improvisational freedom, composer authorship, and a sense of shared purpose.

2.5.1. Enactive Cognition

The ideas surrounding enactive cognition enable a composer to design environmentally integrated scores that are compatible with improvisational freedom. In other words, the theory helps a composer embed a subliminal score within a performance environment. The study of enactive cognition suggests that cognitive processes are deeply entangled in physical action and interaction with the environment:

> "Varela, Thompson, and Rosch introduced the concept of enaction to present and develop a framework that places strong emphasis on the idea that the experienced world is portrayed and determined by mutual interactions between the physiology of the organism, its sensorimotor circuit and the environment."[14]

> "The basic idea of the enactive approach is that the living body is a self-producing and self-maintaining system that enacts or brings forth relevance, and that cognitive processes belong to the relational domain of the living body coupled to its environment."[15]

On the basis of enactive theory, improvisations are always informed by structures found within the performance environment. These environmental factors do not diminish the perceived improvisational freedom. If a composer were to inject their structures into the performance environment in a natural and effective manner, it could theoretically enable the maximizing of improvisational freedom and composer authorship simultaneously.

---

[14] Wilson and Foglia, "Embodied Cognition."

[15] Thompson, "Introduction."

Enactive theory stands in contrast to the Cartesian paradigm, which divides the world into discrete elements, tasks, and events. This quantized view of experience is prevalent in my musical upbringing and education. For instance, a musical score is clearly distinguishable as an external object with specified properties and tasks. Each task is represented within many distinguishable levels of detail. Sections contain phrases, and phrases contain notes. Each of these levels contain properties. Notes have locations, durations, and expressions. Phrases have momentum and dynamics. Sections contain texture and narrative. This framework of individuated properties and tasks is useful and functional, but encourages the dichotomy between improvisation and composition. It reinforces composition as an explicit external structure that generates expectations to be satisfied by a performer.

My initial endeavors with improvisation and composition involved the integration of improvisational prompts throughout a traditional written score. I often used timed sections of improvisation mingled with notated passages of music. Other attempts involved interactive systems that allowed the improviser to explore whatever musical possibilities the system afforded. Despite not having the necessary nomenclature at the time, I perceived the conflict between improvisational freedom and composer authorship. As each attempt was created under Cartesian paradigms, their approaches to composition resulted in externally represented objects that interfered with improvisation. This usually ended with either composition or improvisation—and far too often, technology—unsatisfyingly dominating the work.

Along with compositional, improvisational, and technological elements being regularly out of balance, the general relevance of blending these elements seems to ebb and flow inconsistently. Enactive theory suggests that relevance is derived from the relationship between action and the environment:

> "In formulating the enactive approach, we drew on multiple sources: the theory of living organisms as self-producing or "autopoietic" systems that bring forth their own cognitive domains; newly emerging work on embodied cognition (how sensorimotor interactions with the world shape cognition); Merleau-Ponty's phenomenology of the lived body; and the Buddhist philosophical idea of dependent origination, and specifically that cognition and the experienced world co-arise in mutual dependence."[16]

Applying Thompson's notion of mutual dependence to an improviser and their environment reveals new possibilities for composition. It ultimately allows for composed elements of a work to be integrated into the performance environment. As discussed in 2.1, improvisational freedom is always informed by social and environmental factors. When the composition is an element of the environment, it becomes less visible as an external structure. With perceived external structures minimized, performers can then maximize their sense of improvisational freedom.

To illustrate how enactive cognition can inform an alternative paradigm for composition, I will describe a hypothetical scenario. Imagine a violinist improvising while alone in a small, secluded room. There are countless factors influencing their musical choices. In this example, we will focus on a single, immediate factor; the acoustics of the room. On some level, the room's acoustics are engaging with the performer's sensori-motor system and causing perturbations throughout their improvisational structures. Let the room now transform into a large concert hall with highly reverberant resonant properties. This space will promote a different quality of engagement between the violinist and the environment. The emergent music that arises has been transformed by the environment. Now imagine the stage upon which the performer is standing begins to shift and contract in reaction to the violin. The performer's level of interaction with the space is heightened. As observers, we can hear and see how the motion of the floor influences

[16] Thompson.

the ongoing improvisations. Such control over the venue would demand a new paradigm for composition, one where the composer must consider the possibilities surrounding the relationship between performer and environment. Similarly, the presence of modern technologies and their widespread availability calls for the exploration of new compositional paradigms. Through certain technologies, a composer is effectively able to alter the rules of a performance environment, which can in turn affect the behaviors of performers and improvisers.

An enactive approach to composition, improvisation, and technology changes performance from the translation of ordered symbols to engaged interaction with the environment. Additionally, the improviser is not seen as a single, independent object. Instead, they are viewed as being tightly coupled with the environment.

> "...a cognitive being's world is not a pre-specified, external realm, represented internally by its brain, but is rather a relational domain enacted or brought forth by that being in and through its mode of coupling with the environment."[17]

Composition under this approach, then, is not the act of imposing external, pre-specified tasks. It is the act of thoughtfully organizing environmental interactions. Consequently, when designing a performance environment, one would not specify events. Instead, designed interactions are the primary compositional device. Of course, there is only so much impact a composer can have on a natural performance environment. This is where technology plays a vital role. It enables richly designed behaviors that can be embedded within the environment's properties (see 2.6 Technological Reconciliation of Improvisation and Composition).

[17] Thompson.

2.5.2. Emergence

An understanding of emergence is vital for maximizing composer authorship and improvisational freedom. As composed elements of a work are relocated to the performance environment, they become less obstructive toward a performer's sense of improvisational freedom. However, in order for the composer to maintain authorship of the work, they must somehow communicate desired performance behaviors. Here, it becomes necessary for the composer to mimic the emergent process by which we interact with the environment. Emergent phenomena may be contained and constructed by a composer to facilitate desired performance interactions, thus maintaining unobtrusive authorship.

Under the model of an environmental score, emergence is the primary mechanism by which a composer communicates with their performer. It is no doubt a broad term with many acceptable interpretations. First, I will explain my own understanding of emergence. Following this, I will compare two common perspectives of emergence—one from Timothy O'Connor, and another from Mark A. Bedau. This discussion will include explanations of how emergence is a powerful tool for reconciling improvisational freedom, composer authorship, and a sense of shared purpose.

My interpretation of emergence is as follows: *a phenomenon is emergent while its initial existence is the unanticipated result of a given process*. This interpretation highlights the circumstances of the human vantage point, taking into account the progression of time. I will reconstruct the interpretation, unpacking it as I do so. In its most basic form, it reads, "a phenomenon is emergent when it is the result of a process." This is inadequate. Emergence seeks to define more puzzling phenomena. Adding more to the interpretation, we have, "a phenomenon is emergent when it is the *unpredictable* result of a process." This is closer to capturing emergence. If a process's components are well understood without facilitating predictable

results, then the results are often labelled emergent. However, once the results of a process have been observed over time, an observer will not consider them to be unpredictable. To address this possibility, it reads, "a thing is emergent when its *initial existence* is the unpredictable result of a *given* process." At this stage of the reconstruction, a glaring hypothetical question must be considered. What if you could wholly perceive and comprehend every influential factor of a given process as an omniscient observer? In this scenario, would the observer be able to identify causal relationships for every phenomena, or is there true emergence? I instinctively think that a theoretical omniscient observer would perceive a complete lineage of cause and effect. The implication is that a hypothetical objective reality does not contain emergence. Recall that my interpretation considers the human vantage point. We are, fortunately, not omniscient entities and, as such, there are processes that will likely always prove to baffle us. Considering this, I alter my interpretation, "a phenomenon is emergent while its initial existence is the *unanticipated* result of a given process."

A common understanding of emergence is presented by Timothy O'Connor. His explanation of emergent phenomena is aligned with what may be considered to be the dominant use of the term. According to O'Connor, a property is emergent from an object if, (1) the property supervenes on properties of the object's individual components, (2) the property is not an integral or structural component of the object, and (3) the property has a direct determinative influence on the object's components.[18]

O'Connor's definition suggests that an emergent property's influence and behavior is irreducible and cannot be directly traced to low-level components. Essentially, the definition claims that an object's properties can exist spontaneously without any physical or scientific

[18] O'Connor, "Emergent Properties."

explanation. This view of emergence inhibits its use in empirical fields of study where a pragmatic sense of the term could prove useful. However, in philosophical and artistic fields, it is appropriate to muse over inexplicable phenomena. As such, emergence as described by O'Connor is useful and legitimate to philosophers and creators who regularly deal with perception, consciousness, and the human experience.

A second understanding of emergence is presented by Mark A. Bedau. He makes the distinction between strong and weak emergence.[19] Strong emergence is the mode described by O'Connor. He concisely defines weak emergence in a technical manner. David J. Chalmers paraphrases Bedau's concept in a more accessible manner:

> "We can say that a high-level phenomenon is weakly emergent with respect to a low-level domain when the high-level phenomenon arises from the low-level domain, but truths concerning that phenomenon are unexpected given the principles governing the low-level domain."[20]

There is only a slight difference between Chalmers' paraphrasing of weak emergence compared to strong emergence. He uses "unexpected" for weak emergence, and "not deducible" for strong emergence. Weak emergence acknowledges a causal relationship between low-level components and high-level structures. If an unexpected phenomenon is only knowable after carrying out the process that produces it, let us say it is weakly emergent.

Strong emergence describes a process that is far too transient and inaccessible to attempt to imitate—that is, in a way that merits pragmatic discussion. Alternatively, Bedau's weak emergence provides us with a template for how to design such processes. By further delving into

[19] Bedau, "Weak Emergence."

[20] Chalmers, "Strong and Weak Emergence."

the nature of emergence—particularly weak emergence—we can see how it is possible for emergent phenomena to be cultivated and designed according to the desires of a composer.[21]

2.5.2.1. John Conway's Game of Life (1970)

John Conway's Game of Life illustrates how emergent systems can be designed and cultivated to achieve a known range of results. The Game of Life is a cellular automaton researched and discovered by John Conway in 1970.[22] Cellular automata involve sets of rules dictating the state changes of individual "machines" aligned in a grid, each with the ability to influence neighboring machines during subsequent state changes. The format was developed during the 1950s by John von Neumann and Stanislaw Ulam in order to study and define self-reproducing machines.[23] Since then, many efforts have been made to define a ruleset that could exhibit complex, sustainable patterns. If successful, it would demonstrate how such complex patterns are generated from individual, separate, low-level components—among many other things. Conway's Game of Life, is one such ruleset that generates remarkably complex behavior from four simple rules: (1) any live cell with fewer than two live neighbors dies, as if by under population, (2) any live cell with two or three live neighbors lives on to the next generation, (3) any live cell with more than three live neighbors dies, as if by overpopulation, and (4) any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

---

[21] While this text is specific to composers, the cultivation of emergence can be said to apply to nearly every design process.

[22] Gardner, "Mathematical Games."

[23] Schiff, *Cellular Automata A Discrete View of the World*. 40.

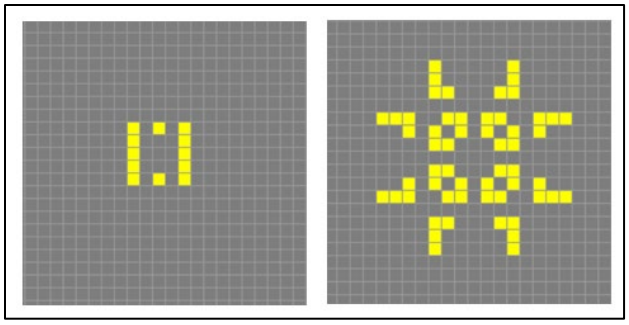

Figure 1 John Conway's Game of Life, the Exploder. Step 0 (left) and step 45 (right).

Conway's Game of Life is an example of a system that produces weakly emergent phenomena. The transparency of the system's algorithm means that the process for any given result can be entirely defined. It is within human capacity to manually calculate each time-step—as was originally done by mathematicians studying cellular automata. Doing so creates a record of causal relationships between starting parameters and any emergent, complex structures. This process is depicted in Figure 1. Two frames are shown. The left frame shows the starting conditions. The right frame shows the cellular automaton's state after forty-five steps. Since step forty-five is not anticipated and only discoverable through the carrying out of a process, its existence is deemed to be weakly emergent.

From this example, we begin to see how emergent phenomena might be designed. Conway's ruleset generates complex structures and behaviors that are not directly defined by the rules themselves. Instead, they emerge as outcomes of the process. The simulation must be carried out for them to exist. Here we can surmise that to design the outcome of the process, one could slowly alter its starting parameters and general structure. Over time, the nuanced behavior of the system can become familiar to the designer—allowing for greater design potential.

While designing emergent systems, it is important to consider scope and complexity. It is one thing to create a process that produces complex results. It is another thing to create a *simple* process that produces complex results. Within my work, I find that the most interesting results come from mapping simple mechanisms. Conway's ruleset efficiently demonstrates this idea. I do not believe that the complexity of a process reduces its ability to produce interesting emergent art. Instead, the complexity of a process should be evaluated by its creator.[24] Simple processes, as demonstrated by many cellular automata, can easily bring about complex system behaviors; complex processes can also bring about simple results. As a composer, I value simple processes over complex processes. I aim to maintain a strong understanding of my performance systems. If a system cannot be abstracted into a simple process then my understanding of its potential wanes. Some composers may find this desirable, seeking to harness complex processes as a means of experimentation and exploration.[25] My work will certainly produce complex systems, but attempting to maintain simple designs will allow me to effectively cultivate desirable outcomes.

2.5.2.2. Alvin Lucier's *I am Sitting in a Room* (1969)

The circular nature of Alvin Lucier's *I am Sitting in a Room* is a clean example of how to generate emergent material. The process Lucier uses is simple. Audio feedback is used to explore the natural resonances of a room. The audio from a spoken text is recorded, then played through

[24] Ulman, "Some Thoughts on the New Complexity."

[25] Fox, *New Complexity*.

loudspeakers.[26] The playback of this recording is also recorded, then played/recorded again. This process repeats until the audio gives way to the natural resonances of the room.[27]

Like Conway's Game of Life, I am Sitting in a Room is a seminal example of complexity emerging from a simple circular relationship between an object and its environment. Lucier's recording begins as a simple string of words that appears to be largely unaware of the sonic properties of the room. Over time, signal processing slowly transforms the words into a soundscape of subtle complexities shaped by the room's sonic qualities.

I am Sitting in a Room demonstrates how emergent phenomena are a result of circular relationships. It contains a straightforward process that magnifies the relationship between two subjects: the room's acoustic properties and the performer's voice. This formula can be followed to design emergent performance systems. For example, when designing a performance system, I choose two subjects and map them together in a simple relationship that results in a pattern of circular causality. This describes a situation where a subsystem's output is repurposed as its own input. An example of this sort of circular relationship if found in section 3.2.4. When I am satisfied with the results, I add another subject, tethering them to either one or both of the established subjects. Depending on the interest level of the outcomes, I may discard or continue using the new subject. This is the process of cultivating emergence—which is, in and of itself, an emergent process.

---

[26] The text for Lucier's *I am Sitting in a Room* (1969) is: "I am sitting in a room different from the one you are in now. I am recording the sound of my speaking voice and I am going to play it back into the room again and again until the resonant frequencies of the room reinforce themselves so that any semblance of my speech, with perhaps the exception of rhythm, is destroyed. What you will hear, then, are the natural resonant frequencies of the room articulated by speech. I regard this activity not so much as a demonstration of a physical fact, but more as a way to smooth out any irregularities my speech might have."

[27] Strickland, *Minimalism*. 281.

2.5.2.3. Cornelius Cardew's The Great Learning, Paragraph 7 (1968)

Cornelius Cardew's *The Great Learning, Paragraph 7* helps clarify the concept of "scope" when dealing with strong and weak emergence. Its score consists of musical phrases and written directions. Each member of the performing vocal ensemble chooses a starting note and sings a musical fragment any number of times before continuing to the next phrase. When a singer begins the next phrase, their pitch must be a note presently sung by a neighboring member of the ensemble. Though the process for this improvisation is simple and straightforward, the resulting music is complex and coherent. The initial sound mass that is built predominantly on noise-like sonorities gives way to pleasant harmonies and coherent structures.

What this work brings to light is the infinitely variable scope that can be used to analyze emergent behavior. For instance, if we delve deep enough into concepts of strong emergence, we are confronted with the need to qualify any given phenomenon as emergent. Consequently, everything we perceive can arguably be labeled as being emergent. Emergence as a term then runs the risk of describing everything, which in turn describes nothing. To maintain its usefulness, the term must carry with it a sense of limits.

Consider the harmonic qualities of a chord as a strongly emergent phenomenon. Harmony emerges as the listener's perception of the relationship between oscillating sound waves. However, as pointed out by McCormack et al., scientific observation has not yet discovered the exact process that results in the perception of harmony.[28] This basic mode of strong emergence is distinguishable from the weak emergence that comes from directed, rule-based circumstances. Harmony from a generic perspective is more closely related to natural synergetic processes that drive day-to-day perception. Harmony from a compositional

[28] McCormack et al., *Generative Algorithms for Making Music*.

perspective, however, can be seen to exist in a given form as a result of a known process carried out by the composer. For the purposes of this paper, emergence will focus on the latter perspective. For instance, the harmonies constructed by performers during *The Great Learning, Paragraph 7* emerge—weakly—from a known process.

The strongly emergent qualities of a chord are inaccessible from a design point of view. On the other hand, weak emergence is highly accessible. Focusing on weakly emergent processes helps to restrict the system's design to a manageable level of detail. Weak emergence inherently provides limits and boundaries. In the context of composition and performance systems, it is useful to favor weak emergence. This is because the limits that weak emergence provides allows for a manageable frame of creative reference. For instance, suppose that a particular state of my performance system invokes a general sense of discovery and arrival. I can comprehend that these macrodynamics—discovery and arrival—are the unanticipated but deducible results of the system's microdynamics. Understanding this enables an attempt to alter the microdynamics in order to further cultivate the desired outcome.

Additionally, Cardew's *The Great Learning, Paragraph 7* demonstrates how the phenomena that emerge from a process somehow maintain a level of recognizability. Brian Eno, who took part in the recording of the work, observed:

> "A cursory examination of the score will probably create the impression that the piece would differ radically from one performance to another, because the score appears to supply very few precise (that is, quantifiable) constraints on the nature of each performer's behaviour, and because the performers themselves (being of variable ability) are not 'reliable' in the sense that a group of trained musicians might be. The fact that this does not happen is of considerable interest, because it suggests that somehow a set of controls that are not stipulated in the score arise in

> performance and that these 'automatic' controls are the real determinants of the nature of the piece."[29]

It is baffling and fascinating that emergent material from compositional processes can somehow become a discrete entity. It is to the composer's advantage that this is the case, as it supports the notion that variability does not equate to a lack of identity. This means that a process can be as recognizable and identifiable as a written score, and is consequently a valid locale for composer authorship.

2.5.2.4. Terry Riley's *In C* (1964)

Terry Riley's *In C* (1964) is another example of how a performance algorithm can maintain its recognizability despite each performance being unique. The work consists of fifty-three musically notated, independent phrases. Performers play through each phrase, skipping phrases when desired, and repeating them each an indeterminate amount of times. The work is aleatoric and does not dictate when a performer should begin or end playing. The score does, however, encourage performers to maintain a two or three phrase distance from one another.

All recognizable forms of music that this piece generates (as well as many performance behaviors) are not present in the score. They exist only when a performance is carried out. This is due to the score's format. Riley's score describes a process. It does not dictate moment to moment events. As a simple process is followed by the ensemble, then novel and compelling musical structures emerge.

Additionally, *In C* illustrates the importance of iteration in relation to the composition process. When crafting and cultivating emergent phenomena, it becomes vital for the creator to iterate through designs and configurations. This is essentially the act of slowly and methodically

---

[29] Eno, *A Year with Swollen Appendices*. 336.

implanting one's artistic identity into the system. In doing this, a composer deliberately shapes the emergent properties of performance systems and injects their inclinations—and consequently their authorship—into the work. The original score for *In C* did not include a metric pulse. It was only added after his contemporary Steve Reich suggested it to Riley. The pulse is a small addition to the score, but it causes a major difference in the emergent structures of the piece. This is a basic example of a composer cultivating desired emergence through iteration.

2.6. Technological Reconciliation of Improvisation and Composition

The previous section discusses how composer authorship and improvisational freedom might be reconciled theoretically, through the philosophy of enactive cognition and the concept of emergence. These ideas establish the foundation for exploring certain technological concepts that will aid in the maximizing of improvisational freedom, composer authorship, and a sense of shared purpose. The concepts discussed in the following sections include continuous process, audiovisual elements, and operational logics.

2.6.1. Continuous Process

Section **2.5.1 Enactive Cognition** establishes that embedding composer authorship within the performance environment can allow for maximized improvisational freedom. Communicating the composer's expectations through the performance environment requires emulating real-world interaction, otherwise it becomes distinguishable as an external structure. Recall that our perceived world is enacted through a "tight coupling" with the environment (see page 17). Thus, high-fidelity emulation of real-world interactions can only occur by simulating continuous processes.

A continuous process is one that implies infinitely distinguishable variables at any given point in time. These variables can also, "theoretically take infinitely many values in a given range."[30] The notion of continuous phenomena can be analyzed and researched at many different levels of detail. Some have even attempted to define the characteristics of continuous processes at the Newtonian level.[31] However, within the context of commonly available hardware, a continuous process is evaluated by its programmed architecture. In other words, a process is deemed continuous if its programming logic is uninterrupted and imperceptibly fast.[32]

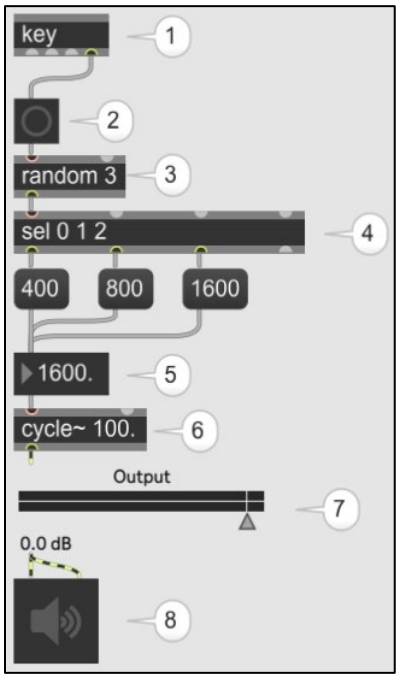

Figure 2 Screenshot of a Max patch demonstrating a discrete process.

[30] Mayya, Monteiro, and Ganapathy, "Types of Biological Variables."

[31] Parr and Friston, "The Discrete and Continuous Brain."

[32] Conrad, "Discrete and Continuous Processes in Computers and Brains."

Table 1 Description of the discrete process demonstrated in Figure 2.

Step Number	Process Description	Discrete or Continuous
1	**Key** reports any keypress event.	Discrete
2	**Button** reports a "bang" message when any data is received from **Key.**	Discrete
3	**Random** generates a number between 0-1 when triggered by **Button**.	Discrete
4	**Sel** sends a "bang" message through the outlet that corresponds with the number received from **Random**.	Discrete
5	**Number** (float) displays the value received from **Message** boxes connected to **Sel** outlets.	Discrete
6	**Cycle** produces a cosine signal with a frequency of the value received by **Number**.	Continuous
7	**Live.gain** monitors the signal from **Cycle.**	Continuous
8	**EzDac** sends the signal to the DAC.	Continuous

Figure 2 demonstrates the notion of a discrete process within a simple Max patch. Each step of the process is described in Table 1. Along with descriptions, each step is categorized as either being discrete or continuous. The patch produces a sine tone at one of three designated pitches. The pitches are randomly decided following human-machine interaction. When a user presses any button on their keyboard, the patch's process is triggered, resulting in a potential shift in the tone's frequency.

Triggers, buttons, and gates are all examples of discrete events. A process is categorized as continuous or discrete based on whether its available interactions incorporate these elements. The overall behavior in Figure 2 is discrete in nature as interactions with the patch are inherently distinguishable events in time and triggered through individual keypresses.

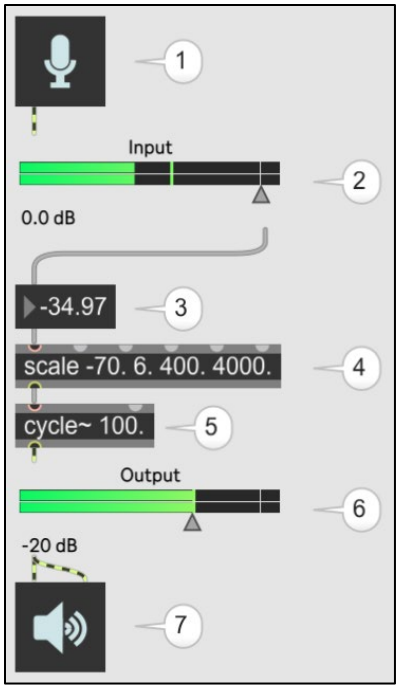

Figure 3 Screenshot of a Max patch demonstrating a continuous process.

Table 2 Description of the continuous process demonstrated in Figure 3.

Step Number	Process Description	Discrete or Continuous
1	**EzAdc** sends a signal from the microphone using the ADC.	Continuous
2	**Live.gain** monitors the signal from **EzDac**.	Continuous
3	**Number** (float) displays the amplitude value received from **Live.gain**'s final outlet.	Continuous
4	**Scale** interpolates the incoming amplitude value between 400 and 4000 from an original range of -70 and 6.	Continuous
5	**Cycle** produces a cosine signal with a frequency of the value received by **Scale**.	Continuous
6	**Live.gain** monitors the signal from **Cycle.**	Continuous
7	**Ezdac** sends the signal to the DAC.	Continuous

Figure 3 demonstrates the notion of a continuous process. Like Figure 2, this patch produces a sine tone. The frequency of the tone is controlled by the amplitude of an incoming audio signal. This control signal is an uninterrupted stream of data that is imperceptibly fast. On certain levels of detail, an audio stream is not continuous. However, the continuousness of a process within this context is evaluated from the perspective of human perception. The speed of

the audio signal, commonly 48 kHz, is imperceptibly fast. As such, we perceive it to be continuous.

Continuous processes are central to the notion of making the composer's expectations unobtrusive. They are what enable embedding a composer's authorship and expectation within the performance environment. The following sections—section **2.6.1.1** and **2.6.1.2**—explore two examples that explore discrete and continuous events, composer authorship, and improvisational freedom.

2.6.1.1. John Cage's *Reunion* (1968)

John Cage's *Reunion* shows how *time* plays a vital role in the overall perception of emergent music—effectively illustrating the difference between continuous and discrete time steps in an algorithmic environment. The work features two performers opposite a chessboard. Each square on the chessboard is configured to trigger audio playback—using photoresistors—when covered by a game piece. Cage recruited Lowell Cross to construct the chessboard, only making two requests about its functionality. He first requested that Cross include contact microphones within the board so that the player's movements could be amplified. His second request was that when played, the board would select and spatialize musical sounds.[33] Cage's self-proclaimed aesthetic was driven by a strong desire to remove himself from the music. Naturally, he was attracted to musical frameworks with emergent properties. Cage uses the emergent nature of a chess game to remove himself from the work, allowing a distant process to dictate the music's structures.

[33] Cross, "*Reunion*."

A game of chess is not unlike a cellular automaton, where each game piece traversing the grid must follow a certain class of rules. *Reunion* demonstrates a similar framework to Conway's Game of Life—discussed in section 2.5.2.1. Both projects demonstrate how a base set of rules in a system can generate results that are not inferable from low-level components. *Reunion* is a prominent model of weak emergence in music. Chess progresses in discrete steps, turn by turn, and the triggering of sonic behaviors can easily be traced. These behaviors are not, however, predicted or expected—qualifying them as weak emergence.

I believe the outcome that Cage ultimately prefers is the undisturbed outcome of the process he invents—whether or not it is the outcome he inwardly hoped for. According to Cross:

> "The circumstances attending *Reunion* permitted no correlation between Cage's elegantly proscribed application of his system of indeterminacy and his underlying hope that elegant games of chess could bring forth elegant musical structures. The games clearly were not elegant, and I, for one, held no expectation that they could have brought forth elegant, or even interesting, musical structures."[34]

Reunion certainly produced emergent musical structures, but Cross' account confirms that they were not the desired outcome. Cage's ability to remain dedicated to a given process is admirable. However, I believe the elegance he hoped for may have come through a more continuous system. In *Reunion,* events are easily distinguished from one another as they are directly coupled to the time-stepped nature of chess. Here we can see how the length of each time-step dominates the perception of the overall process. The musical phenomena appear less emergent as microdynamics become more apparent. From this we can learn that the length of each time-step greatly alters the overall aesthetic and perception of an algorithmic process.

[34] Cross.

2.6.1.2. Tim Blackwell's *Swarm Music* (2003)

The continuous nature of Tim Blackwell's *Swarm Music* enables its users' improvisational freedoms while maintaining the designer's intended interactions. His systems form a unique mode of emergent performance. They are interactive environments where a performer's musical decisions influence digital music generation. They use principles of self-organization in the form of flocking algorithms to generate musical audio.[35] Blackwell's hope is to connect the structures that emerge from improvising ensembles to the self-organization of swarming behaviors found in nature. Simulating a natural swarm is typically implemented in two or three dimensions. Rather than constraining the swarm mechanisms to three properties, Blackwell's systems interpret complex sound events into multidimensional abstractions. These interpretations are mapped to granular synthesizers.

This work demonstrates that a digital computer can appropriately produce continuous interactions suitable for creative works that approximate natural emergence. Emergent behaviors arise despite the deterministic nature of a digital framework. The work avoids being anchored by digital computation and provides a fluid interface with the performer. The underlying digital processes do not interfere with human expression; rather, they enable it. Here we see a continuous exchange of information between performer and performance system, validating the digital world as an appropriate shell for unobtrusive composer authorship.

Nature is abundant with systems that adapt, evolve, and self-organize. Generative art that takes advantage of modern computing powers can imitate (not duplicate) these natural systems. However, the versatility of general purpose computing allows artists to bend natural rules. Just as

[35] Blackwell and Bentley, "Improvised Music with Swarms."

Blackwell repurposed swarming algorithms to drive granular synthesis, an artist can build novel functionality—and consequently their authorship—into an interactive system.

2.6.2. The Audiovisual Element

Relocating compositional elements to the performance environment requires novel ways of communicating the environment's current state. It is effective to use both audio and visual elements for communication when simulating natural environmental interaction. Such communication must maintain the concepts of enactive cognition and emergence as a precedent. Information should not be portrayed in a way that demands expected behavior of the performer. This would reveal authorial intention, establishing it as an external expectation. Instead, it should be portrayed as high-level information that is tightly coupled to the relationship between the behaviors of the performer and the system. This method of communication maximizes both improvisational freedom and composer authorship.

During a live performance with interactive systems, it is common for system states to be communicated in a data-centric format. Information is portrayed in numeric, graphical, or other formats of low-level data features. This method is useful for typical interactive circumstances. However, it communicates information that is far too literal and data-centric to satisfy the conditions for unobtrusive composition. It ultimately reveals the composer's expectations, forcing their authorship to interfere with the improviser's sense of freedom.

An example of a data-centric display is shown in Figure 4. This Max patch conveys real-time information of the current orientation of a MUGIC™ sensor which is shown in Figure 5.[36] It converts multiple streams of data into a single point on the display. This is a form of low-level

[36] Kimura, "MUGIC Sensor."

feature extraction. It is data-centric because it portrays information that is secondary to the composition.

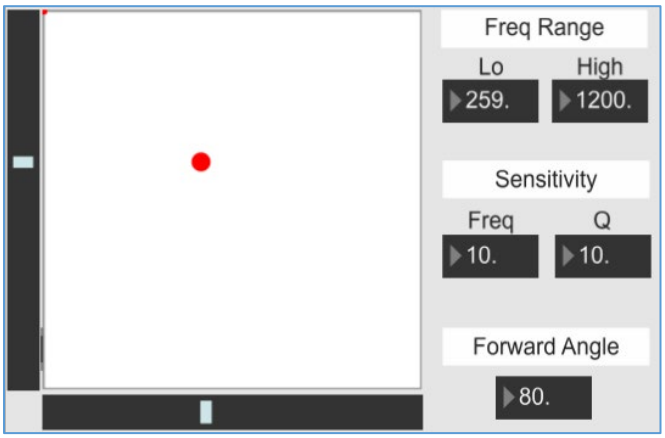

Figure 4 Screenshot of a Max patch used in Ecosystem One (2019) exhibiting data-centric imagery.

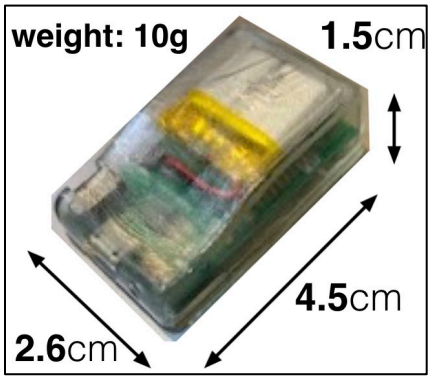

Figure 5 The MUGIC™ sensor. Project led by Mari Kimura.

My work titled *Recourse* (2019) demonstrates communication through high-level abstractions, as shown in Figure 6.[37] The virtual performance environment reacts to the level of musical intensity. As intensity increases, floating objects appear on screen. As intensity decreases, the character portrayed appears sickly and will fade out of view. Time reveals that the

[37] Anthony, "Recourse."

floating objects are adversarial in nature, apparently "attacking" the character portrayed at the bottom of the screen. As they attack, they also perform intrusive musical gestures.

Figure 6 Performance of Recourse (2019).

I embed compositional intent within these performance behaviors, allowing for a maximized sense of improvisational freedom while maintaining composer authorship. Various degrees of intensity should drive the performance, as the performer reacts to the audiovisual elements. The performer will theoretically work to avoid causing the character pain, which occurs variably between too little or too much intensity. I never explicitly inform the performer of this. Instead it emerges naturally through the portrayal of high-level abstractions. As the performance progresses, the performer gains an awareness of how certain musical behaviors have a causal relationship with the imagery. The improviser does not experience a need to satisfy explicit external structures, because I deeply embed them within the performance system.

2.6.2.1. Noisefold's *Emanations*

Transdisciplinary group Noisefold (Cory Metcalf and David Stout) merges interactive visuals, animation, and synthesized sound through the lens of cybernetics and mathematics.[38]

[38] Stout, "Performance."

Their project *Emanations* is an example of how all of these elements lead to the practice of data dramatization. Their compelling geometric visuals are based upon data-centric models of artificial life systems. Metcalf and Stout engage in a sort of game where they interact with bio-mimetic structures in complex environments. They examine the nature of control while trying to tame explosive populations or revive dwindling organisms.

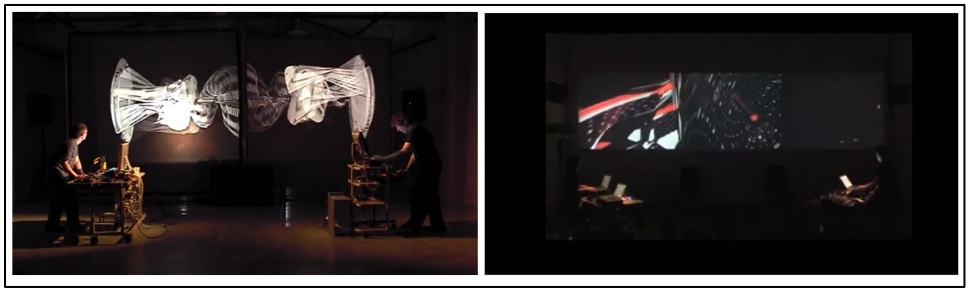

Figure 7 From a performance of Noisefold's Emanations.

Noisefold describes the audio component of their system as being a direct sonification of generated visuals. However, it is clear that the audio is mapped through certain abstractions, not unlike their visuals. Intensely literal data sonification of a moving image tends to be sporadic noise. This is not observed in *Noisefold 2.0*, which generates sustained tonal structures as well as noise. While it likely does not include direct sonification, it is apparent that their system attempts to sonify the data using relatively low-level software abstractions. Ironically—and intentionally—this results in highly abstract imagery, which is open to a vast range of possible interpretations. Such abstract visuals suggest that the system was designed with a spirit of exploration and experimentation. This holds a demeanor similar to the strict algorithmic compositions of the twentieth century, where systematized compositions were carried out with a respectful *come-what-may* attitude (see **2.6.1.1 John Cage's *Reunion* (1968)** on page 32). As a result, "...the audience must...work to assign significance or not to the emergent audio and visual

codes at play within the performance."[39] By using abstract visualizations of low-level properties, *Noisefold* deliberately avoids revealing the functionality of their interfaces. Consequently, there is no effort made to foster a shared sense of purpose.

2.6.3. Operational Logics

A useful technological tool to aid in the reconciliation of improvisation and composition is a concept from the field of game studies called operation logics. In its most simple form, an operational logic is the general description of an abstract process and its underwriting computation.[40] The total collection of a game's operational logics form what Noah Wardrip-Fruin and Michael Mateas describe as a playable model. [41] These are similar to simulations. However, unlike a simulation, playable models 1) require deeply responsive interactions with its user, and 2) support actionable mental models which "provide affordances for making gameplay choices that induce agency." Extending the method of operational logics to the context of live electronic performance provides the composer with practical design methods for maintaining authorship while enabling performer freedom.

When a composer understands the concept of an operational logic, it is a powerful authorial tool for developing interactive systems. A basic example of operational logic in the context of video games is "collision logic." The operational logic for object collisions then encompasses extensive descriptions of high-level verbal abstractions in relation to its lower-level computational processes. Table 3 shows fifteen key operational logics identified by Osborn et

[39] Stout.

[40] Mateas and Wardrip-Fruin, "Defining Operational Logics."

[41] "Michael Mateas on Façade, Part 1."

al.[42] Describing its communicative role, abstract process, abstract operations, and presentation—shown in Table 4—is a method of evaluation. This design method improves the accuracy of evaluation and analysis while developing and testing a system, allowing for greater authorial role and a better understanding of the behaviors which emerge from them. When extended to live performance systems, this translates to greater composer authorship.

Table 3 The fifteen key operational logics as outlined by Osborn et al., 2018

Camera	Chance	Character-State
Collision	Control	Game Mode
Linking	Persistence	Physics
Progression	Recombinatory	Resource
Selection	Spatial Matching	Temporal Matching

Table 4 Example of a catalogued operational logic (collision) as presented by Osborn et al., 2018

Collision Logic	
Communicative role	Virtual objects can touch, and these touches can have consequences.
Abstract process	Detection of overlaps between subsets of entities and the automatic triggering of reactions when these occur.
Abstract operations	Determine or alter which entities may collide with each other. Determine which entities are overlapping. Determine or alter the size of an entity. Separate the positions of two or more entities such that they do not overlap. Whenever one of the above predicates is or becomes true or false, trigger an operation of this or another logic.
Presentation	Shapes, images, or 3-D models for each entity whose dimensions match those of the corresponding entity, projected on a plane. Audiovisual effects when a collision occurs. The presence or absence of text indicating whether two entities are in physical contact.

Operational logic is a useful tool for designing and implementing procedural systems that produce a clearly defined range of emergent interactions. There are many tools an artist can use to create interactive systems, but these tools may or may not inherently encourage sensible or useful design methods. Consider Cycling 74's Max software. The Max environment allows for immense freedoms for constructing interactive performance environments. It has no inherited

[42] Osborn, Wardrip-Fruin, and Mateas, "Refining Operational Logics."

systems of checks and balances as an artist develops their patches. This is what makes Max so appealing to creators. Its unbridled programming environment unlocks the powerful capabilities of custom logics without steep learning curves required by other frameworks. As expected, however, these great freedoms come at a cost. Poorly implemented processes can introduce uncertain outcomes and a poor sense of composer authorship. By using operational logics as a design and analytical tool, composers can build emergent performance environments with a greater sense of purpose and with deeply responsive interactions.

Operational logics can be oriented vertically or horizontally, which informs the overall experience of the player. Vertical orientations create player expectations while horizontal orientations promote player freedoms. The section below will explore *Minecraft*, a game which consists of horizontal operational logics. *Minecraft* creates interactions that maintain authorship of the designer while simultaneously promoting a sense of freedom for the player.

2.6.3.1. Horizontal Operational Logics in Minecraft

Minecraft is an example of how operational logics can enable a seemingly infinite range of possible interactions within an expected milieu.[43] It is primarily developed by Mojang Studios using the Java programming language, and Xbox Game Studios using the C++ programming language. It runs on most available desktop and mobile platforms, and can boast a reported 126 million monthly players.[44] It is one of the most popular video games in recent years, and has proven to be a compelling interactive experience.

A particularly desirable quality of Minecraft is its ability to simultaneously support both developer authorship and player freedom. A player's interactions are carried out with the mindset

[43] Duncan, "Minecraft, Beyond Construction and Survival."

[44] Chiang, "Minecraft."

of, “What can I do?” rather than, “What am I supposed to do?” Minecraft’s playable model consists of several parallel operational logics that result in a dizzying level of potential interactions. A Minecraft world is essentially a three-dimensional cellular automaton where each cell consists of a block with its own operational logic (see **2.5.2.1 John Conway’s Game of Life (1970)**). Each block’s logic includes interactions with the player and their current state, taking into account the current item being held. A player begins the game with an empty inventory and without any prescribed objectives, as seen in Figure 8.

Figure 8 The initial moment of beginning a new Minecraft world.

At this point, the virtual environment offers a very limited range of affordances. The player can interact with the core movement logic of the game which simulates walking, jumping, and swimming. They can use their hand to interact with any block, which allows them to slowly and inefficiently begin collecting them as resources. The affordances of the world evolve as the player crafts various tools, blocks, consumables, and wearable items. Normally it would be appropriate at this point to summarize the player’s experience by describing their end-state. However, the playable model for Minecraft is such that operational logics truly allow for infinite emergent experiences; it has no designated ending. As depicted in Figure 9, the last moment of a

player's experience in a Minecraft world can be anything from building a functional in-game computer[45], to participating in an emergent multiplayer economy[46], to punching a tree.

Figure 9 Potential final experiences in a Minecraft world. Functional computer, LPG 2013 (left). Hermitcraft season six shopping district, GoodTimesWithScar 2020 (middle). New player punching a tree (right).

Minecraft's operational logics exhibit two main qualities that contribute to the discussion of reconciling composition and improvisation through technology. They are 1) the avoidance of scripted interactions, and 2) emergent player progression. This section will first explore the implications of avoiding scripted interactions when designing operational logics. This will then be followed by a discussion of how player progression might be embedded into non-scripted interactions.

If an interaction is scripted, it means that it follows a highly specified chain of events and portrays a specific meaning.[47] Scripted interactions inherently result in a vertical orientation of its underwriting logics. These sorts of interactions are commonly used when telling a narrative or progressing through a linear gameplay model. The operational logics surrounding a scripting interaction are typically focused on the user interface and intended character progression. A simple example of this can be found in the massively multiplayer online role-playing game

[45] LPG, *Minecraft: Redstone Computer*.

[46] GoodTimesWithScar, *My ULTIMATE Hermitcraft Season 6 Tribute*.

[47] "Michael Mateas on Façade, Part 1."

World of Warcraft. The leveling logic in this game involves players being required to complete quests with designated objectives. This can be seen in Figure 10 for the quest titled “Selling Fish.” To complete this quest, a player must travel to one of two prescribed locations to collect ten of the correct variety of fish. When a quest is complete, a known reward is endowed along with general statistical upgrades for the character.

Figure 10 Quest Log from World of Warcraft Classic.

This vertical style of scripted gameplay experience objectively reduces the player’s sense of freedom. Their actions are motivated by a desire to meet explicitly stated expectations. In contrast, Minecraft’s horizontal operational logics do not include scripted interactions, and the result is a vastly different experience; one that allows improvisational gameplay.

Consider the operational logic that drives the piston block, shown in Figure 11. Pistons have become a central component for highly complex behaviors and are a go-to block for building custom player interactions. Piston logic, like most other blocks, accounts for a basic collection of interactions. It is a mechanical block that extends its head when powered with a redstone component.[48] When powered, the piston will push up to twelve blocks. Initially, the piston block is unremarkable and does not appear programmed with any particularly interesting interactions. This changes dramatically when paired with the logic of other blocks. For instance, combining a piston with an observer block enables automated sugar cane farming. When combined with slime and honey blocks, pistons enable elevators, large doors, automatic wall builders, and flying machines. The developers have not directly programmed or intended any of these contraptions and their accompanying interactions. Rather, the interactions emerged as a result of horizontally oriented operational logics.

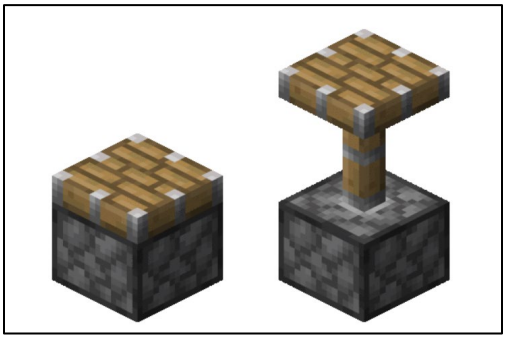

Figure 11 Piston block from Minecraft. Retracted (left) and extended (right).

[48] The redstone logic in Minecraft enables the creation of custom logic designed by the player. This is primarily done by creating redstone circuits which can activate and control various mechanisms. They can be designed to act autonomously or through player interactions. Additionally, they might respond to general world logics such as mob movement, item drops, or plant growth. Redstone circuits enable functionality at many levels of complexity, ranging from simple light switches to elevators or in-game computers.

Minecraft inherently encourages player progression, and yet its operational logic is predominantly horizontal. This presents us with a tantalizing conundrum. How does progression inherently emerge from operational logics that avoid scripted interactions? In other words, without scripted interactions, how does Minecraft motivate a player to act?

The game's survival logic motivates a player's initial actions. The virtual character will lose health if they grow hungry or if attacked by antagonistic mobs. When a player dies, all of their items drop and the character respawns elsewhere. The user's motivation to avoid harm is *intrinsic motivation*, meaning potential consequences for the virtual-self propel the player's interactions.[49] This motivates the player to build some form of shelter and acquire a source of food. Additionally, by acquiring the correct resources a player can eventually shield and defend themselves with various weapons and armor. Eventually, a player can reach a point in their world where survival is an afterthought. Interaction with the world then becomes an improvisational expression motivated by their personal creative inclinations. Minecraft does not prescribe the player's methods for survival, and yet the survival logic is nevertheless the spring board for any number of emergent interactions. Ultimately, intrinsic motivation provided by survival logic initializes action. Then, by horizontal logics, the player is able to improvise progression through unscripted, emergent interactions. As a result, Minecraft's developers retain authorship, and its users experience maximized freedom of interaction.

[49] Reid, "Motivation in Video Games."

Chapter 3.

Developing and Performing Self-regulating Performance Systems

3.1. The Self-regulating Performance System

As established in Chapter 2, typical methods for communicating a composer's expectation are incompatible with the notion of improvisational freedom. The discussion of reconciling these elements declare the need for new paradigms. The goal of a self-regulating performance system is to effectively communicate the desires of the composer through emergent, environmental avenues. I use the term *self-regulating* is to contrast typical paradigms of performance systems that might be deemed *composer-regulated*. A composer-regulated performance system would encompass explicitly stated expectations, making the composer the dominant driver of scripted interactions. Opposite of this is the self-regulated paradigm, where the performance environment is the dominant driver of emergent interactions. Theories of enactive cognition have informed us of the role the environment plays in our experience. As such, the environment becomes an unobtrusive location from which to express composer authorship. Additionally, the concept of weak emergence has outlined the mechanism for environmental communication. Modern technology facilitates these concepts by way of widely available hardware devices, as well as through various well-established software development frameworks. The self-regulating performance system employs these concepts and technologies to maximize improvisational freedom, composer authorship, and a sense of shared purpose.

It becomes necessary to enumerate the elements that characterize a self-regulating performance system. A self-regulating performance system:

1. Expresses the composer's intended interactions through cultivated emergence.
2. Avoids conveying explicit expectations.
3. Is intrinsically continuous.

4. Implicitly conveys cause and effect.
5. Is underwritten by horizontal operational logics.

This list details the most prominent features of a self-regulating performance system. There are, however, more nuanced characteristics expressed by the model which merit discussion. For instance, the process of cultivating emergent properties for a performance system tends to form a system that produces repeated, recognizable characteristics. This quality of emergent systems is ubiquitous in natural systems. A popular example is the termite mound, which emerges as a macro result of the microsensory architecture of the insect.[50] In this example, the mound is produced through horizontal operational logics; the logics in this case being each individual termite and its intrinsic motivation to relocate material. Despite there being no overseer of events, the structure is still able to emerge as a recognizable termite mound.

I am impressed by any process that can integrate the quasi-random behavior of things while remaining recognizable. Human behavior can either help or hinder this recognizability. If an improviser naturally pushes the boundaries of a general interactive system, then a performance's recognizability might waver. If an improviser is particularly rigid, then a lack of variation robs the performance of any sublime interest. Here is where the concept of a self-regulating performance system proves useful, in that it exhibits intrinsic motivation to act according to the composer's design.

The ideal scenario for a self-regulating performance system would have no need to consider a beginning or end. However, a pragmatic point of view will concede that musical performances tend to be confined to specified lengths of time. This implies that starting conditions need to be set. Facilitating these conditions requires assigning a role to each

[50] Johnson, *Emergence*.

performer interacting with the system. This role will motivate their actions, but it will not prescribe them. Recall the intrinsic motivation implemented by *Minecraft*'s survival logic. When beginning a new *Minecraft* world, interactions are informed by the well-being of the virtual self. Players are essentially given the role of *survivalist*. These starting conditions are necessary, as interactions are initially limited to a few logics. This is when the game most resembles a vertical playable model. As the player progresses and affordances expand, survival becomes a background element and operational logics become horizontal. The player ultimately accepts the survival logic as an environmental element, rather than as something generating highly specified expectations. In the case of the self-regulating performance system, this takes the form of an ambiguous role designated by the composer. The performer is then required to interpret how the assigned role translates to action. This exploration initializes the epistemic process of learning the performance environment through interaction. The nature of a performer's role in this system is reviewed within the coming examples.

3.2. *Ecosystem One* (2019)

My composition *Ecosystem One* is an example of live electronic music performance driven by a self-regulating performance system. It premiered in the Experimental Media Performance Lab at UCI in December of 2019. In this work, two improvisers are given an ambiguous role to fulfill; one to soothe and one to disrupt. The soother is Performer One and the disrupter is Performer Two. These performers are entangled in a complex relationship. The soother's goal requires motion to quell positive audio feedback, but this motion contributes to the disruptor's potential for control. The system resists change, and requires concerted effort from both improvisers. Performers experience a high sense of improvisational freedom, as I have not explicitly prescribed events and tasks. They are able to inject their own improvisational

inclinations into the performance as they interact with the environment I have cultivated. The sections below will discuss *Ecosystem One*'s prominent components. This will be followed by a discussion of the work using the analytical lens established in Chapter 2.

3.2.1. Ecosystem One: Intrinsically Continuous

To facilitate the continuous nature of a self-regulating performance system, *Ecosystem One* uses unbroken data streams for control input. Both performers have a microphone, and Performer One interacts with a MUGIC™ sensor. The sensor attaches to a mobile instrument, which was a kamancheh for the 2019 performance.[51] As the improviser performs, the sensor captures motion data from their instrument. The system's sensor logic manages this data, as catalogued in Table 5.

Table 5 *Ecosystem One*'s sensor logic.

Sensor Logic	
Communicative role	Performer can interact with audio feedback and virtual camera position.
Abstract process	Transforms the position of the virtual camera as mapped to the MUGIC™ sensor's orientation data. Affects the filter coefficients within the feedback subsystem.
Visual presentation	Indicates varying camera angles in accordance with the position of the sensor.
Audio presentation	Manifests as varying aesthetic qualities of feedback, enabling a wide range of improvisation.
Performative Outcome	Performer can explore a wide range of musical interactions involving the variable filter configurations. Greater control over this is established through camera position.

The first component controlled by the MUGIC™ sensor's orientation data is the center frequency and resonance coefficients for the project's central filter. The audio feedback subsystem for *Ecosystem One* runs through a bandpass filter. Performer One's instrumental

[51] The kamancheh is an Iranian bowed string instrument used in Persian, Azerbaijani, and Kurdish music. The kamanchech is related to the rebab which is the historical ancestor of the kamancheh and the bowed Byzantine lyra. It is played upright using a variable-tension bow.

motions determine the width and frequency of the filter. Leaning their instrument to the left or right changes the center frequency. Any motion forward or backward alters the filter's resonance. Their audio signal routes through the filter-feedback system, resulting in a tightly coupled relationship between their performance and the aesthetic of the feedback. Over time, the performer intuits the relationship between the audio feedback and their improvisations. After gaining some familiarity, not only can the performer control the pitch of a singular feedback tone, they are able to craft complex multiphonic feedback.

The second component controlled by the MUGIC™ sensor's data is the real-time position of the camera in the virtual environment. This virtual camera permanently keeps focus on the foremost spherical object within the virtual space, as seen in Figure 16. When Performer One moves their instrument, the camera mimics their motion. The motion control is not a one-to-one translation of position and rotation. Instead, the camera remains angled toward the foremost orb. This mechanism creates a parallax effect between the orb and the background elements. In perceiving this effect, Performer One is able to intuit the system's perception of their instrument's motions.

The combination of camera control and filter control enable a continuous exchange of audiovisual information between the performer and the system. The performer understands this information through intuition as the system does not contain explicitly stated compositional expectation. Consider an alternative method for portraying the same information in a more data-centric format. Depicted in Figure 12 is the main Max patch that displays data from *Ecosystem One*'s major subsystems. Both the bandpass graphic and the MUGIC™ graph—more closely depicted in Figure 4—convey the same data as the camera position and sonified feedback. The difference between these two methods of communication illustrates one of the primary points in

this text; that is, in order for compositional intent to remain unobtrusive to a performer's improvisational freedom, it must organically emerge from within the performance environment. If this patch were to replace the virtual environment, the performer would receive this information as symbolically represented abstractions. This sort of mental process is located at the forefront of cognition, and ultimately detracts from improvisational freedom.

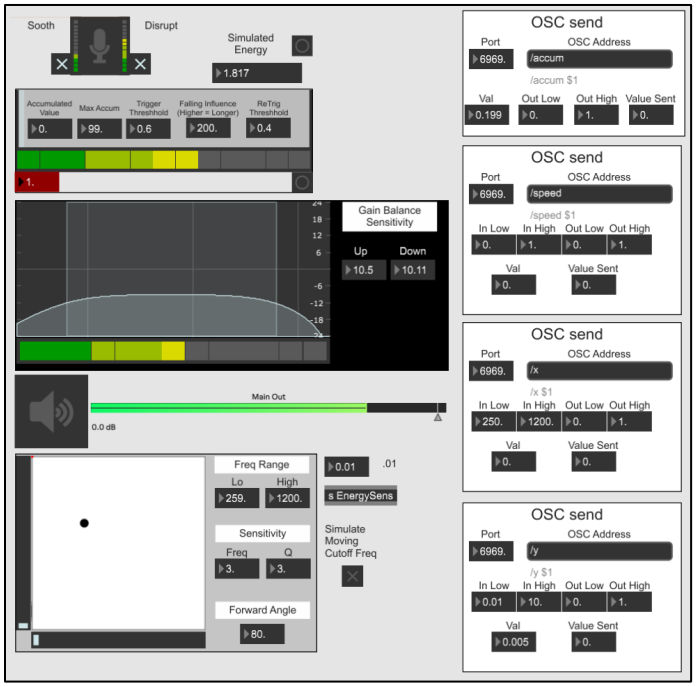

Figure 12 Max patch displaying performance data for Ecosystem One.

The third major component directly affected by the MUGIC™ sensor's data is the influence that Performer Two has on the system. *Ecosystem One's* influence logic manages the influence value, catalogued in Table 6.

Table 6 *Ecosystem One*'s influence logic.

Influence Logic	
Communicative role	Performers can interact with audiovisual components at different times.
Abstract process	Calculates influence values according to the actions of all performers.
Visual presentation	Indicating influence levels with the global particle system and other interactive atmospheric elements.
Audio presentation	Indicating influence levels by the severity and/or lack of distorted synthesis inserted into the feedback subsystem.
Performative Outcome	Performers are incentivized to pace their interactions so as not to exhaust their influence. It creates a repetition of musical structure with individual sections being roughly two or three minutes.

Performer Two—the disrupter—can interject audiovisual distortions into the work at any time. However, the level of potential distortion is dependent on their current level of influence. This is represented in Figure 20 as the "influence factor" square. The motion data from the MUGIC™ sensor generates this value, which the Max patch accumulates over time. Figure 13 show the subpatch that displays the amount of accumulated energy from the sensor. The left image shows zero accumulated influence, and the right image shows how the patch appears after it accumulates. This is typically a few minutes into a performance. Essentially, Performer Two's influence grows as Performer One soothes the unresting feedback. Here the performers engage in a never ending cycle of influence over the aesthetic, oscillating between calm or violent milieus.

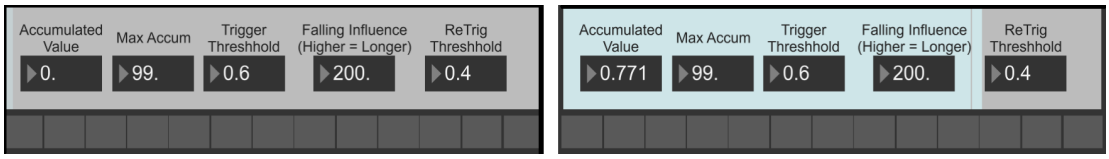

Figure 13 Max subpatch.

Performer One's motion also generates an *energy* value, which reflects the intensity of motion at any given point in time. *Ecosystem One*'s energy logic manages the energy value, catalogued in Table 7. It impacts the rate at which Performer Two's influence grows. It also

alters the visual renderings discussed in section **3.2.3 *Ecosystem One***: Visual Elements. The value processes through a smoothing function, meaning on a technical level, some may not consider it a real-time interaction. However, the response-delay resulting from the smoothing function is in a range of twenty to thirty milliseconds. For the purposes of this writing we can consider the energy value as reflecting the magnitude of motion in real-time—or at least as having a fast rate of response.

Table 7 *Ecosystem One*'s energy logic.

Energy Logic (Performer One)	
Communicative role	Performer can interact with the audio feedback logic at various magnitudes and aesthetics.
Abstract process	Calculates a level of intensity according to magnitude of physical motion while performing.
Visual presentation	Indicating intensity by the level of shader distortion on the foremost orb.
Audio presentation	Manifest as aural distortion over time as managed by the influence logic.
Performative Outcome	Performer can manage their level of intensity to balance the demands communicated by the audio feedback logic and the influence logic.

Performer Two interfaces with the performance environment through a microphone. They perform using a sledgehammer and a large pallet of wood placed on the ground. They are free to produce noise using whatever actions they deem appropriate. As described above, their noise has the potential to disrupt the stability of the audio feedback, but only at a magnitude determined by their current level of influence. Functions that analyze spectral content process Performer Two's audio. Noises from the hammer and pallet—and potentially from vocalizations if they are so inclined—route through a subsystem that resynthesizes the signal into heavily distorted audio based on the analyzed spectral data. The signal feeds into the audio feedback loop. This causes perturbations throughout the system that heavily distorts every audio and visual element in the system. If they patiently wait until their level of influence is at its maximum value, their injected distortions will have a loud, intense, and prolonged effect.

3.2.2. Ecosystem One: Conveying Cause and Effect

Using high-level abstractions to control logics of audiovisual elements allows a composer to guide the interpretations of observers. In doing so, the composer can design the perceived cause and effect of a self-regulating performance system. However, this presents the composer with the responsibility to successfully communicate purpose. Consider the most direct form of music visuals like those found on iTunes or Windows Media Player. It is difficult to "fail" when directly visualizing audio. This style of audio visualization exhibits very simple behaviors that minimally react to little more than the low-level data from the computer's audio buffer. The Ocean Mist music visualizer in Windows Media Player, depicted in Figure 14 likely derives abstractions of frequency magnitude through a Fast Fourier Transform. These low-level abstractions are straightforward and behave in a predictable manner. Mapping this data to lengths of lines does not leave much room for "failure." It is easy to see that the visualizer has fulfilled its function.

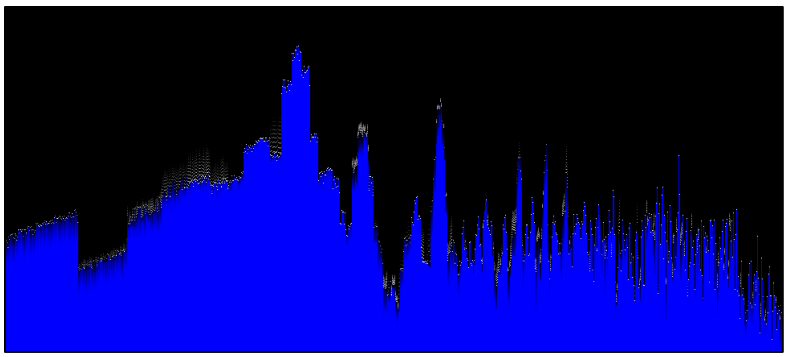

Figure 14 Screenshot of the Ocean Mist music visualizer in Windows Media Player.

By contrast, *Ecosystem One* requires that visual behaviors are handled with much greater delicacy. The visual elements seen in Figure 15 must communicate many things, as outlined by their operational logics. A few of these communications include which performer has the most control over the system, when other players can take control, how much disruption can occur, the

state of the feedback system, and the state of the primary biquad filter. Failing to accurately communicate these elements to the performers would result in general confusion, performer frustration, and a lacking sense of shared purpose.

Figure 15 From a performance of Ecosystem One (2019).

The high-level abstractions inform and govern visual behaviors, including the field of view, vertex displacement noise, floating particle density and turbulence, camera object tracking, image saturation and brightness, general shader properties (e.g., ambient occlusion, and global illumination), etc. If a single mapping is too powerful, the entire visual system can fail to communicate. I present this as a disadvantage, but in reality it is equally advantageous. It is a difficult balance, but successfully wielding high-level visual abstractions can create a powerful performance experience that exhibits the core characteristics of a self-regulating performance system.

Additionally, using high-level abstractions means that the visual elements can have non-disruptive, asynchronous relationships with the audio elements. This is because the mapping between their operational logics is not a direct, one-to-one translation of data from one dimension to the other. Instead, there is another level of abstraction at which the visuals, audio, and interactions are mapped. This liminal mapping level allows for the existence of quasi-autonomous entities within the visual space. Their fundamental behaviors can be completely

independent of audio data. They can be driven by other abstractions or by unrelated logics like direct performer interaction. There is also the option of mapping their independent behaviors to affect audio parameters. This asynchrony is not present within systems akin to music visualizers where low-level abstractions drive visual behavior. However, this liminal mapping level introduces the risk of incoherent chaos.

Developing high-level abstractions and their relationships requires a lot of time and iteration. It is easy to grasp the implications of a single mapping between logics, but the clarity of a system's functionality can become obscured as early as the second and third mapping. The optimal way to understand the system is to take the time to observe what it produces. Otherwise, careless mapping is emphasized by incoherent results and the loss of any communicative properties. Again, as stated above, this disadvantage is duly a great advantage when approached correctly.

Asynchrony lends itself to emergent expression of composer intention. With the visual and audio components being independently formed, I can express two contradictory forces simultaneously. For example, *Ecosystem One* contains a mapping that allows the instrumentalist to quell any major noise disruptions caused by the second performer. She must engage complex noise structures with vigorous improvisations. In doing so, she slowly regains control of the system and returns it to a calmer state. In this example, there are two distinct timescales at play. The first is the system's long timescale. It takes anywhere from thirty to sixty seconds to calm the system. The second is the smaller, quicker timescale that contains microstructures emerging from the instrumentalist's improvisations. The interactions for both of these timescales take place separately between the audio and visual domains. Her improvisation is primarily reflected in the audio, while the system's stubbornness is reflected in the visuals. As a result, two pertinent

domains of expression are emphasized, not cancelled out. If the communicative properties of their governing logics accumulated in a single domain—audio or visual—these separate timescales lose their significance. Through a separation of domain, the operational logics for *Ecosystem One* can take advantage of this domain-asynchrony and simultaneously express these multiple timescales.

3.2.3. Ecosystem One: Visual Elements

The virtual environment for *Ecosystem One* consists of a single scene that dynamically shifts between states according to the interactions of the performers. I designed the environment using the Unity framework. Each visual element contributes to a general communication of available interactions at any given point in the piece. I categorize these elements into three visual locales: the foreground, the background, and the atmosphere. Figure 16 depicts these elements in their basic state.

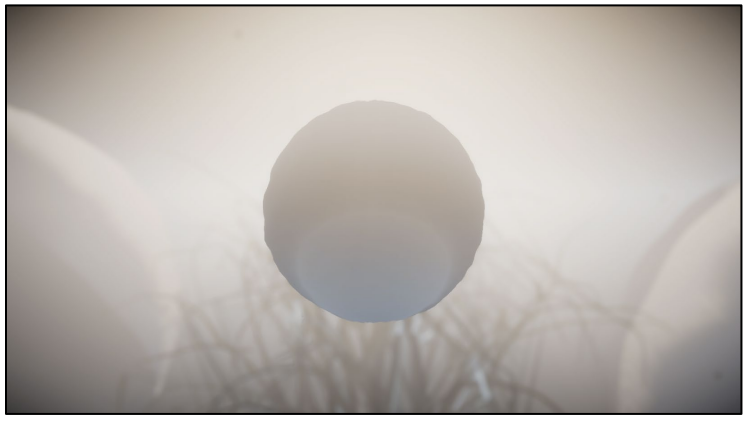

Figure 16 The basic state of Ecosystem One's virtual environment.

At the foreground of the environment is a spherical object. For the majority of the work, this object remains in focus while the rest of the scene is obscure. A custom distortion shader using Unity's shader graph workflow facilitates interaction with this and other objects. This

method allows for fast iteration cycles during the development phase, as opposed to programming custom shaders with GLSL. A C# script that manages information from the abstraction in Figure 19 control the shader's elements. Specifically, a noise map factors into the positions of the orb's vertices at a magnitude informed by. This serves as visual feedback for the performer, informing them of the system's perception of the real-time magnitude of their movement. From Performer One's perspective, the more they cause the orb to distort, the greater the potential for Performer Two to disrupt the system's equilibrium. This causes Performer One to pace themselves while soothing the audio feedback. If they are too active or aggressive in quelling the feedback, it will sooner enable Performer Two's disruptions. Their actions are thus inherently restricted to a particular range of intensity; they must perform within a range of momentum that soothes the audio feedback but simultaneously hinders the inevitable disruptions.

The background elements of the scene include two spherical objects visible on either end of the frame and a plant-like object between them. The background orbs share the same shader as the foremost, mimicking the same vertex distortions caused by Performer One. These are out of focus and contribute primarily to the overall reactivity of the visual scene. Additionally, they emphasize Performer Two's disruptions due to their large size. The objects remain in frame at the peak of the disrupted state depicted in Figure 17 because their scale factor is ten. The dramatic shifting in the camera's field of view during a major disruption shrinks the foremost orb into a small portion of the screen. Without the large background orbs, much of the image would consist of the near-featureless skybox texture.

Figure 17 A state of extreme distortion in Ecosystem One.

The plant-like object centered in the background of the frame is a custom tree mesh developed using Blender.[52] The purpose of the tree is to help emphasize the emotional state of the scene. To facilitate this, I created two animations. The first animation is of the tree blowing slowly and peacefully. The second animation portrays the extreme opposite, as if the tree thrashes in threateningly fast gusts of wind. Figure 18 depicts the model of the tree as affected by the animations. Typical animation workflows involve triggering various animations as informed by events. However, a self-regulating performance system does not include discrete events—meaning I must place these animations in a continuous relationship with its control signal. To accomplish this, I continuously blend two discrete animations at various magnitudes. This results in a dynamic animation that is compatible with the rest of the performance system. The tool enabling this animation blending is found in the animation framework available in Unity—specifically the feature called "blend tree."

The tree is obscure and out of focus in the virtual environment. I do not intend it to portray a literal tree. Rather, I want it to subliminally evoke the aesthetic of a tree or plant—as depicted in Figure 16. I map the interpolation value that blends its animations to Performer

[52] "Blender - Free and Open 3D Creation Software."

Two's influence logic. As a result, the tree's animation indicates Performer Two's current potential for disruption.

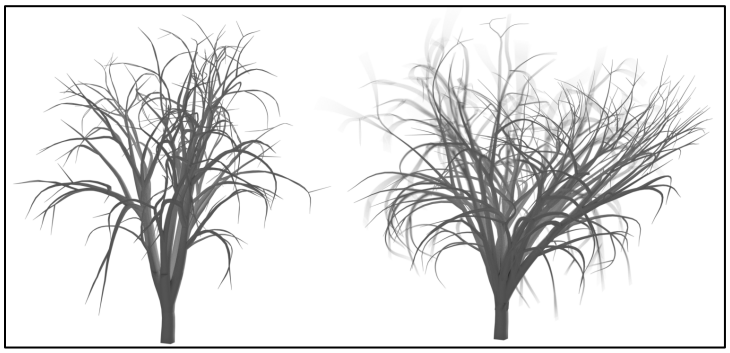

Figure 18 Animated tree mesh built in Blender. Calm (left), turbulent (right).

The atmospheric elements of the scene include volumetric fog, a global particle system, and post-processing effects on the final render. Some of these elements simply improve the perceived quality of the visual render, while others are included as interactive elements of the system. I provide general descriptions of these in Table 8. Note that I use the term *atmospheric* in this context descriptively, and not in reference to Unity's virtual atmosphere—though in the case of the volumetric fog and the particle system, both uses of the term confusingly collide. I find it appropriate to discuss post-processing effects as atmospheric elements because of the pronounced influence they can have on the emotional tone of the environment.

Table 8 Descriptions and interactivity of *Ecosystem One*'s atmospheric elements.

Atmospheric Element	Description	Interactive
Volumetric fog	Simulates the interaction of lights with fog, enabling realistic rendering of crepuscular rays (colloquially referred to as "god rays").	No
Global particle system	Generates up to 200,000 particles throughout the virtual scene. The behaviors of the particles are designed using a visual effect graph.	Yes
Post-processing effect: Anti-aliasing	Smooths jagged or pixelated edges on curved lines and diagonals that are originally considered to be artefacts of the rendering process.	No
Post-processing effect: Bloom	Creates graduated light that surpasses the borders of bright areas in an image.	Yes
Post-processing effect: Vignette	Darkens (typical) or desaturates the edges of an image with respect to the center. Often simulated in a subtle manner to increase the photorealism of a render.	Yes
Post-processing effect: Film grain	Randomly simulates the noise typical of small particles on photographic film. Grain is typically considered qualitatively undesirable, but can help add realism to a simulated image.	Yes

3.2.4. Ecosystem One: Mappings in a Self-regulating System

Mapping typically involves the connection between some sort of gesture and an audible result in a musical performance.[53] *Ecosystem One* extends the concept of mapping to incorporate the connection between any two operational logics in a way that facilitates self-regulating behavior. This differs from the practices typically found in algorithmic composition, which is often the process of designing a gesture and then "applying a mapping process to turn that structure of the conceptual domain into sound which may display the original conception in some way."[54] This algorithmic approach to mapping seeks to ensure that the gesture is embodied and perceptible in the music as an end result. This approach more closely relates to vertical logic orientations, while self-regulating behavior results from horizontally oriented logic. The traditional vertical approach to mapping does not facilitate emergent behaviors. In contrast to

[53] Doornbusch, "A Brief Survey of Mapping in Algorithmic Composition."

[54] Doornbusch.

this, the mapping in a self-regulating performance system fosters circular causality and seeks to generate emergent musical phenomena as cultivated by the composer (see **2.5.2 Emergence**). This circular causality is apparent in Figure 20.

A communication subsystem handles interaction with *Ecosystem One*'s virtual environment. This subsystem transfers information from a Max patch to the visual software using the OSC protocol. This subsystem is what connects each of the system's operational logics. Rather than reprogramming the same process for each connection between logics, I created a reusable abstraction. Before delivering, I use the Max abstraction in Figure 19 to appropriately scale each stream of data. Using this tool the OSC address, OSC port, and scaling parameters can be easily set for multiple control signals. A communication workflow between different logics becomes quite tedious to develop as data streams increase in number. Cultivating emergence exacerbates the tedium, which is a process that requires hundreds—if not thousands—of iterations through system configurations. Using general purpose tools like this abstraction greatly optimizes the development process and helps to ensure successful expression of composer authorship.

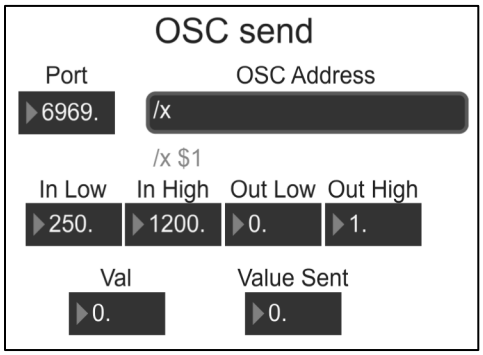

Figure 19 "OSCsend" Max patch abstraction for OSC communication.

Figure 20 displays a generalized diagram of the mappings for *Ecosystem One*. The influence of each component and subcomponent link to any other. Additionally, all audio signals can be both sonic data and control data. Essentially, the signal is both the musical content and the controller that drives it. The overall process of *Ecosystem One* takes place in a continuous loop of circular causality. Each component of the system's structure both begins and ends the process.

3.2.4.1. *Ecosystem One*: Creating Self-regulating Behavior

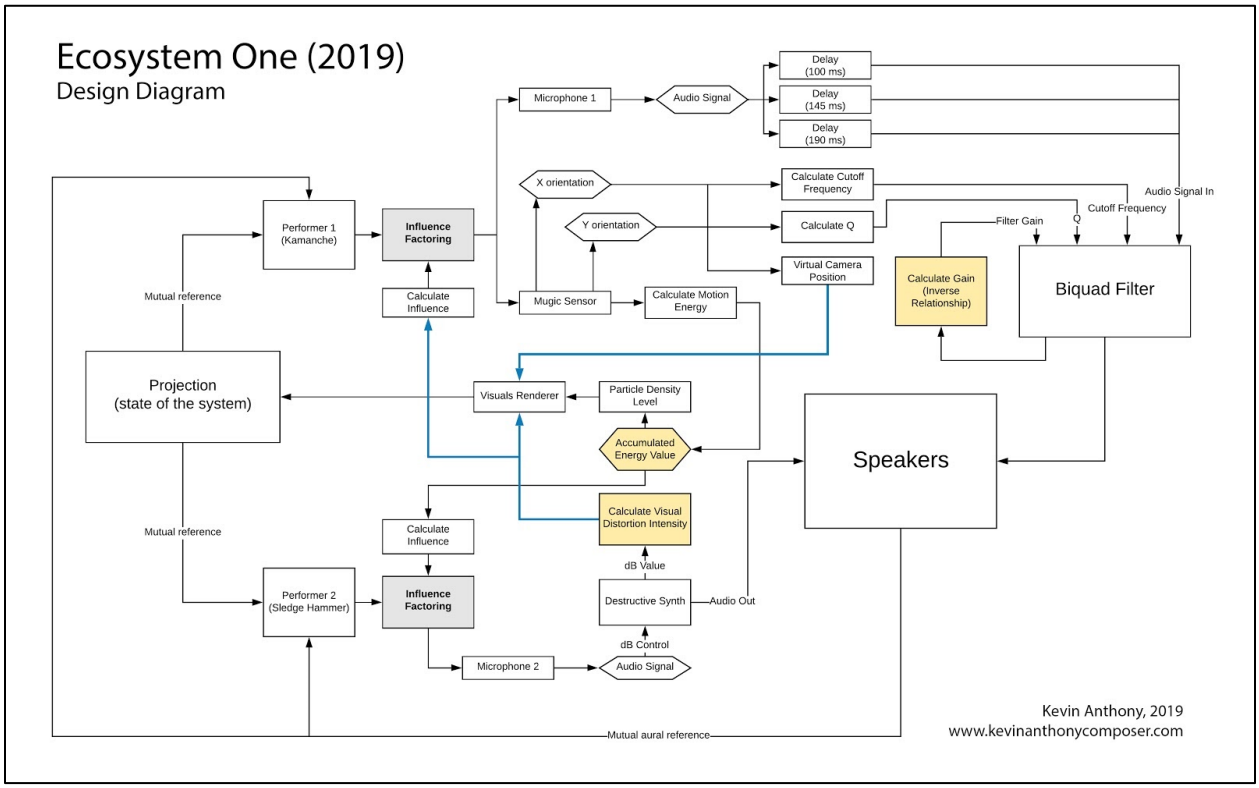

Figure 20 Mapping diagram for Ecosystem One (2019).

Self-regulating behaviors are facilitated through circular causality. Its implementation involves circular relationships through and between the system's operational logics. This produces emergent behavior informed by the notion of weak emergence. A small example of this is *Ecosystem One*'s filter subsystem, which produces emergent musical structures as a result of its circular mapping. Essentially, the filter becomes aware of itself by reinterpreting its output as

one of its own control signals. Figure 21 is a simplified version of the filter subsystem used in *Ecosystem One*. In Figure 20, this is visualized in the components surrounding the "Biquad Filter" square of the diagram. Designating a starting point is contrary to the circular nature of the system's structure, but it intuitively makes sense to begin an explanation with the incoming audio signal. This signal is fed through a biquad filter to three destinations. The first destination is the loudspeakers within the performance venue. The second destination is through a time delay and then back into its originating biquad filter. The third destination is through a scaling component which interpolates the signal's decibel value to a number between one and zero. The relationship is inverted, meaning the louder the signal, the lower the value between zero and one. After this value is inversely scaled, it is used as a control signal for the gain coefficient of the biquad filter. Without this mapping, the filter would blow up as a result of its feedback-inducing architecture. However, the circular mapping produces self-regulating behaviors that manage the feedback. In tandem with Performer One's control from the MUGIC™ sensor, this results in emergent performance structures.

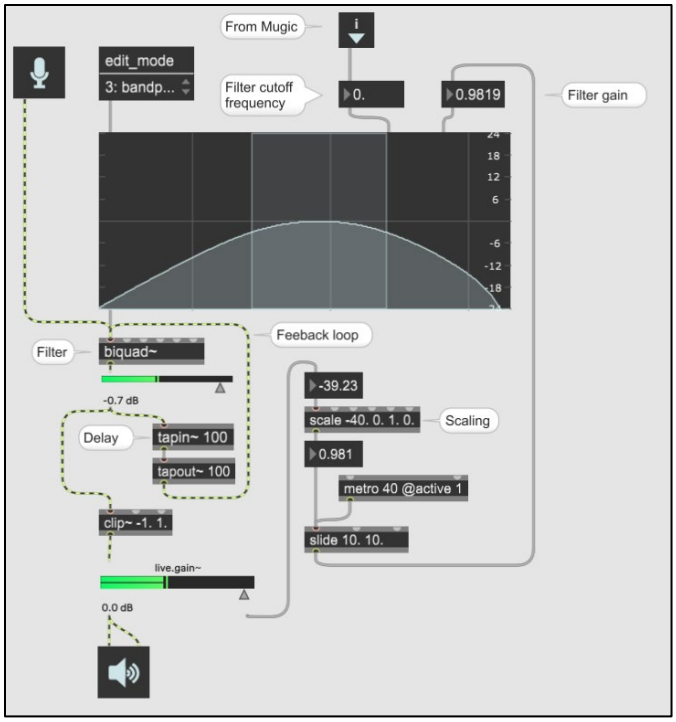

Figure 21 Max patch demonstrating self-regulating behavior.

It is important to note that designing self-regulating behavior with computer software requires the manual insertion of weight and resistance. The mapping shown in Figure 21 is self-regulating in that it seeks an equilibrium between several components. Without any forced resistance, the computer will find this equilibrium very quickly—often instantaneously. From a compositional point of view, this creates uninteresting behavior and does not provoke performer response. In a way, the parameters become too clean and thin. Without resistance, the system fails to capture a certain heaviness that comes with interacting with our environment. In Figure 21, a simplified example of manually inserting resistance is seen with the *slide* object. This prevents instantaneous calculation of the system's immediate equilibrium. The result is a continuous performance component with composed interactions that are reminiscent of the physical world.

3.2.5. Ecosystem One: Analysis

Ecosystem One effectively maximizes a sense of authorship within the composer. As the composer, I am certainly incentivised to make this claim. Unfortunately, there is no way for me to be entirely objective in relating my experiences. To instill confidence that my claim is earnest, I direct the reader to the analysis of *Sinew* in section 3.3.4. There, I express a level of dissatisfaction in my sense of composer authorship for that work—a dissatisfaction that exists in part because of my perceived success with *Ecosystem One*'s authorship. In light of this, I assure the reader that I am honest in expressing a strong personal sense of authorship.

My heightened sense of authorship in *Ecosystem One* is a result of the musical structures that emerge. A performance begins with Performer One delicately interacting with audio feedback. She interfaces with the system through her audio input, as well as the MUGIC™ sensor attached to her instrument. Her gestures and interactions inform her given role, which is the *soother*. When the feedback subsystem is relatively steady, her improvisations are soft and delicate. When tones inevitably become unruly, her improvisations grow in intensity—soothing the feedback. As a direct result of maintaining this balance, Performer Two's potential to disrupt grows. When he feels inclined to do so, he dramatically smashes his sledge hammer onto his wooden pallet. This results in a severe unbalancing of the system's component relationships, manifesting as powerful audiovisual distortions. The echoes of this distortion slowly dissipate, allowing Performer One to regain control of the system. At this point, Performer Two's influence is spent. He can only regain it through Performer One's balancing act.

There are two prominent musical structures described above. The first involves the overall form of the work. The form consists of two conjoined sections of material. The first section is Performer One's attempts to balance the audio feedback. The second section is Performer Two's disruption. Performer One regaining control serves as transitional material that

leads to another repetition of these two sections. These sections repeat roughly three times within the context of an eight- to ten-minute performance.

Another musical structure that emerges during *Ecosystem One*'s performance involves the general intensity of the performer's improvisations. I designed the audio feedback system to become more difficult to balance for Performer One as Performer Two's influence grows. This means that Performer One's interactions begin with gentle musical gestures that eventually become quite dramatic. As this happens, there is a tangible increase of tension that builds toward Performer Two's intense moment of disruption. Following the disruption, Performer Two's improvisations mirror the previous material. Instead of improvising with increasing levels of intensity, Performer Two's gestures slowly dissipate in reaction to his waning influence.

The manifestation of these forms in a practical performance setting created a very strong sense of composer authorship. I cultivated the ebb and flow of influence through the highly iterative compositional process necessary for designing a self-regulating performance system. I fully intend for these structures to emerge as a result of carefully designed component relationships.

As established in Chapter 2, strong composer authorship typically indicates a higher potential for a sense of shared purpose, and a decreased level of improvisational freedom. *Ecosystem One* avoids this pattern by expressing compositional intent through the performance environment. Typically, the intention for such improvisational structures might take the form of a written timeline—similar to Agostino Di Scipio's *Two Pieces of Listening and Surveillance*. However, for *Ecosystem One*, I do not explain these structures to the improvisers. Instead, they are discovered through the intrinsic motivation established by the ambiguity of their given roles—the soother and the disrupter. Improvisers interact with the environment according to their

personal improvisational inclinations, resulting in strongly perceived improvisational freedom. Following the 2019 premiere of the work, I queried Performer One regarding the general performance experience. She replied:

> "I have never performed something like this. There was structure…[I] could tell there was a musical form, but I did not have to look for cues or wait to be told to do another section. [I] can just perform."

It is interesting to note that this experienced improviser had a difficult time describing the simultaneity of improvisational freedom and composer authorship. What she and I experienced is atypical of prevalent paradigms, and made possible by the model for a self-regulating performance system.

Chapter 2 establishes that reviewing the patency of cause and effect throughout a performance is a practical method for evaluating its potential to foster a sense of shared purpose. As detailed in section 3.2.2, *Ecosystem One* tightly couples the improvisational interactions with audiovisual expressions. This means that audiovisual behaviors are easily seen and heard to be related to the physical and musical gestures of the performers. As such, the performers' struggle for control over the system is made obvious and apparent. It is true that a sense of shared purpose is inherently only an ideal, and is not necessarily assured by any method. However, a strong display of cause and effect—like the one expressed by *Ecosystem One*—allows observers and performers alike to interpret the work from the same understanding of component relationships. This greatly increases the potential for fostering a sense of shared purpose.

Following the premiere of *Ecosystem One*, I was approached by several individuals who freely offered to me their interpretation of the work. The purpose of these exchanges seemed to be for them to receive validation that their theory was correct. Though it is a less practical method for evaluating shared purpose, I saw this as a general indicator that concert attendees

sensed a motivating purpose behind the work. To my delight, each of the interpretations described a personal setting of the archetypical balance of control and freedom. One individual described the ebb and flow of weather and the climate; another, the struggle of a codependent relationship. I validated each interpretation, pointing out the underwritten war between control and freedom with each exchange.

Ecosystem One successfully maximizes improvisational freedom by expressing compositional intent through the performance environment. As intended interactions and musical structures emerge from the performer's improvisations, my sense of composer authorship is heightened. The similarity of interpretive narratives offered to me by observers indicate that patent cause and effect enabled a prevalent sense of shared purpose. *Ecosystem One* successfully accomplishes all of this by satisfying the demands of a self-regulating performance system.

3.3. *Sinew* (2020)

Sinew is a stand-alone application for interactive electronic music performance. A self-regulating performance system drives interactions with the software. To facilitate a post-COVID environment, the work premiered as a streamed performance with four improvisers on June 25th, 2020. This work presents the performance by way of a video production as well as a recording of the streamed premiere.

A performance of *Sinew* involves interacting with various on-screen objects through audio input. A performer periodically sees a prompt assigning an ambiguous role that establishes intrinsic motivation. Given this role, the performer is able to interpret *Sinew*'s virtual habitat according to their own inclinations. Each virtual object performs musical improvisations based on their relationship to a given improviser.

Moving forward, the following section will discuss how the COVID-19 pandemic informed *Sinew*'s creation. After this, I will present in detail the architecture of its underwriting self-regulating performance system.

As per the requirements of emergent, non-scripted interactions, the operational logics for *Sinew* are oriented horizontally. Each logic acts according to its present circumstances. Additionally, each logic supports interaction with other designated logics. In using the concept of operational logics, I was able to design and cultivate emergent behaviors according to my artistic inclinations. The key logics for *Sinew* are feature logic, movement logic, energy logic, and performing logic. I catalogue each of these logics below using a method similar to the cataloguing presented by Osborn et al. However, I accommodate this musical context by altering the cataloguing method to include descriptions of visual presentation, audio presentation, and performative outcome. Following each logic is a presentation of related tools, as well as an overview of its role within the performance system.

3.3.1. Impact of the COVID-19 Pandemic

The COVID-19 pandemic had a transformative impact on *Sinew*'s development and performance. *Sinew*'s premiere run was originally scheduled to take place on May 9th and 10th of 2020 as live performances in the Experimental Media Performance Lab (xMPL) at the University of California, Irvine. On March 12th, students and faculty with scheduled performances were notified that in efforts to mitigate the spread of a novel coronavirus, COVID-19, no audiences would be allowed for any performances or exhibitions in the xMPL. Fortunately, plans were made to support the streaming of all performances in the venue in lieu of live audiences. On March 16th, however, only four days later, we received a brief message: "The xMPL Theatre is now closed and unfortunately, you will not be able to put on your project." No

context was given, and none was needed. In a very short window, the number of COVID-19 cases at this point had risen to nearly 180,000 and spread to over 155 countries.[55] At the time of this writing, cases are in excess of 17.2 million, with more than 670,000 deaths worldwide.

In response to the grave uncertainty that came from the impending threat of COVID-19, students were encouraged to return to their place of permanent residence. After an unsuccessful outing to purchase groceries, my wife and I decided it was best to relocate ourselves and our three young boys. Three hours after making this decision, we were on the freeway headed north to my in-laws' farm.

The many prominent social and political events surrounding the COVID-19 pandemic have led me to contemplate the nature and purpose of my work. Music serves countless functions: calming anxiety, providing focus, being a welcome distraction, encouraging the downtrodden, lifting those who are mourning, reconciling differences, and accompanying the isolated. Through present circumstances, I have come to the firm conclusion that musicians are the medics in a psychological war between facts and fear.

This conclusion greatly motivated the continuation of my work and research. Without the possibility of a live performance, I decided to redesign *Sinew* in a way that was compatible with a post-COVID environment. My goal was to make it widely accessible. Originally, *Sinew* used a similar framework to *Ecosystem One*, which used a substantial Max project to process audio and Unity to render the visual elements. Such a setup is difficult to distribute to performers and other users remotely, as I cannot be present to assist with any issues specific to one individual's hardware. I needed *Sinew* to be a stand-alone software that did not require manual configuration

[55] "Covid-19 Situation Report #9, March 16, 2020 - World."

for machines on an individual basis. It also needed to be compatible with both Windows and Apple operating systems.

Naturally, I chose Unity as the ideal tool for developing such a piece of software. However, the typical Unity workflow does not natively include many options for advanced audio processing. This obstacle necessitated the creation of several new tools and methods for musical interactions within Unity. I present these in this paper along with a streamed performance of the work.

The COVID-19 pandemic certainly proves to have been highly transformative toward *Sinew* in its present form. Had the pandemic not been a factor, *Sinew*'s performance would have come and gone, satisfying the requirements for my dissertation. This would have been easier and much less complicated. However, the circumstances of the pandemic forced my work to consider more important—and I feel, much less frivolous—aspects of music and its relationship to us as humans. The new tools and methods that emerged from the altered development process stand as useful contributions that would not exist otherwise. I offer *Sinew* as a reflection of how we engage with our environment and those around us, and I offer it to the public in the hope that it generates meaningful musical experiences.

3.3.2. Sinew: Operational Logics

Before exploring *Sinew*'s operational logics, it will be beneficial to explain the *common attributes* class attached to every object. Figure 22 depicts the class as a monobehaviour component in the Unity editor. This class is a container for an object's attributes. Its name correctly implies that every object shares the same attributes, including the improvisers. The significance of this is that despite being internally the same, content-wise, each object still behaves in a distinguishable and recognizable matter. This is because their behaviors are

emergent in nature, originating from the relationships formed by *Sinew*'s underwriting logics. Understanding this will help further the discussion of *Sinew* as a self-regulating performance system.

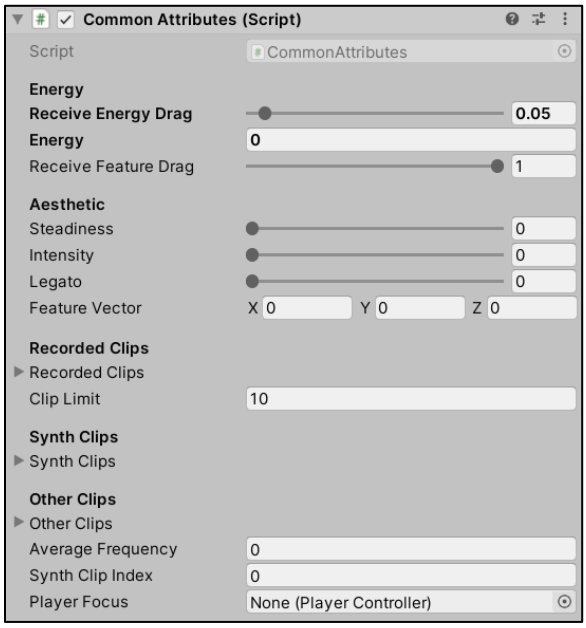

Figure 22 CommonAttributes C# class for Sinew's objects and performers.

Each of *Sinew*'s objects interact with an energy value in a specific way. The operational logic that manages energy is a natural starting point, as it affects all other logics. Along with describing the objects encountered while performing *Sinew*, this section will review their underwriting operational logics. It makes sense to begin with energy logic, as it influences every logic in the system.

Table 9 *Sinew*'s energy logic.

Energy Logic	
Communicative role	Each object is dependent on a source of energy, whether it is audio input or the energy of other objects. Each object is attracted to their preferred energy source as they need it.
Abstract process	The proximity of objects either increments or decrements energy levels according to their preferred energy source. Can also transfer energy altruistically.
Audio presentation	Indicated through descriptive audio performance as energy transfer occurs.
Visual presentation	Indicated by particle systems flowing in the direction of an energy transfer
Performative Outcome	Performers are incentivized to perform in a manner that benefits certain game objects according to that object's expressed energy.

Energy transfers within *Sinew* are not discrete events. Every single transfer is constantly occurring. However, proximity is a major factor in the transfer equation. The distance value between objects transferring energy passes through a sigmoid function (see section 3.3.3.5). This results in a specifically crafted s-curve between zero and one which factors into the equation for energy transfer. This allows for continuous mappings while designating some form of limitation. For instance, I automatically map any two spheroids together with a continuous relationship. Through this relationship, they continuously balance their values. If their distance is too great, the attempt is minimalized and almost non-existent.

Performing *Sinew* is a similar experience to that of *Ecosystem One*. However, for *Sinew*, the system gives performers their roles dynamically according to their performance behaviors. Their interactions with on-screen objects also impact which new roles are given. At the outset of the performance, a performer can receive the prompt, "Inform them." Further along in the performance, they might see, "Listen to them." These prompts are displayed through on-screen text. They generate intrinsic motivation for the performer to interact with what they see. The ambiguity allows for interaction informed by their own improvisational inclinations, as well as by my embedded compositional intentions.

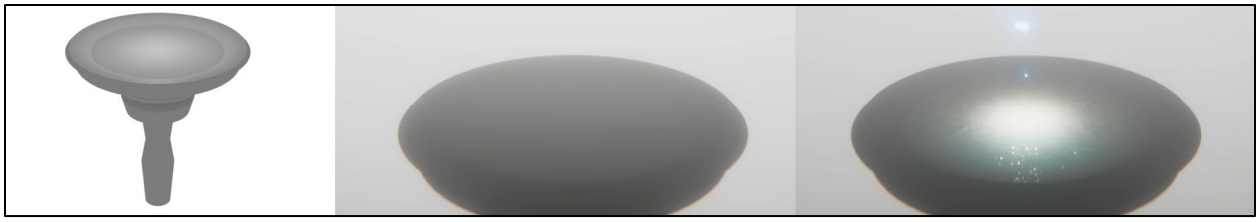

Figure 23 Sinew's well. Render of model (left). Unlit in performance from low energy value (middle). Lit it performance from high energy value (right).

The first object that performers encounter is the well, depicted in Figure 23. This well does not immediately exhibit any obvious affordances. The well offers new interactions as an ambiguous prompt informs the performer's improvisations. Their audio input generates lights that flow into the well. This light appears to accumulate, and the more they perform, the brighter the well becomes. Figure 24 depicts this interaction. After enough accumulation, the well extinguishes its light and produces a floating spheroid. This process repeats, producing more and more spheroids over the course of a performance.

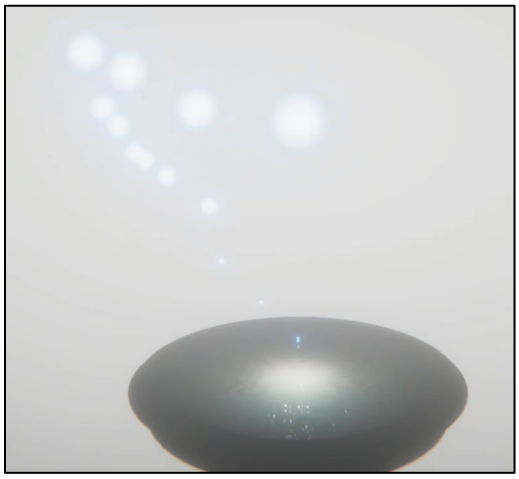

Figure 24 Energy value transfer depicted as lights flowing from a single performer to the well.

Figure 25 Energy value transfer depicted as lights flowing from multiple performers to the well.

The floating lights that accumulate in the well are a visual abstraction of an energy value transfer. Each object contains an individual energy level as a common attribute. As the performer improvises, they increment their energy value. This energy then transfers to the well according to its current state. The transaction is not a complete one-to-one transfer of value. The well's reception of energy is dependent on its current disposition. The poorer the well's disposition, the less efficient the energy transfer. The well's disposition is dependent on the state of the objects in the environment as well as the behaviors of the performer. For example, if the well's energy level is very low, this has a negative impact on its disposition. Additionally, if the objects in the scene exhibit low energy values, this also has a negative impact. The inverse of this is also true, where higher energy levels equate to the well exhibiting a more favorable disposition.

The spheroid object that is produced by the well begins with an energy level that is equivalent to the well. Its movements, interactions, and performance behaviors are all motivated by a need for energy. For the spheroid, the energy value might be representational of its general health. Without energy, the spheroid will return and dissolve back into the well.

Immediately after being produced by the well, a spheroid searches for energy sources, of which there are only two. The first is an improviser. As a spheroid approaches a performer—visually represented by movement toward the virtual camera—it has a greater opportunity for a

value transfer. Upon finding the performer, they do not force a value transfer. In a sense, they "request" it. The spheroid attempts to adopt the same aesthetic parameter values as the performer—see Figure 22 on page 74. In doing so, it achieves parity with the performer and can more efficiently receive their energy.

The second source of energy for a spheroid is other spheroids. The value transfer between spheroids is similar to the process just described, though I have designed the spheroids to be inherently altruistic. As spheroids approach one another, they attempt to balance an equal share of energy between them. The effectiveness of their energy transfer results from their level of aesthetical parity. To increase their chance of a fair outcome, they will simultaneously attempt to balance their aesthetic values along with their energy levels. The overall effectiveness of this process is determined by their total levels of energy. The higher their collective energy levels, the more efficient this transfer becomes.

Table 10 *Sinew*'s movement logic.

Movement Logic	
Communicative role	The disposition of objects evolve according to their interactions with the performer and other objects.
Abstract process	Calculates motion patterns according to the disposition of the object as determined by energy logic.
Audio presentation	Indicates the flow of motion in tandem with sound production produced by performing logic.
Visual presentation	Emotes a particular level of energy and responsiveness through a change in position and blended animation.
Performative Outcome	Performers are incentivized to interact sonically with nearer objects, motivated by the objects particular disposition.

Initially, a spheroid is limited to movement within a very narrow turning radius. They can, however, move forward and backward, making slight turns as they do so. Their turning radius increases as their level of energy increases. Additionally, their need for energy informs their target destination. As mapping in *Sinew* is continuous, they are individually aware of their

closest source of energy. Reaching this source might necessitate great effort. With low energy, their movements appear sporadic, as if lost. As their energy increases, their movements appear more determined along with their performance behaviors.

Table 11 *Sinew*'s performing logic.

Performing Logic	
Communicative role	Performers can influence the performative expressions of various objects through improvising different musical aesthetics.
Abstract process	Performs musical gestures in a format and style informed by the performer. Gathers and analyzes audio using feature logic. Performs according to gathered features and features of a targeted performer.
Audio presentation	Indicates aesthetical preference and experiences through various modes of performing logic
Visual presentation	Indicates aesthetical preference through various expressions of color.
Performative Outcome	Performers are incentivized to produce various aesthetics according to its long-term effects on various audiovisual elements.

Along with transferring energy and aesthetic values, spheroids also transfer audio. When a spheroid first receives energy from an improviser, it also records their audio signal. This becomes the primary material with which that spheroid performs. When in close proximity, spheroids will listen to each other's performance behaviors. When the spheroids achieve parity, they perform with similar emergent behaviors.

When improvising music, a spheroid does not simply playback the recorded audio. They perform using a dedicated audio performance system of my design. I implement this using the performer class depicted in Figure 26. There are two performance modes. While these modes indicate a discrete split in the operational logic, they ultimately blend in and out of one another.

The synth mode uses the object's recorded audio clips as material for granular synthesis. I map the ADSR of the synthesizer to the spheroid's aesthetic values, and its energy informs the length of a note. The interactivity of this performance mode comes from its source of pitch, which is the current pitch of the spheroid's preferred improviser. Notice in Figure 22 that one of

the common attributes is a “player focus.” This focus designates a favorite improviser, which the system typically assigns as the one that most closely matches its aesthetic values. If a spheroid is favoring the synth performance mode, they often perform in unison with this improviser. This unison is not constant, however. The spheroid’s true performance behavior is a blending between both modes. The second mode is recorded mode, which generates improvisations by reorganizing and processing its acquired audio recordings. The blending of these two modes results in a sort of circular causality between the performer and the spheroid. The line blurs in terms of who is following which improvisation. As a performer engages in unisons with the spheroid, they both simultaneously inform each other’s newly improvised elements.

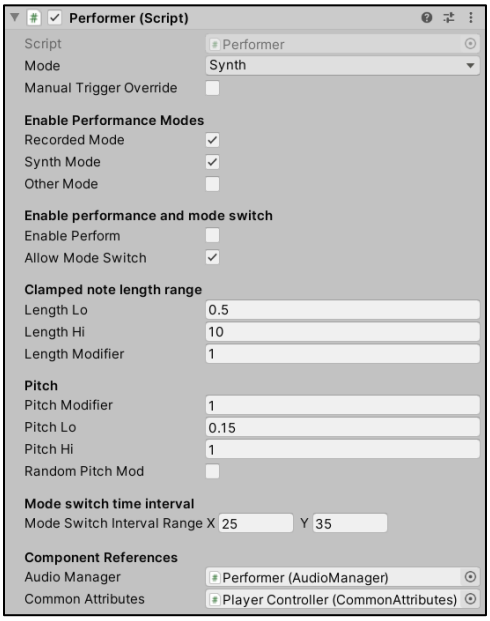

Figure 26 Sinew's C# Performer class as a monobehaviour component in the Unity editor.

Figure 27 Three bulbs in Sinew approaching the well.

Another object that performers eventually encounter is a bulb, three of which are depicted in Figure 27. Bulbs are large, slow, organic-looking objects with openings at their base. They are not highly expressive and they do not exhibit many behaviors. Their pulsing musical gestures are very low in pitch, with increased intensity as they move closer to the well. They are attracted to, and repelled by, the aggregate energy of the objects in the scene. While in close proximity to the well, they force energy away from all objects in the scene, including the performers. The bulbs use this energy to produce and release their spawn. After expelling their energy, a bulb will retreat for a time.

Figure 28 Spawn from a bulb in Sinew.

The spawn that bulbs leave behind are visibly antagonistic objects—shown in Figure 28. Unlike the seeking behaviors of the spheroids and the bulbs, spawn move in a stochastically dominated manner. This helps to reinforce a notion of recklessness and danger. They indiscriminately drain energy from the spheroids, the performers, and the well. Their musical performance behaviors are similar to that of the spheroids. However, their attribute values reflect their recklessness, which causes unpleasant and distorted improvisations. After filling their energy values, they retreat from the scene as if in search of more energy.

3.3.3. Sinew: Feature Logic

Table 12 *Sinew's* feature logic.

Feature Logic	
Communicative role	Performers can influence the audio and visual expression of various objects through improvising various musical aesthetics.
Abstract process	Derives magnitudes of several musical features according to the aesthetic of the improviser's audio input.
Audio presentation	Indicates aesthetical preference through various modes of performing logic
Visual presentation	Indicates aesthetical preference through various expression of color.
Performative Outcome	Performers are incentivized to produce various aesthetics according to its long-term effects on various audiovisual elements.

Sinew's feature logic introduces new tools for feature extraction for the Unity development platform. The extracted musical features serve as abstractions that allow the system to be aware of performance aesthetics. Prominent features used in *Sinew* separate into low- and high- levels of abstraction. Low-level features include frequency, decibel level, and spectral flux. High-level features include steadiness, intensity, and legato. It is through these core performance features that improvisers interact with the system, as described in Table 12.

Unlike *Ecosystem One*, *Sinew* does not take advantage of any proprietary sensors and custom devices built by myself or others. If the software is to be accessible to a general user-base, it could not require such devices. Instead, it demands compatibility with commonly owned hardware and their respective frameworks. For this reason, the microphone is the single sensing interface used by the performer. The MUGIC™ sensor used in *Ecosystem One* proved to be a very immersive method for connecting performers to the virtual performance system. How could *Sinew* attempt to create a similarly immersive interfacing using a single microphone? I solve this issue by extracting features from available audio data, which requires custom tools within Unity framework.

3.3.3.1. Low-level Features

The Unity scripting API provides limited access to the engine's low-level audio data and behaviors. This is intentional, as managing audio data at a scripting level is fickle and unpredictable. The API's audio function primarily used in *Sinew* is `AudioSource.GetSpectrumData`[56]. The `GetSpectrumData()` function provides an array of floating-point values representing the frequency magnitudes of current audio data output. The length of the array must be a power of two. A similar function call in an audio-centric development environment might occur in successive data chunks, processing every audio sample. As Unity is a visual-centric development environment, `GetSpectrumData()` does not account for which samples have or have not been returned. The result is an asynchronous stream of data that may or may not be processed. Performing an inverse Fourier transform on the array and readying it for audio playback is possible, but this does not produce clean audio for the reasons described above.

Though the audio data available within the Unity scripting API is unreliable for resynthesis, it remains useful for general feature extraction. Calculating frequency, decibel level, and spectral flux remain adequately accurate despite being asynchronous from the system's core dsp. For *Sinew*'s audio input, frequency and decibel level are calculated using existing open source C# libraries[57]. The implementation for calculating spectral flux is novel to this project.

---

[56] Technologies, "Unity - Scripting API."

[57] "Pitch Detection - C# Library - 3y3.Net."

3.3.3.2. Low-level Features: Spectral Flux in Unity

Sinew introduces a simple method for calculating spectral flux natively in C# within the Unity monobehaviour framework. I offer this as an alternative to bulkier solutions that require proprietary libraries for individual target platforms:

```
float   previousData;
float   currentFlux;

void CalculateFlux()
{
       float[] spectrumData =
              audioSource.GetSpectrumData(
                     spectrumData,
                     channel,
                     FFTWindow.BlackmanHarris);

       float[] variation =
              new float[spectrumData.Length];

       for (int i = 0; i < spectrumData.Length; i++)
       {

              variation[i] =
                     Mathf.Abs(
                            spectrumData[i] - previousData[i]);

       }

       currentFlux = variation.Sum();

       previousData = spectrumData;

}
```

This method retrieves spectrum data through `GetSpectrumData`. It then calculates frequency variations by taking the absolute value of the difference of the current and previous array values. These values are summed using the LINQ method extension `Sum()` for simplicity. Overall, the process is computationally inexpensive. Specific to *Sinew*, accessing the same spectrum data used for calculating pitch detection and decibel level further optimizes the function.

3.3.3.3. Low-level Features: Smoothing and Scaling

Each low-level feature exists within a wide range of values as seen in Table 13. Added to this group of features is the unique pitch count—the amount of entries in a list of unique pitches detected over the previous ten seconds. These values are analyzed as MIDI values received from open source pitch detection libraries[58]. In order to design meaningful derivations through various combinations of these features, it becomes necessary to scale them. Using the minimum and maximum values shown in Table 13, each data stream is scaled and clamped between zero and one.

Table 13 A summary of value ranges for *Sinew*'s extracted low-level audio features.

Feature	Minimum Value	Maximum Value
Frequency	140	1400
Decibel level	-30	6
Spectral flux	0	.001
Unique pitch count	0	13

Smoothing is applied to the low-level data streams by averaging all values received within a given timeframe. For *Sinew*, this time frame is roughly two seconds. Processing low-level audio features in the Unity scripting environment requires a time-centric approach to the smoothing function. This is because different hardware will achieve varying frame rates. One performer's computer may produce sixty low-level values per second while another may produce upwards of two hundred. *Sinew* introduces a time-centric C# class for smoothing low-level feature values while accounting for disparate frame rates:

```
public class TimedAverage
{
    public List<float> times;
    public List<float> values;

    public float timeLimit;

    public TimedAverage(float _timeLimit)
    {
```

[58] "Pitch Detection - C# Library - 3y3.Net."

```
            UpdateTimeLimit(_timeLimit);
            times = new List<float>();
            values = new List<float>();
        }

        public void UpdateTimeLimit(float _timeLimit)
        {
            timeLimit = _timeLimit;
        }

        public float AddValue(float _value)
        {
            if (times.Count > 0
                    && (Time.time - times[0]) > timeLimit)
            {
                values.RemoveAt(0);
                times.RemoveAt(0);
            }

            values.Add(_value);
            times.Add(Time.time);

            return values.Average();
        }
    }
```

The time limit for `TimedAverage` can be set using the class constructor. Its primary function—`AddValue()`—returns the average value of all values received within the designated limit. I process all of *Sinew*'s low-level features through the `TimedAverage` class before using them to derive high-level features. Use of a type integer variant in *Sinew*'s final build further optimizes this class. I use the integer variant for unique pitch count, which also has a longer time limit of roughly five seconds.

Smoothing adds a base level of resistance for *Sinew*'s feature logic. Without this resistance, the computer's perception of each feature becomes noisy and less usable for my purposes. Smoothing helps the computer perceive more data over time, each point having a longer impact within the data stream. I do this to reduce the unrelatable perfection of computation, and to more closely mimic natural processes. By including resistance in the feature logic, I encourage a tighter coupling between improvisers and *Sinew*'s environment.

Recall the discussion of Zorn's *Cobra* from section 2.1.1. In *Cobra*, very little is done to prevent an improviser from dominating the performance if they so choose.[59] The balance between affordances, actions, and consequences seems to be largely out of the composer's control. The structures that emerge are volatile. To manage potential chaos within a self-regulating performance system, I establish greater resistance between the forces that drive state changes. Mechanisms within the system provide momentum and the pooling of energy. The relationships I build between parameters are heavy and stable, mimicking the enormous force of many natural processes. A performer's influence is dampened and requires persistence. The software does not include discrete abstractions such as toggles, switches, or buttons. As a result, my systems do not often exhibit discrete state changes. Any events in my work that appear discrete are still implemented within a continuous structure. The overall result is a performance experience that is weighted by resistant forces.

3.3.3.4. High-level Features

The high-level features derived through *Sinew*'s feature logic are used to communicate with other logics. These features are steadiness, intensity, and legato. Each of these values come from relationships between the logic's low-level features. These relationships are shown in Table 14 in the form of equations. Note that all features written in Table 14 are scaled between zero and one using the ranges shown in Table 13. Intensity and legato are straightforward derivations of their implied meaning. This is not the case for calculating steadiness, a highly nuanced quality of performance.

[59] Van Der Schyff, "The Free Improvisation Game."

Table 14 Equations demonstrating *Sinew*'s high-level features as derived from low-level features.

Feature	Equation
Steadiness	$Max\left(0, \left(Decibel\ level\ - \frac{Spectral\ flux + Unique\ pitch\ count}{2}\right)\right)$
Intensity	$Decibel\ level * \frac{(Frequency + Spectral\ flux)}{2}$
Legato	$Decibel\ level * Unique\ pitch\ count$

Designing an equation that expresses a level of steadiness proves difficult. Steadiness describes levels of perceived variation within the performer's musical behaviors. For *Sinew*'s feature logic, this was ultimately calculated using two main low-level features: spectral flux and unique pitch count. When these values are low, the value for stability needs to increase. Inverting the value appears to be a valid solution, knowing that all feature values are clamped between one and zero. However, the system must take into account performer silence. Thus, stability is derived as the difference between decibel level and the average value of flux and pitch count. If the performer's scaled decibel value is 0.5, and the flux-pitch average is 0.1, steadiness has a value of 0.4. The relationship between decibel level and the flux-pitch average favors louder stability. Quiet sustains will have lower steadiness values than loud sustains. To balance this bias, I process the resulting steadiness value through a sigmoid function (see 3.3.3.5 Mapping Tool: The Sigmoid Function on page 89).

3.3.3.5. Mapping Tool: The Sigmoid Function

In the early stages of *Sinew*'s development, it became very clear that the methods frequently used in a scripting environment were incompatible with the project's core principles. Programming in C# relies on several discrete tools—most commonly being *if-then* and *switch*

statements. I say these tools are incompatible because they often create forks in the programming logic. Under such circumstances, a value must be processed through one path or another with no options for in-between. Instead, *Sinew* requires tools that enable data streams to flow freely between multidimensional states.

The sigmoid function is highly compatible with *Sinew*'s ideals for continuousness. It is a hyperbolic, logistic function that is able to constrain values to a known range without the need for limits or comparator checks. The function is written in Equation 1 and graphed in Figure 29 where $k = 1$, $a = 0$, and $b = -5$.

$$\left(\frac{k}{(1 + (e)^{(a+bx)})}\right)$$

Equation 1 Sigmoid function

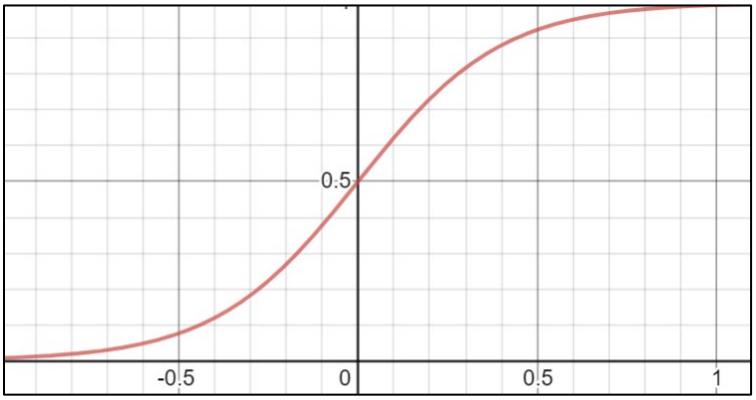

Figure 29 Graph of a basic sigmoid function.

The sigmoid function replaces discrete statements in many of *Sinew*'s systems, one of which is its flocking behaviors. A basic flocking algorithm contains three main subsystems: separation, alignment, and cohesion.[60] Early implementations of *Sinew*'s flocking algorithm gated the influence of these three subsystems using if-then statements. For instance, if two

[60] Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model."

objects were at a distance of less than one unit from each other, only then would the separation subsystem influence those objects' final positions. The sigmoid function changes this model from gated subsystems to continuous processes. Rather than using the if-then statements, the sigmoid function allows the flocking logic to run continuously. Every flocking object is continuously affected by all three subsystems, resulting in more natural behaviors.

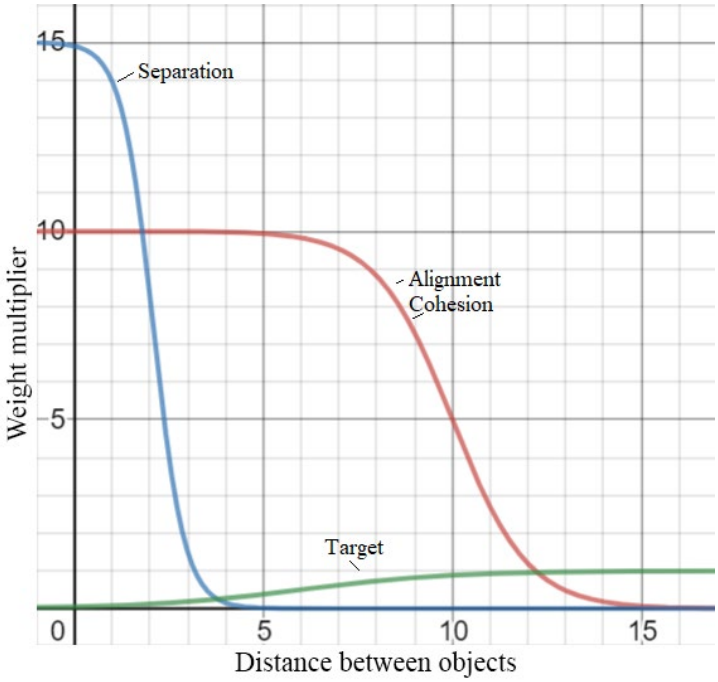

Figure 30 Graph of sigmoid functions used in Sinew's flocking algorithm.

Table 15 Coefficients for sigmoid functions used in *Sinew*'s flocking algorithm

Subsystem	*a*	*b*	*k*
Separation	-5	2.4	20
Alignment	-10	1	10
Cohesion	-10	1	10
Target	3	-.5	1

Figure 30 shows the s-curves of three functions. These curves demonstrate the amount of weight each subsystem has on the position and rotation of a flocking object. Table 15 contains the coefficients for each of the functions (see Equation 1 Sigmoid function).

Consider the separation curve in Figure 30. As an object's distance from other objects decreases, the separation subsystem's weight increases. This results in a greater repellent force between the objects. As the distance diminishes to less than two units, the repellent force has a greater weight than the aligning and cohesive forces. Notice that each force has a continuous influence on the objects, but shift in weight according to their respective positions. This method is highly compatible with the goals of a self-regulating performance system, and is not exclusive to only the flocking algorithms. I use multiple parallel sigmoid functions are form dynamic mappings between performer behaviors and the performance environment.

3.3.4. Sinew: Analysis

Sinew exhibits all of the key characteristics of a self-regulating performance system. In doing so, it helps to resolve the conflict between improvisational freedom, composer authorship, and a sense of shared purpose. However, as expressed in section 3.2.5, I am in some ways dissatisfied with the patency of some of these characteristics. This section will analyze *Sinew* using the key characteristics of a self-regulating performance system, starting with the points I see as most successful. Following these points will be a discussion of what can be learned by its less successful points.

Sinew avoids conveying explicit expectations. The most explicitly expressed instruction given to performers is the on-screen text of an ambiguous role. The ambiguity of these roles is akin to the starting conditions that enable horizontal operational logic. They provide intrinsic motivation for the performer. They do not qualify as explicit instructions. All other expectations,

as previously described, take the form of emergent environmental interactions. This avenue for communication is in line with the explored concepts of enactive cognition and emergence.

Sinew is intrinsically continuous. In a programming environment, it is difficult to construct a highly continuous system. The development of *Sinew* resulted in many tools and techniques for facilitating continuous data streams. There are circumstances that require discrete programming techniques. However, instances of this were made continuous by the dynamic blending of discrete elements.

Sinew is underwritten by horizontal operational logics. All value transfers between objects are continuous, unscripted occurrences. Tools, like the sigmoid function, allow for the horizontal interactions of objects, including the human performer. Observing the programming logic will not reveal the interactions that take place, because none are scripted. Instead, *Sinew*'s logics, including the mappings between primary logics, are entirely horizontal.

Sinew expresses my intended interactions through cultivated emergence. Emergence in this work is generated in the relational domain between performer interactions and self-regulating objects. As the composer, I was able to embed intended ranges of interactions by delicately crafting the mappings between operational logics. The improvisational behaviors of human performers are informed by the interactions of my design. Through the emergence I have cultivated, I successfully expressed my intended interactions. This leads me to ask, "Why is my sense of composer authorship slightly lacking, despite checking all of the boxes for a self-regulating performance system?"

Analysis of the work using the lens established in Chapter 2 reveals that *Sinew*'s timescale is the primary contributor toward a diminished sense of composer authorship. Many of the intended interactions can take place over a long period. A significant length of time for one

interaction can result in incongruous perceptions of moment-to-moment improvisations and the overall gestures that contain them. For example, consider when a performer is focused on interactions with the well and its spheroids. Performers can playfully interact with the transfer of energy, visually indicated by lights flowing from the performer to the well. The well will eventually produce a spheroid that takes on the performative characteristics of a favored performer. As the spheroid occasionally displays a loss of energy, the performer is incentivized to assist by improvising its preferred aesthetic. Overall, this interaction takes place over a five- to fifteen-minute period. This long stretch of time diminishes the patency of its intended relevance to musical structures. From the perspective of the system-design, I retain a strong sense of composer authorship. However, from the perspective of realized interactions, my sense of composer authorship is obscured.

Another unintended consequence of Sinew's stretched timescale is the smearing of interaction domains. For example, the interaction described above eventually overlaps itself as more spheroids are brought into the scene. Additionally, these interactions persist as others are introduced—for instance, interactions with bulbs and their spawn. The overall interactivity of the scene can at times become busy and unfocussed. As a result of these two main issues—a timescale that is too broad, and smearing interactions—a performance of *Sinew* appears to favor improvisational freedom over composer authorship.

Related to the issue of timescale is *Sinew*'s potential to foster a sense of shared purpose. *Sinew* implicitly conveys cause and effect. When performing *Sinew*, an improviser is able to slowly and intuitively discover their general impact. Particle systems, shaders, and performance logics were implemented to express the attitude of every interaction. As performers observe each object's behavior, they are able to intuit the impact of their interactions. Consequently,

performers and observers alike can derive meaning from the same underlying causalities. The understanding of these causalities are not as strong as they might be with a more manageable timescale—a smaller timescale. So, in addition to a diminished sense of composer authorship, *Sinew*'s large timescale lessens the potential for a sense of shared purpose. My experience following the work's premiere was similar to that of *Ecosystem One*, where individuals contacted me with their interpretations. This indicated that *Sinew* successfully exhibited an underlying purpose. The interpretations offered to me indicated that the display of cause and effect was not as clear as my previous work.

It is important to note that these issues are not unique to self-regulating performance systems. In general, the timescale and scope of musical gestures is a common element of composition as a craft. What *Sinew*'s diminished sense of shared purpose and lessened composer authorship demonstrates is that the self-regulating performance system and its accompanying paradigms are not an alternative to composition. Rather, effective implementation of a self-regulating performance system entails using well-established principles of music composition.

Chapter 4.

Concluding Thoughts

In this text, I outlined a conflict between improvisational freedom and composer authorship. I also reviewed how a sense of shared purpose links to this conflict. I addressed this conflict by presenting theories of enactive cognition and concepts surrounding emergence. Enactive cognition aids in validating the environment as an appropriate location for composer authorship. I present emergence as the mechanism by which a composer can unobtrusively express their intent to an improviser. I showed these two theoretical tools to be a foundation for a creative work that gives equal weight to its improvisational and compositional components. Additionally, they provided a pragmatic context for discussing composer intended relevance. Collectively, these two topics informed the technological concepts explored in this paper.

I reviewed relevant technological concepts that aid in the reconciliation of improvisation and composition. I described how continuous processes are an essential factor in relocating composer authorship. Concepts surrounding audiovisual elements progressed the exploration of pragmatic tools for such relocation. Importantly, I present operational logics as a vital tool for designing, implementing, and analyzing interactive software. I show how the orientation of these logics affect user interactions and elements of emergence. Specifically, horizontal orientations of operational logics proved to be a necessary tool for crafting emergent interactions.

Following discussions of research and literature, I presented the *self-regulating performance system*. I offer the model for this system as a practical implementation of the previously established concepts. The model's key characteristics are:

1. Expresses the composer's intended interactions through cultivated emergence.
2. Avoids conveying explicit expectations.
3. Is intrinsically continuous.

4. Implicitly conveys cause and effect.
5. Is underwritten by horizontal operational logics.

I presented two substantial compositions as the creative activity portion of this research. They are *Ecosystem One* and *Sinew.* Both works follow the established model of a self-regulating performance system. In keeping to this model, they attempted to maximize improvisational freedom and composer authorship while promoting a shared sense of purpose. I explained useful tools and processes that outline successful implementation of the performance model. I catalogued the prominent operational logics of these works, and showed that they exhibit a horizontal orientation. Additionally, both works demonstrated how to better maximize improvisational freedom, composer authorship, and a sense of shared purpose. Following an analysis of both works, I concluded that the self-regulating performance model should be implemented while maintaining well-established principles for composition.

Moving forward, I intend to continue the use of tools and concepts found in this paper. I believe modern technologies have required composers to rethink current practices. Specifically, I intend on future research and creative works investigating remote performance. This current project uses Unity as both a dedicated visual renderer, and a hybrid audiovisual performance environment. At the time of this writing, Unity is developing new networking tools for online interaction. I am particularly interested in the idea of using these tools to build a persistent online environment for music improvisation. The concepts laid down in this paper provide the necessary theoretical framework for such a project.

In conclusion, I am pleased with the outcome of this research and its creative activity. I reiterate that the debate surrounding the relationship between composition and improvisation will likely never resolve. My approach to the conversation is in the spirit of progression, and with the

hope that my writing and my music will aid the efforts of another creator. I offer this research in the hope that it challenges readers with new ideas and motivates the creation of new things.

References

Alperson, Philip. “A Topography of Improvisation.” *The Journal of Aesthetics and Art Criticism* 68, no. 3 (August 8, 2020): 273–80.

Anderson, Christine. “Dynamic Networks of Sonic Interactions: An Interview with Agostino Di Scipio.” *Computer Music Journal* 29, no. 3 (September 2005): 11–28. https://doi.org/10.1162/0148926054798142.

Anthony, Kevin. “Recourse | Kevin Anthony.” Performance, September 6, 2019. https://www.youtube.com/watch?v=ATO7UFYFkN0.

Armstrong, Newton. *An Enactive Approach to Digital Musical Instrument Design Theory, Models, Techniques*. Saarbrücken: AV Akademikerverlag, 2012. http://nbn-resolving.de/urn:nbn:de:101:1-201205252060.

Bailey, Derek. *Improvisation: Its Nature and Practice in Music*. New York: Da Capo Press, 1993.

Bedau, Mark A. “Weak Emergence.” In *Mind, Causation, and World*, Vol. 11. Philosophical Perspectives. Oxford, Eng.; New York, N.Y.: B. Blackwell, 1997.

Minecraft Wiki. “Bedrock Edition,” n.d. https://minecraft.gamepedia.com/Bedrock_Edition.

Bello, Juan Pablo. “Low-Level Features and Timbre,” n.d., 31.

Berio, Luciano, Rossana Dalmonte, Bálint András Varga, David Osmond-Smith, and Luciano Berio. *Two Interviews*. New York: M. Boyars, 1985.

Bertolani, Valentina, and Friedemann Sallis. “Live Electronic Music.” In *Routledge Encyclopedia of Modernism*, 1st ed. London: Routledge, 2016. https://doi.org/10.4324/9781135000356-REM577-1.

Blackwell, T.M., and P. Bentley. “Improvised Music with Swarms.” In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No.02TH8600)*, 2:1462–67. Honolulu, HI, USA: IEEE, 2002. https://doi.org/10.1109/CEC.2002.1004458.

“Blackwell.Pdf.” Accessed July 31, 2020. http://axon.cs.byu.edu/~dan/673/papers/blackwell.pdf.

blender.org. “Blender - Free and Open 3D Creation Software,” n.d. https://www.blender.org/.

Blum, Stephen. “Representations of Music Making.” In *Musical Improvisation: Art, Education, and Society*, edited by Gabriel Solis and Bruno Nettl, 239–62. Urbana: University of Illinois Press, 2009.

Borio, Gianmario, ed. "'Live Is Dead?': Some Remarks about Live Electronics Practice and Listening." In *Musical Listening in the Age of Technological Reproduction*. Musical Cultures of the Twentieth Century. Burlington: Ashgate, 2015.

———, ed. *Musical Listening in the Age of Technological Reproduction*. Musical Cultures of the Twentieth Century. Burlington: Ashgate, 2015.

Chadabe, Joel. "Interactive Composing: An Overview." *Computer Music Journal* 8, no. 1 (1984): 22. https://doi.org/10.2307/3679894.

Chalmers, David J. "Strong and Weak Emergence." In *The Re-Emergence of Emergence*, 244–54. Oxford University Press, 2008. https://doi.org/10.1093/acprof:oso/9780199544318.001.0001.

Chiang, Helen. "Minecraft: Connecting More Players Than Ever Before." *Xbox Wire* (blog), May 18, 2020. https://news.xbox.com/en-us/2020/05/18/minecraft-connecting-more-players-than-ever-before/.

Clark, Andy, and David J. Chalmers. "The Extended Mind." In *The Extended Mind*, edited by Richard Menary, 26–42. The MIT Press, 2010. https://doi.org/10.7551/mitpress/9780262014038.003.0002.

Clarke, Eric F., and Mark Doffman, eds. *Distributed Creativity*. Vol. 1. Oxford University Press, 2017. https://doi.org/10.1093/oso/9780199355914.001.0001.

Collins, Karen, Bill Kapralos, Holly Tessler, and Tim van Geelen. "New Tools for Interactive Audio, and What Good They Do." In *The Oxford Handbook of Interactive Audio*, edited by Karen Collins, Bill Kapralos, and Holly Tessler. Oxford University Press, 2014. https://doi.org/10.1093/oxfordhb/9780199797226.013.033.

Collins, Nick, Margaret Schedel, and Scott Wilson. *Electronic Music*. 1. publ. Cambridge Introductions to Music. Cambridge: University Press, 2014.

Conrad, Michael, ed. "Discrete and Continuous Processes in Computers and Brains." In *Physics and Mathematics of the Nervous System: Proceedings of a Summer School Organized by the International Centre for Theoretical Physics, Trieste, ... Held at Trieste, August 21 - 31, 1973*. Lecture Notes in Biomathematics 4. Berlin: Springer, 1974.

"COVID-19: Housing, Student Life, and Commencement Updates | UCI." Accessed July 30, 2020. https://uci.edu/coronavirus/messages/200313-student-life-updates.php.

Johns Hopkins Coronavirus Resource Center. "COVID-19 Map." Accessed July 30, 2020. https://coronavirus.jhu.edu/map.html.

ReliefWeb. "Covid-19 Situation Report #9, March 16, 2020 - World," n.d. https://reliefweb.int/report/world/covid-19-situation-report-9-march-16-2020.

Cross, Lowell. "*Reunion* : John Cage, Marcel Duchamp, Electronic Music and Chess." *Leonardo Music Journal* 9 (December 1999): 35–41. https://doi.org/10.1162/096112199750316785.

Dannenberg, Roger B. "Interactive Visual Music: A Personal Perspective." *Computer Music Journal* 29, no. 4 (December 2005): 25–35. https://doi.org/10.1162/014892605775179964.

De Benedictis, Angela Ida. "Authorship and Performance Tradition in the Age of Technology." In *Live Electronic Music: Composition, Performance, Study*, 195–216. Routledge Research in Music. Abingdon, Oxon ; New York, NY: Routledge, 2018.

Di Scipio, Agostino. "Dwelling in a Field of Sonic Relationships." In *Live Electronic Music: Composition, Performance, Study*, 17–45. Routledge Research in Music. Abingdon, Oxon ; New York, NY: Routledge, 2018.

———. "Sound Is the Interface: From *Interactive* to *Ecosystemic* Signal Processing." *Organised Sound* 8, no. 3 (December 2003): 269–77. https://doi.org/10.1017/S1355771803000244.

Doornbusch, Paul. "A Brief Survey of Mapping in Algorithmic Composition." *ICMC*, 2002.

Dreyfus, Stuart E. "The Five-Stage Model of Adult Skill Acquisition." *Bulletin of Science, Technology & Society* 24, no. 3 (June 2004): 177–81. https://doi.org/10.1177/0270467604264992.

Duncan, Sean C. "Minecraft, Beyond Construction and Survival," October 23, 2019. https://doi.org/10.1184/R1/10029221.v1.

Emmerson, Simon, and Denis Smalley. *Electro-Acoustic Music*. Vol. 1. Oxford University Press, 2001. https://doi.org/10.1093/gmo/9781561592630.article.08695.

Eno, Brian. *A Year with Swollen Appendices*. London: Faber and Faber, 1996.

Foucault, Michel. "What Is an Author?" In *Language, Counter-Memory, Practice: Selected Essays and Interviews*, edited by Donald F. Bouchard and Sherry Simon, 1. printing, Cornell paperbacks, [Nachdr.]. Cornell Paperbacks. Ithaca, NY: Cornell Univ. Press, 1980.

Fox, Christopher. *New Complexity*. Vol. 1. Oxford University Press, 2001. https://doi.org/10.1093/gmo/9781561592630.article.51676.

Fujibayashi, Hidemaro, Satoru Takizawa, and Takuhiro Dohta. "Breaking Conventions with The Legend of Zelda: Breath of the Wild." Game Developers Conference, March 10, 2017. https://www.youtube.com/watch?v=QyMsF31NdNc.

Gardner, Martin. "Mathematical Games." *Scientific American* 223, no. 4 (October 1970): 120–23. https://doi.org/10.1038/scientificamerican1070-120.

George Lewis and Vijay Iyer in Concert. Stream, 2012. https://www.youtube.com/watch?v=IBPJ2HAmsc8.

GoodTimesWithScar. *My ULTIMATE Hermitcraft Season 6 Tribute*. GoodTimesWithScar, 2020. https://www.youtube.com/watch?v=GQpAJBUGW9Y.

Green, Owen. "Audible Ecosystemics as Artefactual Assemblages: Thoughts on Making and Knowing Prompted by Practical Investigation of Di Scipio's Work." *Contemporary Music Review* 33, no. 1 (January 2, 2014): 59–70. https://doi.org/10.1080/07494467.2014.906698.

Hill, Peter. "Xenakis and the Performer." *Tempo*, no. 112 (March 1975): 17–22. https://doi.org/10.1017/S0040298200018830.

Johnson, Steven. *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. New York: Scribner, 2001.

Key, Susan, Larry Rothe, Michael Tilson Thomas, and San Francisco Symphony Orchestra, eds. *American Mavericks*. San Francisco, Calif. : Berkeley, Calif: San Francisco Symphony ; Published in cooperation with the University of California Press, 2001.

Kimura, Mari. "MUGIC Sensor." MARI KIMURA - violinist/composer, n.d. http://www.marikimura.com/mugic-sensor.html.

Krishnamurthy, Kapil. "Generation of Control Signals Using Pitch and Onset Detection for an Unprocessed Guitar Signal," n.d., 5.

Lewis, George E. "Too Many Notes: Computers, Complexity and Culture in *Voyager*." *Leonardo Music Journal* 10 (December 2000): 33–39. https://doi.org/10.1162/096112100570585.

Linson, Adam, and Eric F. Clarke. "Distributed Cognition, Ecological Theory and Group Improvisation." In *Distributed Creativity: Collaboration and Improvisation in Contemporary Music*, edited by Eric F. Clarke and Mark Doffman. Studies in Musical Performance as Creative Practice 2. New York, NY: Oxford University Press, 2017.

LPG. *Minecraft: Redstone Computer (V1.0) (Interactive PC, Calculator, Day/Night Controller)*. LPG, 2013. https://www.youtube.com/watch?v=GQpAJBUGW9Y.

Mateas, Michael, and Noah Wardrip-Fruin. "Defining Operational Logics." *UC Santa Cruz*, June 18, 2018. https://escholarship.org/uc/item/3cv133pn.

Mayya, Shreemathi S., Ashma D. Monteiro, and Sachit Ganapathy. "Types of Biological Variables." *Journal of Thoracic Disease* 9, no. 6 (June 2017): 1730–33. https://doi.org/10.21037/jtd.2017.05.75.

Mazierska, Ewa. “Improvisation in Electronic Music—The Case of Vienna Electronica.” *Open Cultural Studies* 2, no. 1 (December 1, 2018): 553–61. https://doi.org/10.1515/culture-2018-0050.

McCormack, Jonathan, Alice Eldridge, Alan Dorin, and Peter McIlwain. *Generative Algorithms for Making Music: Emergence, Evolution, and Ecosystems*. Oxford University Press, 2011. https://doi.org/10.1093/oxfordhb/9780199792030.013.0018.

McDonough, Richard, ed. “Emergence and Creativity: Five Degrees of Freedom.” In *Creativity, Cognition, and Knowledge: An Interaction*, 284–320. Perspectives on Cognitive Science. Westport, Conn: Praeger, 2002.

Meric, Renaud, and Makis Solomos. “Analysing Audible Ecosystems and Emergent Sound Structures in Di Scipio’s Music.” *Contemporary Music Review* 33, no. 1 (January 2, 2014): 4–17. https://doi.org/10.1080/07494467.2014.906690.

YouTube. “Michael Mateas on Façade, Part 1.” Video Streaming Service, August 25, 2013. https://www.youtube.com/watch?v=LRUvHV7_fXk.

Monson, Ingrid T. *Saying Something: Jazz Improvisation and Interaction*. Chicago Studies in Ethnomusicology. Chicago: University of Chicago Press, 1996.

Mooney, Chris. “Multiplicative Design.” *Goblin Artisans* (blog), August 27, 2018. http://goblinartisans.blogspot.com/2018/08/multiplicative-design.html.

O’Connor, Timothy. “Emergent Properties.” *American Philosophical Quarterly* 31, no. 2 (1994): 91–104.

Osborn, Joseph C., Noah Wardrip-Fruin, and Michael Mateas. “Refining Operational Logics.” In *Proceedings of the International Conference on the Foundations of Digital Games - FDG ’17*, 1–10. Hyannis, Massachusetts: ACM Press, 2017. https://doi.org/10.1145/3102071.3102107.

Parr, Thomas, and Karl J. Friston. “The Discrete and Continuous Brain: From Decisions to Movement-And Back Again.” *Neural Computation* 30, no. 9 (2018): 2319–47. https://doi.org/10.1162/neco_a_01102.

Payling, Dave. “Approaches to Composition in Visual Music: An Artist’s Reflection on Three Original Pieces.” *Leonardo Music Journal* 29 (December 2019): 62–66. https://doi.org/10.1162/lmj_a_01065.

Pitch Detection - C# Library. “Pitch Detection - C# Library - 3y3.Net,” n.d. http://www.3y3.net/.

Reid, Gavin. “Motivation in Video Games: A Literature Review.” *The Computer Games Journal* 1, no. 2 (November 2012): 70–81. https://doi.org/10.1007/BF03395967.

Reynolds, Craig W. “Flocks, Herds, and Schools: A Distributed Behavioral Model.” *Computer Graphics* 21, no. 4 (July 1987): 25–34.

Roads, Curtis. *Microsound*. 1. pbk. ed. Cambridge, Mass.: MIT Press, 2004.

———. "The Second STEIM Symposium on Interactive Composition in Live Electronic Music." *Computer Music Journal* 10, no. 2 (1986): 44. https://doi.org/10.2307/3679484.

Rudi, Jøran, and Neal Spowage. "Editorial: Sound and Kinetics – Performance, Artistic Aims and Techniques in Electroacoustic Music and Sound Art." *Organised Sound* 23, no. 3 (December 2018): 219–24. https://doi.org/10.1017/S1355771818000122.

Ryan, Kevin, and Andrea Schiavio. "Extended Musicking, Extended Mind, Extended Agency. Notes on the Third Wave." *New Ideas in Psychology* 55 (December 2019): 8–17. https://doi.org/10.1016/j.newideapsych.2019.03.001.

Sallis, Friedmann, Valentina Bertolani, Jan Burle, and Laura Zattra, eds. *LIVE ELECTRONIC MUSIC: Composition, Performance, Study.* Place of publication not identified: ROUTLEDGE, 2019.

Schiff, Joel L. *Cellular Automata A Discrete View of the World*. New York, NY: John Wiley & Sons, 2011. http://nbn-resolving.de/urn:nbn:de:101:1-201412309226.

Schröder, Julia H. "Emergence and Emergency: Theoretical and Practical Considerations in Agostino Di Scipio's Works †." *Contemporary Music Review* 33, no. 1 (January 2, 2014): 31–45. https://doi.org/10.1080/07494467.2014.906722.

"Sol LeWitt: A Wall Drawing Retrospective | MASS MoCA." Accessed July 8, 2020. https://massmoca.org/event/sol-lewitt-a-wall-drawing-retrospective/.

Solomos, Makis. "Agostino Di Scipio: Audible Ecosystems." *Contemporary Music Review* 33, no. 1 (January 2, 2014): 2–3. https://doi.org/10.1080/07494467.2014.906673.

Spolin, Viola. *Improvisation for the Theater: A Handbook of Teaching and Directing Techniques*. 3rd ed. Evanston, Ill: Northwestern University Press, 1999.

Stout, David. "Performance." Noisefold, June 14, 2012. http://noisefold.com/performance.

Strickland, Edward. *Minimalism: Origins*. 2. [ed.], corr.rev. Version. Bloomington: Indiana Univ. Press, 2001.

Tahiroğlu, Koray, Juan Carlos Vasquez, and Johan Kildal. "Facilitating the Musician's Engagement with New Musical Interfaces: Counteractions in Music Performance." *Computer Music Journal* 41, no. 2 (June 2017): 69–82. https://doi.org/10.1162/COMJ_a_00413.

Technologies, Unity. "Unity - Scripting API: AudioSource.GetOutputData," n.d. https://docs.unity3d.com/ScriptReference/AudioSource.GetOutputData.html.

Thompson, Evan. "Introduction." In *The Embodied Mind: Cognitive Science and Human Experience*, 8. print. Cambridge, Mass.: MIT Press, 2000.

Thurlow, Jeremy. “Intervention On the Conundrum of Composing an Improvisation.” In *Distributed Creativity*, edited by Eric F. Clarke and Mark Doffman, Vol. 1. Oxford University Press, 2017. https://doi.org/10.1093/oso/9780199355914.001.0001.

Ulman, Erik. “Some Thoughts on the New Complexity.” *Perspectives of New Music* 32, no. 1 (1994): 202. https://doi.org/10.2307/833163.

Unemi, Tatsuo, and Daniel Bisig. “Playing Music by Conducting BOID Agents.” In *Artificial Life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, edited by Jordan Pollack, Mark A. Bedau, Phil Husbands, Richard A. Watson, and Takashi Ikegami. The MIT Press, 2004. https://doi.org/10.7551/mitpress/1429.001.0001.

Van Der Schyff, Dylan. “The Free Improvisation Game: Performing John Zorn’s Cobra.” *Journal of Research in Music Performance* 0, no. 0 (May 5, 2013). https://doi.org/10.21061/jrmp.v0i0.726.

Varela, Francisco J., Evan Thompson, and Eleanor Rosch. *The Embodied Mind: Cognitive Science and Human Experience*. 8. print. Cambridge, Mass.: MIT Press, 2000.

Wardrip-fruin, Noah, Michael Mateas, Steven Dow, and Serdar Sali. *Agency Reconsidered*, n.d.

Wilson, Robert A., and Lucia Foglia. “Embodied Cognition.” In *The Stanford Encyclopedia of Philosophy*, edited by Edward N. Zalta, Spring 2017. Metaphysics Research Lab, Stanford University, 2017. https://plato.stanford.edu/archives/spr2017/entries/embodied-cognition/.

Zattra, Laura. “Points of Time, Points in Time, Points in Space. Agostino Di Scipio’s Early Works (1987–2000).” *Contemporary Music Review* 33, no. 1 (January 2, 2014): 72–85. https://doi.org/10.1080/07494467.2014.906699.