SensorChimes: Musical Mapping for Sensor Networks toward Augmented Acoustic Ecosystems

by

Evan F. Lynch

Submitted to the

Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2016

Author_____

Department of Electrical Engineering and Computer Science
January 29, 2016

Certified by_____

Prof. Joseph A. Paradiso, Professor of Media Arts and Sciences, Thesis Supervisor
January 29, 2016

Accepted by_____

Dr. Christopher J. Terman, Chairman, Master of Engineering Thesis Committee

SensorChimes: Musical Mapping for Sensor Networks toward Augmented Acoustic Ecosystems

by

Evan F. Lynch

Submitted to the Department of Electrical Engineering and Computer Science

on January 29, 2016, in partial fulfillment of the

requirements for the degree of

Master of Engineering in Computer Science and Engineering

Abstract

SensorChimes aims to create a new canvas for artists leveraging ubiquitous sensing and data collection. The Tidmarsh Living Observatory Initiative, which is documenting the transformation of a reclaimed cranberry bog with a large-scale sensor deployment, provides an opportunity to explore data-driven musical composition based on large-scale environmental sensor networks.

This thesis presents a framework that facilitates musical mappings for such sensor networks. A library of C-externals called ChainFlow for the graphical programming language Max/MSP that provides an interface to real-time and historical data for large sensor deployments was designed and implemented.

This thesis envisions musical mapping for sensor networks as a tool for augmenting presence and telepresence in real and virtual worlds, by adding to the acoustic ecosystem. Physical processes are manifested as musical ideas rendering an ambient display. The ChainFlow library along with spatial audio techniques were used to create immersive musical compositions that are complemented by a graphical 3D virtual world. These works, driven by the sensor network deployed at Tidmarsh, are presented as case studies in augmented presence through musical mapping.

Thesis Supervisor: Joseph A. Paradiso
Title: Professor of Media Arts and Sciences

Acknowledgements

I would like to acknowledge the help and support of my advisor, Joe Paradiso, and the members of the Responsive Environments Group at the MIT Media Lab. Particularly, I thank Spencer Russell, Gershon Dublon, and Brian Mayton for helpful technical discussion and advice. I thank Nan Zhao and Juliana Cherston for their insights and thoughts about musical mapping and Jie Qi for insights about being a graduate student at the Media Lab. Additionally, I would like to thank composers Ricky Graham and Evan Ziporyn for jumping at the opportunity to collaborate and contributing to the core of this project. Thanks also to Daniel Manesh and Nick Joliat for conversations and thoughts. Finally, many thanks to all my friends and to my family, especially my mother who helped copy edit and listened to me complain about the structure of my writing.

Contents

List of Figures

Chapter 1

Introduction

The modern world is increasingly documented not only by our writing, recording, and collective memory, but by the many sensors that are embedded in ubiquitous devices. Modern sensor technology allows for efficient collection of these data at a large scale. Our ability as humans to use this wealth of information is constrained by the physical limitations of human sensory perception and the limitations of the interfaces that mediate it. This thesis focuses on how these data can be leveraged for new forms of musical composition, envisioning a synthesis of electronic music composition and ubiquitous sensing toward augmented acoustic ecosystems.

This thesis presents a composition framework consisting of several tools that integrate into existing systems with the dual goal of facilitating artistically meaningful interactive music and facilitating auditory display driven by environmental sensor networks. While the framework is general, a sensor network deployed in a wetland in southern Massachusetts called Tidmarsh is the impetus and focal point of the exploration. This thesis contributes both the composition framework and specific patterns of musical mapping demonstrated through case study works by collaborating composers both inspired and driven by Tidmarsh.

The main components of this framework are ChainFlow and DoppelMarsh. ChainFlow is an interface developed for this project that endeavors to make it easy to route any real-time or historical data from a sensor network to any point in the graphical programming environment Max/MSP,

allowing for quick realization of mapping ideas with a minimum of work and expertise.[1] Doppel-Marsh is a virtual replica of Tidmarsh created with the Unity game engine as part of a related project [24]. In the musical works presented in this paper, the virtual environment is leveraged to render an immersive virtual exploration of the wetland with spatialized musical mappings.

1.1 Musical Mapping Composition Framework

Musical mapping is the process of designing and implementing the relationship between a non-audio input signal or collection of data and a musical audio output. Much work has been done on the problem of musical mapping in the realm of instrument design for performance. With an electronic instrument, unlike an acoustic instrument, the input device which the musician plays is decoupled from the mechanism for producing sound. The connection between the two can be any programmable mapping. The input can be mapped linearly or non-linearly to the fundamental parameters of sound such as pitch, volume, and timbre, or to more abstract musical parameters such as rhythm, tempo, affect, style, harmony, etc. This project is concerned with a slightly different kind of musical mapping where the input is collected from the environment rather than produced by a performer. The goal of mapping for a performance instrument is generally to recreate the performer's conceived gesture and expression with sound [7]. In contrast, environmental data lacks a performer's expression and conception, so the goal is now to skirt the boundary between "musical mapping" and what might be better termed "data sonification"—to generate sound that renders data as a kind of audio "visualization" or "auditory display." Along this boundary, a musical idea is created via the elucidation of a physical or environmental process through sound, a compositional technique which composer Polansky calls 'manifestation' [27].

One of the great challenges with a project aiming to create a composition framework is determining the correct scope and the right constraints. Which constraints on the composer's creative control are liberating and which stifling? What interface is developed and directed enough to be a meaningful step up from starting with nothing while at the same time general enough to not be found irrelevant and discarded when the creative process takes an unexpected turn? These are the questions the

[1] Max/MSP: https://cycling74.com/products/max/

designs in this thesis are motivated by and evaluated with.

1.2 Augmented Acoustic Ecosystems

When an observer enters a space, some aspects of their environment are obvious—sound is audible, reflected light is visible, a light breeze is palpable. However, many phenomena remain imperceptible because we do not have appropriate biological sensors to detect them, they are too large or small, they change on timescales that are too long or short, or they are beyond our reach. How "present" we are in an environment relates not to how much we know abstractly about the environment but to how much we feel about the environment. This project aims to augment our "presence" by providing additional information about the environment through the acoustic medium, expanding the scope of information that we can readily and immediately intuit.

The windchime, a prehistoric wind sensor that makes music, inspires this project. Windchimes are augmentations to the acoustic environment that mechanically couple wind speed and direction to sound. A "composer" decides what material to make the chime out of, which pitches the chime should play, and the geometry of the chimes, crafting a space of musical expression that the wind will then actuate. The basic envelope and structure of local wind can be gleaned from the music made by the chimes. This project reimagines, generalizes, and augments this concept in the digital domain with electronic sensors that measure many parameters, electronic music composition, and virtual reality.

1.3 Tidmarsh Farms and the Living Observatory

Tidmarsh is a 577 acre wetland restoration project in southern Massachusetts, the largest current restoration project in the state, into which we have installed a large environmental sensor network as part of the Living Observatory Initiative [24]. It was a cranberry bog until 2010, when the restoration of its wild ecology began. The Living Observatory Initiative aims to document ecological processes in the wetland as it undergoes a ten year restoration process. The scale of the Tidmarsh sensor deployment is large, with many sensors distributed across a large area. It represents a

testing ground for ideas that require this kind of rich data that will be available everywhere in the near future, and is the focal point of a variety of ongoing investigations ranging from data visualization and augmented reality, to innovative ways of studying environmental processes and change [8]. While the landscape is reshaped and the wild ecology restored, sensors monitor the transformation.

The sensor network at Tidmarsh consists of dozens of battery powered nodes designed by Responsive Environments RA Brian Mayton, which form a low-power mesh network built on the IEEE 802.15.4 specification [24]. Each device measures environmental parameters including temperature, humidity, illumination, and pressure, and is designed to be extensible so that other sensors of interest (such as soil moisture, wind, gas levels, etc.) can be easily added. Live audio is also streamed from various sites in the wetland. The data are relayed to a base station via a wireless mesh network, where they are uploaded to a remote server. From this server, historical data can be browsed, and the live data can be streamed as they are collected. Sensor updates arrive on approximately thirty second intervals to keep the energy usage of the battery-powered nodes at a minimum.

With descriptions, it is difficult to capture the essence of the space which is palpable to a visitor. The site is extremely dynamic across seasons. In the morning in spring, a chorus of birds begin chirping, and in the evening, the frogs take over from the birds. On a bright summer morning, the air is crisp and smells of wet earth, and there is an overwhelming sense of greenness and life. Recording everything that is happening in a place like Tidmarsh is still the realm of science fiction, but the dense sensor network at Tidmarsh is step forward.

1.4 Project Overview

SensorChimes integrates several existing tools and systems and a novel composition interface to experiment with musical mapping for the network of environmental sensor nodes deployed at Tidmarsh. Some of these components are specific to this site, such as the deployment of sensor nodes themselves, while other components could be reused for other sites or applications. The "product" of this integration is an application that runs on a client machine that renders both a graphical

and sonic display.

The first main contribution is a low-level and general interface to sensor network data for Max/MSP called ChainFlow described in Chapter 3. ChainFlow provides an interface to real-time and historical data for each device on the site, as well as basic analysis of the spatial variation of the parameters measured at the site. ChainFlow and Max/MSP together form a powerful musical mapping interface, the design of which is discussed in depth. However, this thesis focuses on a few specific designs that were implemented using ChainFlow.

A system was created to render an immersive sonic environment with each sensor device at Tidmarsh generating an audio voice. The implementation of this virtual sonic environment is discussed in Chapter 4. In summary, a different musical audio signal is generated for each sensor device based on the environmental conditions local to that device together with aggregate features calculated across groups of devices. Then, all of these signals are delivered to the listener simultaneously, each spatialized to sound as if emanating from the devices themselves. The listener moves around the virtual space and explores the different signals and by consequence the underlying environment.

The immersive sonic environment discussed in Chapter 4 is only one layer of the mapping; it leaves the all important audio generation from device-local data unconstrained. Chapter 5 of this thesis presents three case studies that look at a different strategies for accomplishing this device voice generation and mapping: parametric mixing, effects-chain mapping, and historical data mapping.

Chapter 2

Motivation and Prior Art

There are many works of music and sound art inspired by or driven by the natural world. Immersive sound and musical mapping are broadly researched fields. This chapter will discuss the most relevant prior art which motivates this project, focusing on Acoustic Ecology, presence, and telepresence. Then, relevant prior works related to auditory display, low-power sensor networks, and interfaces for musical mapping will be presented.

2.1 Presence

In "Composing perceptual geographies", Maryanne Amacher references the emerging "technologies of presence" that allowed for immersive experiences that function at a very basic perceptual level. In previous decades, technologies like amplifiers, loudspeakers, spatialized audio, DSP, etc. provided the means to author immersive sonic experiences that leverage presence like Amacher's pioneering "Music For Sound Joined Rooms" [1]. "Public Supply" and "Drive in Music," two installations by another pioneering composer Max Neuhaus, augmented specific locations with sound to accentuate their perceptual characteristics, using the sound and space itself as an expanded instrument [18].

Recent developments, particularly sound spatialization techniques and the advent of ubiquitous sensing devices and cheap, low-power wireless networks provide an opportunity to go a step further:

to augment our perception through immersive sound and image in a dynamic way that responds to the environment itself. In "3-D Sound for Virtual Reality and Multimedia," Begault points out that spatialized 3D audio can contribute greatly to immersivity, and as a display of information, provide fast and fluid attention shifting in comparison to visual displays, providing "situational awareness" [2]. Lombard and Ditton define presence as "the perceptual illusion of non-mediation " [20]. With 3D audio, new axies of presence measured by sensor networks can perhaps be presented with this illusion.

2.2 Acoustic Ecology

In a discussion of music sourced from a wetland, it would be remiss to ignore Acoustic Ecology. The term comes from composer R. Murray Schafer and has a philosophical connection to the work of John Cage. It is a philosophy that "suggests that we try to hear the acoustic environment as a musical composition and further, that we own responsibility for its composition" [35]. In recent history, expanded technology and recording techniques such as bioacoustics have created new niches for musicians and artists studying the natural world and the term "Acoustic Ecology" has broadened it's meaning beyond the academic school of thought that coined it. Works like David Dunn's "The Sound of Light in Trees" and John Bullit's "Earthsound"[1] record the imperceptible and make it audible "achieving a deeper understanding of how sound and our sensory modality of hearing are unique organizing forces within human society, and our physical/ecological environment." David Dunn contrasts this new concern to the more traditional role of the composer in western art music, to "express" the self through compositional acumen [10]. In contemporary practice, both concerns are frequently present, and some have pointed out the connection between sonification and romanticism [32].

Related to Acoustic Ecology is the art of emulating natural soundscapes with synthetic and manipulated sounds. Works like Wendy Carlos' "Sonic Seasonings" and Apostolos Loufopoulos's "Bee" are inspired by and evoke phenomena of the natural world in profound ways [3, 21]. Other artists, like Jana Winderen, have made music concrete using detailed field recordings of the natural world [26].

[1]Earthsound: http://www.jtbullitt.com/earthsound/

While these works do not use real-time sensing or sonification, they set a high bar for composition that seeks to evoke and transform nature. It is hoped that this project will facilitate sensor-aware compositions that achieve the same resonance.

Where this project transcends Acoustic Ecology is its embrace of non-audible "sensory modalities" that can only be understood by the human through mediation. Previous works that use non-audible data in the context of ecological consideration are numerous. Matthew Burtner's "Iceprints" uses the sound of melting glaciers and a century of data marking the extent of Arctic ice in a composition for piano and electronics.[2] Marty Quinn's "Climate Symphony" makes prosaic use of sonifications of climate-related data on huge timescales including chemical analysis of ice cores, ice sheet movement and change, and solar intensity [28]. These works make use of static data sets rather than real-time sensing. Sturm's "Pulse Pulse" is an eight-channel composition constructed from sonifications of data from 14 buoys in the Pacific Ocean [33]. While these sensors provide real-time data that is spatial in nature, the composition itself is a fixed construction and does not consider the spatial distribution of the buoys. This thesis presents musical compositions that are driven in real-time and focus on the spatial as well as temporal variation of environmental parameters.

2.3 Telepresence

In Maryanne Amacher's first works, an installation series called "CITY-LINKS" begun in 1967, she pioneered the use of telepresence in art. In these works, sound from disparate locations were transmitted to an exhibition space in real-time over telephone lines. Distant spaces were brought together in a small space to be experienced synchronously, inviting perception beyond the walls of the exhibition, inspiring simultaneous presence in all these spaces [17].

Echoing Amacher's work, this project records the soundscape of Tidmarsh in real-time with numerous microphones. In "Cricket Radio," Himmelman suggests that we are unaware of the sounds of nature and encourages us to go out into the world, collect, and listen to night-singing insects [15]. In our contemporary world, many do not have frequent opportunity to connect to the spaces and sounds of the natural world. One can tune into the birds, frogs, and crickets at Tidmarsh from

[2]Iceprints: http://matthewburtner.com/iceprints/

anywhere on earth. These audio streams can be used and blended into a musical composition with sonification elements driven by the environmental sensors, which add extra dimensions of telepresence. These audio streams from the wetland were presented as part of a project called ListenTree which used transducers attached to the roots of trees to create an audio-haptic display that blends into the natural environment [9].

2.4 DoppelLab

DoppelLab is a 3D cross-reality representation of the MIT Media Lab populated with visual representations of sensor devices located throughout the lab and spatialized audio streams from microphones distributed throughout the lab [16]. DoppelLab is the predecessor and namesake of DoppelMarsh and builds on a variety of projects at the MIT Media Lab dealing with ubiquitous sensing and cross-reality [19]. It began the process of imagining spatialized sonification of non-audible data in a cross-reality virtual environment, but stopped short of building a platform for composers to realize music. SensorChimes is a logical next step, building a versatile system for exploring both sonification and composition with a new sensor deployment and 3D environment.

2.5 Auditory Display for Spatial Data and Sensor Networks

Sonifications are already present in our world, from the classic Geiger counter which blips for each detected ionizing particle, to most recent work like SensorTune which uses sonification to assist with setting up a wireless sensor network like the one at Tidmarsh [6]. However, auditory display for spatial data, especially data collected over a wide area, is an inherently difficult problem which has been researched much less. Nasir et. al. provide a good review of strategies for handling spatial data in sonifications focusing on the synergy between spatialization of sound and spatial data [25].

A recent project from our lab called Quantizer is a musical mapping framework for compositions based on calorimeter data from the ATLAS experiment at CERN [4]. The data associated with a collision event in the Large Hadron Collider are energy deposit distributions across the geometry

of four layers of detectors. Compositions produced with Quantizer took a several approaches to handling the spatial nature of the data. One sweeps across the geometry of the distribution over time. Another uses spatialization to "approximate the sensation of the listener's head position at the center of the detector," with synthesized sounds following approximate particle trajectories. The data collected at Tidmarsh are of a very different nature, each sensor measures only one value (not a distribution), and the real-time rate of variation of the parameters is much slower than the extremely short-duration collisions at the LHC.

Chapter 3

ChainFlow

ChainFlow is a low-level interface for working with data from a sensor network in the graphical programming language called Max/MSP, built on top of ChainAPI. The goal of the ChainFlow interface is to make it easy to route any data from ChainAPI to any point in a Max patch to allow for quick realization of mapping ideas with a minimum of work and expertise. Real-time sensor updates from specified devices, aggregate measures over multiple devices, and historical time-series data can all be pulled into Max. ChainFlow is designed to stand on its own and be a fairly general tool for integrating a sensor network into Max/MSP, agnostic to both its ultimate use and the specifics of the site. However, it incorporates a number of objects that were designed to facilitate the specific patterns of mapping that are described in later chapters. This chapter gives a brief background in Max/MSP and ChainAPI, and then discusses ChainFlow. A complete description of the objects in ChainFlow is presented in Appendix A.

3.1 Max/MSP

Max/MSP is a well-respected visual programming language for media, owned and distributed by Cycling '74, which is in common use by electronic musicians and composers [23]. It was chosen as the composition interface for this project over some alternatives like SuperCollider or Chuck because of its accessibility to artists that lack a background in software engineering or experience with text-

based programming, and because of its polished graphical interface which makes it appropriate for building listener-facing interactive interfaces [34, 5].

To understand the discussion related to Max/MSP it is important to now define a few terms. A Max/MSP graphical program or subprogram is called a "patch." An "object" is an abstraction which is defined by its attributes and methods and is represented in the patch as a rectangle with "inlets" and "outlets." For example a `[sum]` object might output the sum of a sequence of numbers from its outlet when it receives that sequence at its inlet. The patch routes information from object outlets into other object inlets in the form of "messages" which are scheduled as they happen in one thread, or in the form of "signal" buffers which are scheduled in a thread constrained to run at a chosen signal rate. These routes are represented in the patch by lines or "cables." The methods of an object define its behavior when it is created, destroyed, and when it receives a message at one of its inlets.

3.2 ChainAPI

ChainAPI is an HTTP/WebSocket API developed by Russell et. al to provide a common interface for accessing data and metadata from sensor networks in the absence of broadly adopted standards [31]. It is built on top of HAL/JSON and follows the principles of REST [13, 22]. For SensorChimes and other projects related to Tidmarsh, ChainAPI provides strong separation between the sensor network that produces data and the client applications that consume it. This allows the sensor network to be developed independent of applications that use it. Additionally, the ChainAPI abstraction layer means that applications built on top of it, like ChainFlow, can easily be reused with a different sensor deployment or system that implements ChainAPI. A client talks to a ChainAPI server instance over HTTP. When available, ChainAPI exposes websocket connections for real-time data updates.

ChainAPI is quite flexible, but for this project, it is organized around three abstractions: "site," "device," and "sensor." The "site" is the broad wrapping abstraction that carries a collection of devices and metadata—Tidmarsh is a "site." Each "device" corresponds to a physical device or node of the sensor network that has a single position in space, a unique identifier, and carries a

collection of sensors. Each "sensor" corresponds to one of the sensors physically attached to a device and carries the name of the metric that it measures plus the timestamped data that have been collected by that sensor.

3.3 Building an Interface in Max/MSP

The first version of ChainFlow was implemented entirely in Max/MSP's visual language as a series of "patches." Because Max does not have built-in objects for connecting to websockets or for making HTTP requests, an external server written in python communicated with the ChainAPI server and relayed information to Max over OSC[1]. Although functional, there were a number of downsides to this approach. First, the OSC bridge complicated the interface and subverted the goal of making a convenient and flexible interface that would be easy to reconfigure. Second, it proved very difficult to implement more complicated operations like querying, resampling historical data, caching recent values, and maintain spatial metrics on recent sensor readings; these were left to the python server which necessitated an increasingly complex OSC interface and increasingly complicated OSC message routing in Max. Third, having an external python process running made the system less portable.

The second version of ChainFlow takes advantage of the Max SDK, which provides a way for developers to integrate their own objects into Max written in C. Implementing ChainFlow as a collection of C-externals means web transactions and other complex operations are implemented in C-code (and thus not directly present in the patches), simplifying the interface.

3.4 Site Abstraction

ChainFlow is built on a master/worker pattern. The master `[chain.site]` object is associated with the ChainAPI "site" abstraction. For this thesis, the site is universally Tidmarsh, but any chain-compatible site can be used. When the `[chain.site]` object loads, it requests a summary of the site from ChainAPI, including the list of devices and sensors and the URL for the websocket

[1]Open Sound Control: http://opensoundcontrol.org

stream of real-time update events if they exist. It can then start a process to pull from a websocket stream or query historical data and broadcast sensor updates as they arrive to any attached worker objects described below. The `[chain.site]` object saves cached data and metadata in a SQLlite database in memory.[2]

Each master object is instantiated with a "name" attribute. Worker objects are also instantiated with a "name" attribute and attach to an existing `[chain.site]` of the same name. If no such `[chain.site]` exists, the worker object exists in an unattached state until it finds a `[chain.site]` to attach to. When a `[chain.site]` gets destroyed, attached worker objects detach.

Site-wide state, including websocket connections, cached data, and metadata, must be saved somewhere in memory, ideally without redundancy. The master/worker paradigm saves this state only in a single `[chain.site]` object which other objects can still access and modify. The usual Max paradigm would have that state be propagated to other objects via messages through the outlets and inlets. By maintaining a connection between master and worker objects in C-code, the amount of patching required to integrate the various objects in ChainFlow is reduced significantly.

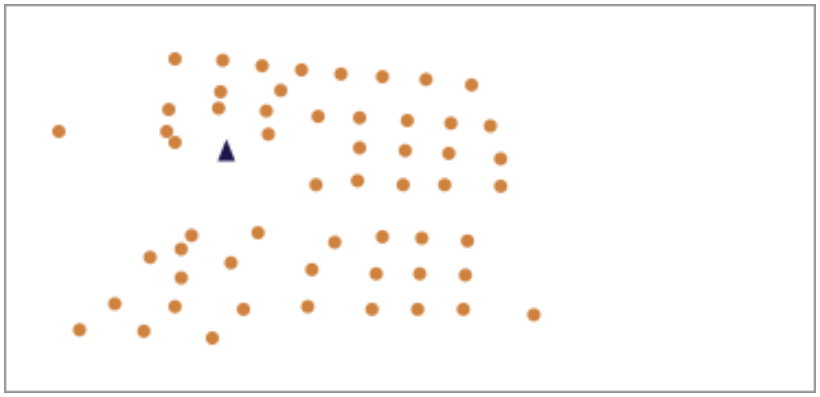

Figure 3.1: The `[chain.map]` object uses the UI portion of the Max SDK to present a map of the associated site. Each dot represents a sensor device which can be selected with a mouse click. The blue arrow is a virtual "listener" position that can be set with a message.

The most basic worker object is `[chain.info]` which provides access to the metadata stored in the associated `[chain.site]` including the list of devices and metrics. Additionally, it can return the nearest n devices from a selected point in order of distance, which can be useful for finding a device near a point of interest. Figure 3.1 shows the `[chain.map]`, a UI object that presents a map

[2]SQLite: https://www.sqlite.com

of the sensor devices at the associated site. Each device appears as an orange dot, and the position of a virtual listener represented by a blue arrow can be set.

3.5 Space Abstractions

The `[chain.device]` object provides an interface to all sensor metrics measured at a given point by a physical device. These metrics can also be normalized against the mean and standard deviation of the entire site. Each instance of `[chain.device]` carries it's own location as Latitude/Longitude coordinate pair and in Unity meters in the X-Z plane in a manner consistent with the coordinate system of DoppelMarsh. These meter coordinates are useful for determining distances and are used throughout the internals of ChainFlow.

The `[chain.zone]` object is designed to statically manage the list of devices that are within a circular "zone," specified by center point and radius, as the center point and radius vary. The "zone" is specified with an "enter" and an "exit" radius. If the zone moves such that a device that was formerly outside the "enter" radius is now within the "enter" radius, that device is added to the list and the object outputs `added [device_name]`. If the zone moves such that a device that was formerly inside the "exit" radius is no longer, the device is removed from the list and the object outputs `removed [device_name]`. Setting the "enter" radius inside the "exit" radius provides hysteresis which prevents oscillation between adding and removing a device when it is very close to the boundary. This object is used in the implementation of the immersive sonic environment.

The `[chain.metric]` object provides a more general interface for a specified metric which provides access to its value and measures on its spatial variation across all of the devices that measure it. For each `[chain.metric]`, a metric is selected and a circular zone around point p with radius r is specified. Then, the object can return a number of things:

1. The proximal or bilinear interpolation of the metric at p based on near devices
2. The mean value of the metric for the devices within the r of p
3. The standard deviation of the metric for the devices within r of p

4. The deviation of the interpolated value of the metric at p from the mean within r

5. The max value of the metric among the devices within r of p

6. The min value of the metric amoung the devices within r of p

7. The median value of the metric among the devices within r of p

Importantly, using the proximal interpolation, provides a way to retrieve the sensor reading for the selected metric that is most relevant to the chosen point in space. With this interface, even if a metric is only measured by a few devices in the site, the best guess for the metric at a any point is still returned.

3.6 Time Management

The marsh undergoes changes on many timescales at once. There are daily cycles, weather patterns changing day to day, seasonal changes month to month, and climate change from year to year. Any and all of these timescales of change could be interesting for an artist to realize in music.

3.6.1 Real-time and Pseudo-Real-time

Each `[chain.site]` instance has a clock, which can either run in "real-time" following the system clock, or in "pseudo-real-time" specified by a chosen rate in historical seconds per second and a starting point in the past. In both modes, the site object notifies attached workers when an update from a sensor (real or historical) is received. When running in "pseudo-real-time," the `[chain.site]` instance starts two threads and a priority queue. The first thread looks ahead into the future-past and makes HTTP requests to ChainAPI for events which it puts into the priority queue. The second thread consumes the priority queue as the historical clock advances past the earliest event in the queue.

3.6.2 Sequences of Historical Data

The site-wide clock is useful for mapping patterns that use the instantaneous state of the environmental conditions (now or in the past) to determine musical characteristics. However, mapping patterns that consider the trajectory of an environmental parameter over the course of a period of time for determining musical characteristics on a timescale require access to series of historical data. To facilitate this kind of pattern, the `[chain.device]` object implements a `data` method, which fetches a sequence of measurements for a chosen metric from a chosen start time to a chosen end time. For large time spans, this can be an enormous number of data points, so the behavior can be modified to return only one point per chosen interval. The data point in each interval can be determined by linear interpolation around the interval boundaries or by averaging the points in each interval.

Requests to ChainAPI for large data sets are not currently well optimized and there is no caching system in place at the ChainAPI level, so ChainFlow includes its own object `[chain.data]` for saving a requested data sequence to disk so that it need not be re-requested. A convenient abstraction, `[chain.cache]`, will make a specified data request only if a file containing that data is not available that it will then write.

Data in ChainAPI are not necessarily sampled periodically, and each have a potentially arbitrary timestamp. The `[chain.data]` abstraction also provides an interface for resampling a data sequence on a specified fixed interval via linear interpolation.

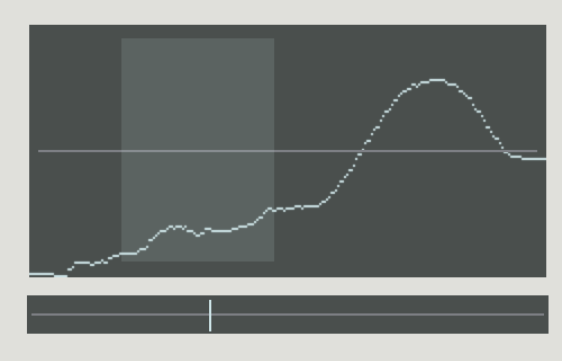

Figure 3.2: The `[chain.itertable]` abstraction combines an instance of `[chain.data]` with an graphical interface for iterating through stored data.

The `[chain.itertable]` pictured in Figure 3.2 provides a means of visualizing and iterating through a sequence of historical data.

3.6.3 Time Formatting

When handling time-series data, it is important to establish a consistent representation of time. All time-series data in ChainAPI are stored in ISO formatted strings with the UTC timezone. This format is usefully human-readable; however, it does not easily afford mathematical operations. In ChainFlow, time is consistently represented in seconds since the UNIX epoch (UTC) with possibly a floating point partial second. With the exception of the `[chain.time]` object, objects in ChainFlow use the seconds since UNIX epoch format in inputs and outputs to make it easy to perform mathematical operations in seconds on the message within Max/MSP. The `[chain.time]` object provides a `parse` method and a `format` method to go back and forth between the two representations.

To make it more convenient to specify a time range for a historical series request, two additional abstractions were created. The `[chain.tdparse]` abstraction converts human-readable time deltas into seconds. For example, "1day 1hour 1min 4sec" would be converted to 90,064. The `[chain.timerange]` abstraction converts a human-readable time range specification relative to "now" or "mdnt" (the most recent midnight) to a start-end pair of seconds since the UNIX epoch. These specifications can come in three forms: "last [timedelta]", "next [timedelta]", or "from [time] to [time]." A few examples are presented:

- `last 4hour` - range spanning the previous four hours
- `next 3hour 2sec` - range spanning from now until three hours two seconds in the futre
- `from now -1day to now` - range spanning from yesterday at this time to now
- `from mdnt -1day to mdnt` - range spanning from the midnight that began yesterday to the midnight that began today

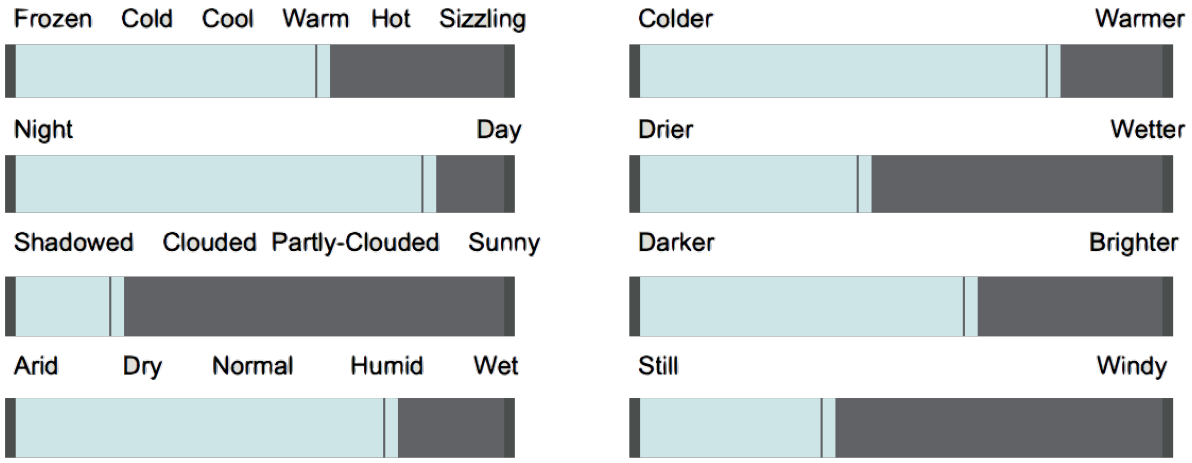

Figure 3.3: Semantic Parameters Interface. Each of these sliders corresponds shows the local conditions at a selected device projected onto a "semantic" axis. These sliders are moved to follow real conditions or can be moved by the composer to simulate conditions in the process of composition.

3.6.4 Semantic Parameter Mapping

An experimental composition interface, shown in Figure 3.3, was created by adding a "semantic" mapping layer between the sensors and the instrument. This is a site-specific abstraction implemented with ChainFlow. Rather than presenting the composer with the raw or analytic environmental parameters (temperature in degrees, illuminance in lux, temperature deviation, etc.), the composer is presented with "semantic" parameters, each on a 1 to 128 scale.

For a metric like temperature, most people have an intuitive sense of what the raw data means. However, for measurements like illuminance and humidity, these semantic parameters can be reasoned about more intuitively than the raw data. For this experiment, the semantic parameters and the equations of raw parameters that define them were authored. Future work could experiment with generating this first layer of mapping to semantic parameters through principle component analysis or dimensionality reduction from user input.

Chapter 4

Virtual Sonic Environment

At the marsh, one sensor will be less illuminated because it is in the shadow of a tree. Another sensor will read more humid because it is shrouded in mist from the nearby pond. Much of the interest in collecting data from many sensors scattered across an environment is capturing the spatial variation across the environment. The sense of being present in one part of the marsh vs. another is reflected in these variations, and a good musical mapping should manifest these variations. Interactivity can make a musical mapping or sonification much more convincing [14]. Exploration of space is an obvious candidate for interaction with a sensor-augmented space. This chapter will describe the high level picture of how interfaces and systems are integrated to render a sensor-driven immersive virtual environment that is both graphically and musically active.

At a high level, SensorChimes follows a client-server architecture. Figure 4.1 gives an overview of the entire system. The "server" is Tidmarsh, the sensors deployed there, and the ChainAPI server that provides access to sensor measurements. The "clients" are DoppelMarsh and Max/MSP with ChainFlow.

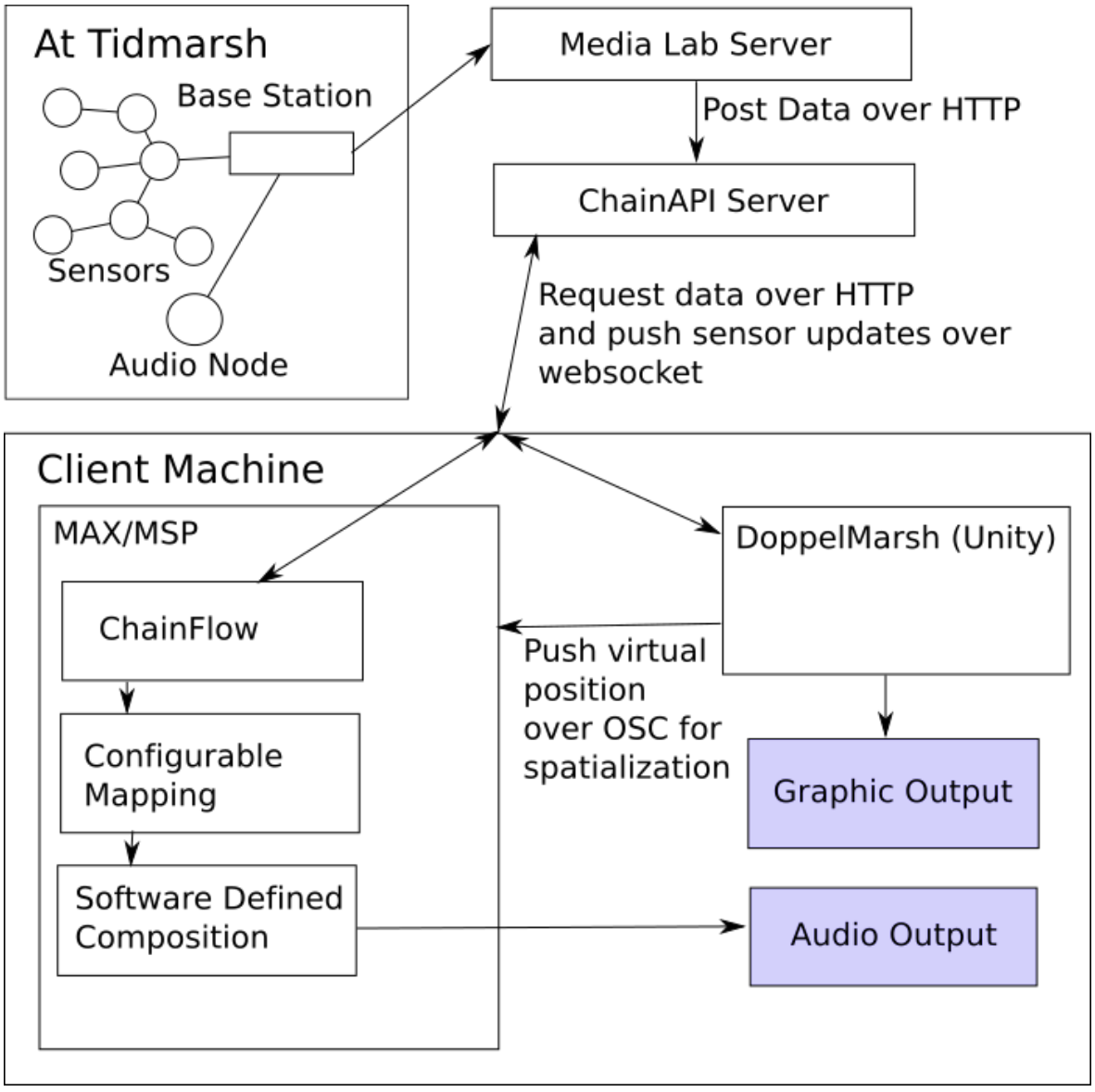

Figure 4.1: Within Tidmarsh, sensor nodes communicate low-level messages to a base station connected to the internet which relays to a server at the lab. These messages are parsed and posted to the ChainAPI server over HTTP. The DoppelMarsh and Max/MSP clients interface with ChainAPI and communicate with each other over OSC.

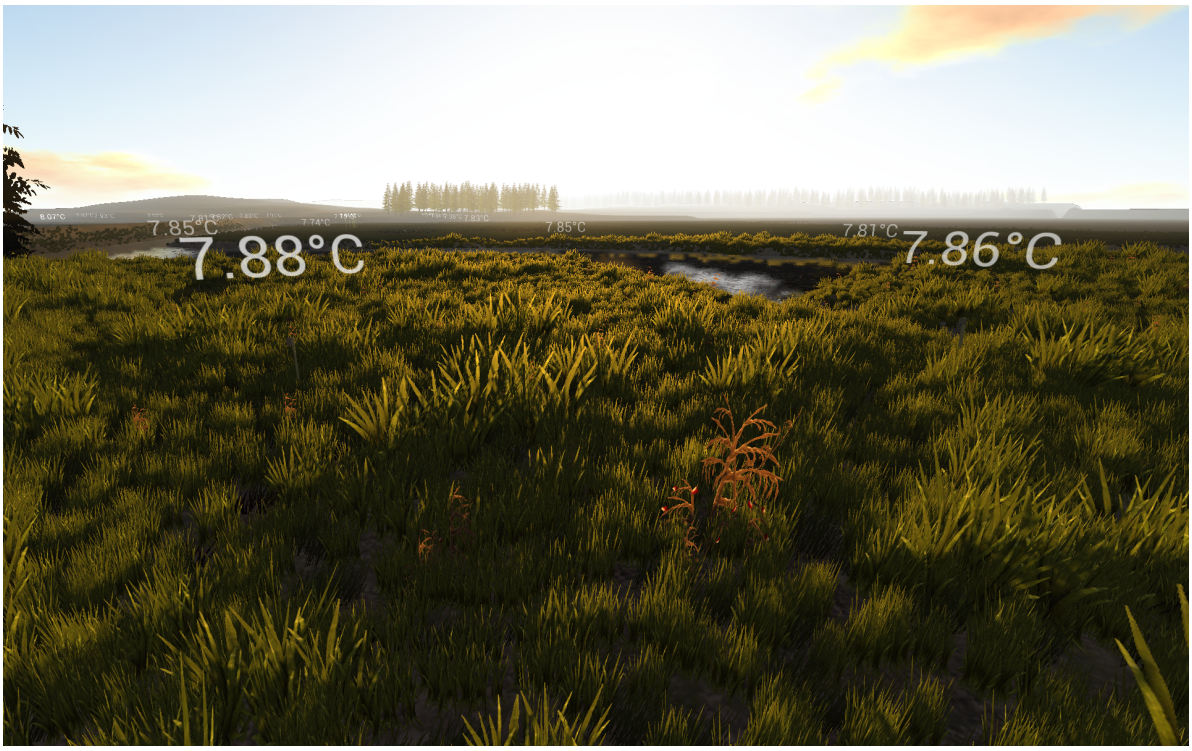

Figure 4.2: Virtual Tidmarsh

4.1 DoppelMarsh

DoppelMarsh, an experiment in resynthesized-reality, is a "cross-reality sensor data browser constructed using the Unity[1] game engine to experiment with presence and multimodal sensory experiences" created by Gershon Dublon and Brian Mayton [24]. It presents a virtual rendering of the marsh with real-time sensor readings visualized as floating numbers above the sensor nodes pictured in Figure 4.2. In the future, graphical elements that more subtly blend into the scene will visualize sensor readings. The user controls the position of the camera using controls common to first-person video games. This virtual position and direction are sent to Max/MSP over OSC[2] so that audio generation that is contingent on the player position is synchronized with the graphical display. Previous versions of DoppelMarsh have integrated both the live audio streams from Tidmarsh and a hard-coded sonification of the temperature and humidity data composed by Responsive Environments RAs Spencer Russell and Gershon Dublon.

[1]Unity: https://unity3d.com/5
[2]Open Sound Control: http://opensoundcontrol.org/

4.2 Immersive Sound in Max/MSP

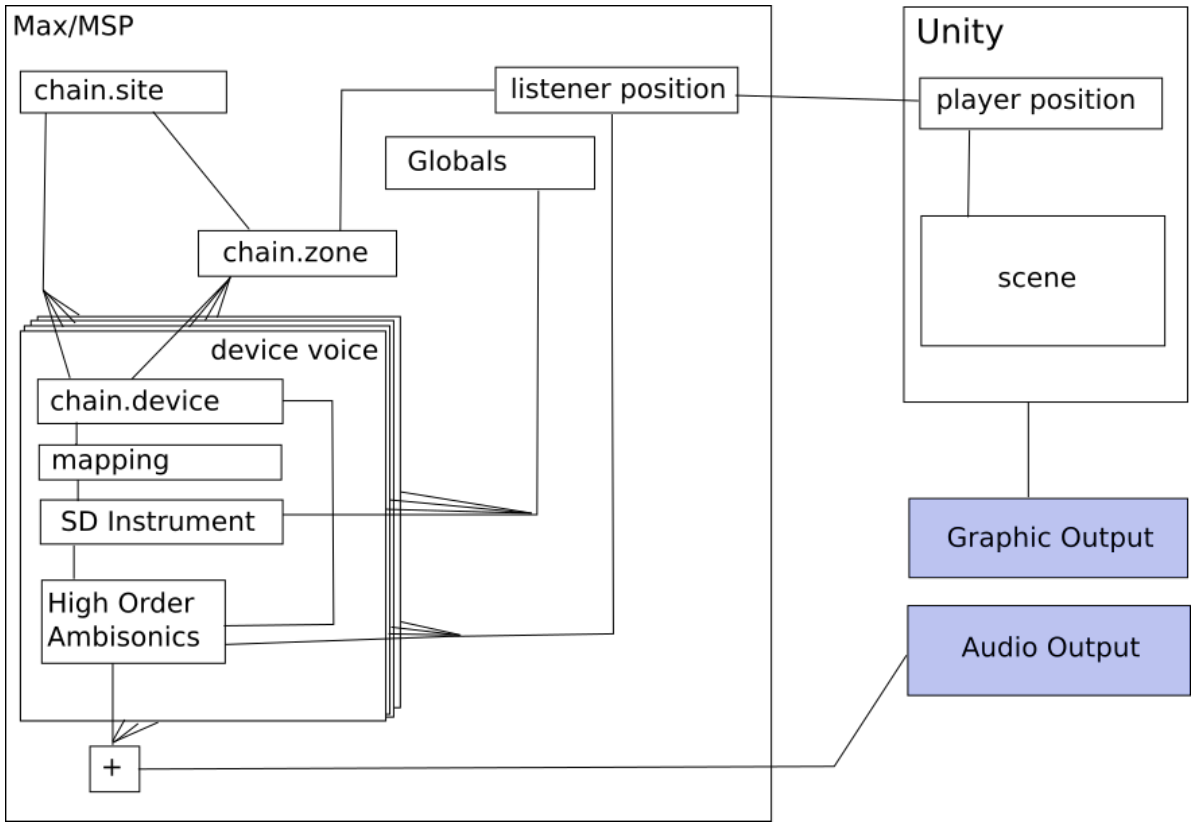

Figure 4.3: Virtual Sonic Environment Overview. Many instances of the device voice patch are managed by `[chain.zone]` which follows the virtual player position in Unity. Each device voice generates a signal based on a mapping from device-specific data, global data, and a parametric software defined instrument. This signal is spatialized based on the player's virtual position.

A system was created to render an immersive sonic environment with each sensor device at Tidmarsh generating an audio voice. A different musical audio signal is generated for each sensor device based on the environmental conditions local to that device. These signals are delivered to the listener simultaneously, each spatialized to sound as if emanating from the device. The listener moves around the virtual space and explores the different signals and the underlying environment.

In Max, the environment patch is broken down into three parts: device voice, spatializer, and polyphony handler. Each component is described below.

4.2.1 Device Voice

The device voice *is* the composition in some sense. Each device will contribute a voice to a chorus of nearby devices, the device voice patch determines this voice. A `[chain.device]` object is used to access real-time data for the specified device. The device voice patch is responsible for using that input along with aggregate features to produce an audio signal output. A few strategies for the musical mapping that makes these devices sound are explored in the case studies that follow.

4.2.2 Spatializer

Audio spatialization refers to the practice of processing an audio signal to give the impression that it is incident on the listener from a particular direction at a particular distance. The spatializer processes the device voice for spatialization using the High Order Ambisonics (HOA) library for Max/MSP from CICM to encode the signal of a device voice patch on the spherical harmonics based on the displacement of the device from the listener.[3] Spatialization itself is not the focus of this thesis, and we currently use a very simple spatialization model that does little to characterize the space with reverb and distant-dependent filtering. A more complicated model could give a stronger sense of immersive presence in the space, and improvements along these lines will be made in the future.

4.2.3 Polyphony Handler

The polyphony handler maintains a set of spatializer objects to which it assigns devices that are relatively near the listener. It uses a combination of the `[chain.zone]` object and Max/MSP's built in `[poly~]` object to instantiate a number of unassigned spatializers and assign and reassign them to nearby devices as the listener's position moves around.

[3]HOA: http://www.mshparisnord.fr/hoalibrary/en/

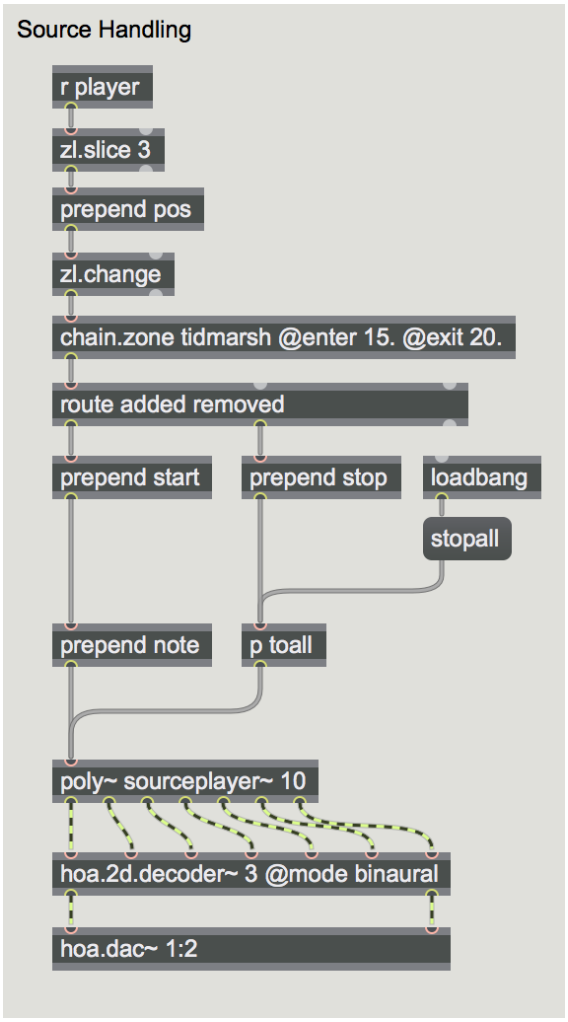

Figure 4.4: Polyphony Handler

4.3 Client Bundling and Deployment

Initial demonstrations were deployed as an installation with a dedicated computer, display, and sound system that remote visitors were invited to use. With a keystroke, the user could switch between different real-time musical conceptions of the marsh. However, to reach a broad audience, a web deployment was devised. In the future it may be possible to attempt an onsite deployment for visitors of Tidmarsh to enjoy.

4.3.1 Installation Deployment

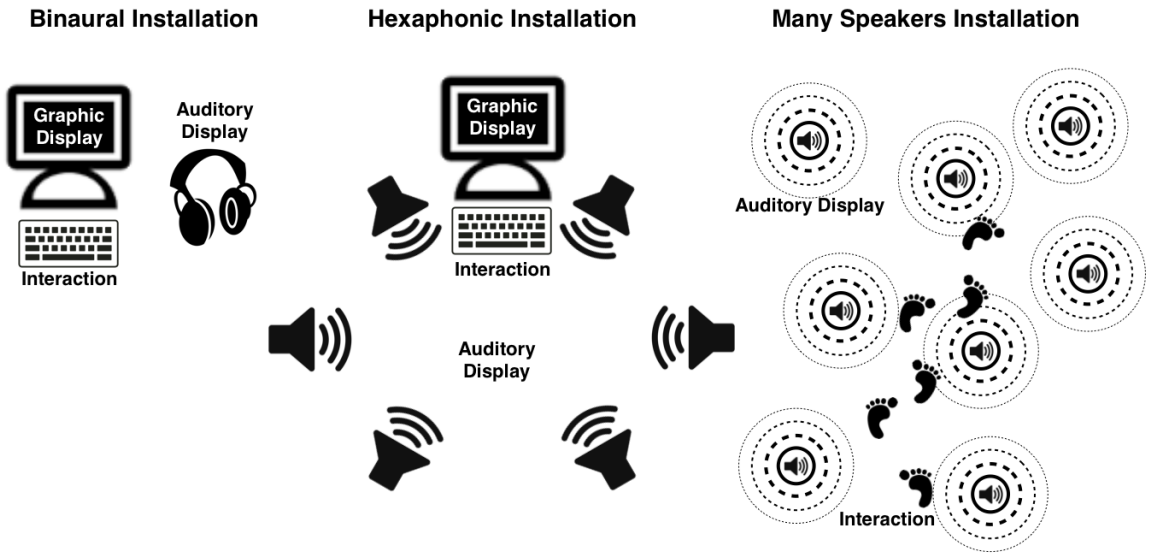

Figure 4.5: Three different physical deployment scenarios.

Several installation types have been imagined, diagrammed in Figure 4.5. For the first installation, the computer ran both the Max/MSP and DoppelMarsh clients simultaneously. The user sat in front of the display wearing headphones and used the keyboard to navigate through the virtual world. As the user moved around the virtual space, the Max/MSP client followed the virtual position via OSC and used binaural spatialization to make a convincing immersive sonic environment consistent with the graphical experience.

The first demonstration was limited to a single user. To make a larger installation that could be viewed by multiple people at once, six speakers were set up in a hexagon. The display remained

the same, with just a single user controlling the virtual location of the camera. However, sound was spatialized using high-order ambisonics using the six speaker channels so that anyone in the ring of speakers could listen.

A proposed installation that has not yet been put into to practice would make use of a fairly large space and a large number of omnidirectional speakers. These speakers would be arranged in the space following the arrangement of sensor devices at Tidmarsh. The DoppelMarsh client would not be used, and spatial exploration would instead occur in the very real space filled with speakers. A computer running the Max/MSP client would route a signal to each speaker for each mapped node at Tidmarsh. Each of these installation deployments is diagrammed in Figure 4.5.

4.3.2 Web Deployment

Deploying this project to the web was a necessary step to reach a large audience without having a dedicated space to deploy an installation. Embedding Unity in the browser is difficult and not currently supported by all major browsers and embedding Max/MSP in the browser is currently impossible, so the web deployment was achieved by publishing a downloadable application. The two clients (DoppelMarsh and Max/MSP) were bundled together as a client application with a wrapping application written in Objective-C for Mac OS X. The user can switch between compositions with a keystroke.[4]

4.3.3 Onsite Deployment

In the future it may be possible to deploy SensorChimes onsite at Tidmarsh using a mobile version of the auditory display combined with a head tracking system. An onsite deployment would build on "HearThere," a project by Russell et. al. presented at the 7th Augmented Human International Conference [30].

[4] http://resenv.media.mit.edu/sensorchimes/

Chapter 5

Case Studies

5.1 Case Study 1: Parametric Mixing

This study is a piece written by the author entitled *The Bog Blues*. We call the pattern of mapping at it's core "parametric mixing." In the overall picture, parametric mixing functions as a process for generating audio from device-local conditions to integrate into the Virtual Sonic Environment as described in Chapter 4.

This piece is constructed from looping improvised acoustic passages performed on cello, guitar, bass, and drum. The layers of sound emulate the many layers of narrative that unfold in a complex ecosystem like Tidmarsh. Many distinct tracks for each instrument were recorded with consistent meter and harmonic rhythm. By design, these tracks were composed to produce a range of moods and perceived energy levels when layered on top of each other in different mixes while maintaining aesthetic consistency.

Each device in the virtual environment adds a voice which is a mix of these tracks. For each device, the weights which determine the mix are mapped from the sensors readings of that device.

5.1.1 Parametric Mixing

"Mixing" is simply the process by which audio signals are linearly combined. Mixing occurs in performance, recording, and production settings where different voices, parts, instruments, etc. are layered, each gained appropriately to achieve a desired combined signal. A large amount of variation in the musical character of a recording can be achieved by varying the mix of voices. The idea behind parametric mixing is to create a composition which is designed to take that variation to the limit by supplying many different voices that are wildly different in character but aesthetically cohesive, and then to mix these voices together in myriad ways parameterized by other input. In this case reacting to sensor measurements at Tidmarsh. Composition of a parametric mixing piece consists of two tasks: creation of the many tracks to be mixed, and specification of the parametric mapping which determines the mix. Two strategies for the second task were attempted with *The Bog Blues* presented below.

Mixing Specification Version 1: Stochastic Mixing

The recorded tracks were split into groups: all *arco* cello tracks, all drum tracks, etc.. The final mix would be formed from one track from each group. Periodically a stochastic process would select one track from each group, with the option to select no track for some groups, based on a predetermined mapping and the current environmental conditions. Starting from the next looping point, those tracks would be mixed together at equal gain. The predetermined mapping worked as follows. Each track was associated with a vector of sensor values. From each group, the selected track was randomly chosen with a distribution weighted by the Cartesian distance between the real conditions as measured by the sensors and the vector associated with each track.

Stochastic mixing by this strategy exhibited a number of undesirable characteristics. First, assigning a vector to a track to make it more likely to be in a particular region of the parameter space frequently produced unexpected results and proved to be a non-intuitive tuning problem. Second, because the tracks were selected in a binary way, there was no way to smoothly transition from one mix to another. Since the mix would only update at looping points, there was a notable lag between changing conditions and changing mix.

Mixing Specification Version 2: Distribution Mixing

The second version solved the short-comings of the first by using smoothly varying gains on each track rather than binary weights.

The mapping is specified by a max-one axis-aligned multivariate Gaussian distribution in the abstract parameter space defined by the sensors of a single device. This parameter space should not be confused with the physical space in which the many devices are distributed that is also at play in this composition. For example, for "The Bog Blues," a five-dimensional parameter space consisting of temperature, illuminance, deviation temperature, deviation illuminance, and deviation humidity is used, so each track is associated with a five-dimensional Gaussian specified by it's center and covariance matrix in this five-dimensional space. The Gaussians were restricted to axis-aligned (i.e. diagonal) covariance matrices for ease of implementation.

The mix of tracks for a given point in this five-dimensional space is the linear combination of the tracks, each weighted by the evaluation of its respective Gaussian distribution at the relevant point. This can be thought of as a zeroth-order spatialization of the tracks in the parameter space heard from the perspective of a listener in the space.

For example, if the environmental conditions are exactly the mean of the Gaussian associated with a particular track, that track will be included in the mix with weight 1.0. If the conditions are exactly the mean of the Gaussian except that the temperature is off from the mean by a standard of deviation, the track will be included in the mix with weight $1/e$.

This mapping strategy means that each track is confined to a specified region of the parameter space. Two tracks that clash aesthetically can be confined to disjoint regions, and the Gaussian roll-off means that smooth transitions occur as long as the conditions change continuously.

5.1.2 *Dynamix* Interface

Dynamix is a graphical interface created to illustrate and experiment with the distribution-based parametric mixing pattern described in Section 5.1.1. The interface, implemented as a web application with ReactJS and as a native application with Electron, is presented in Figure 5.1 [29,

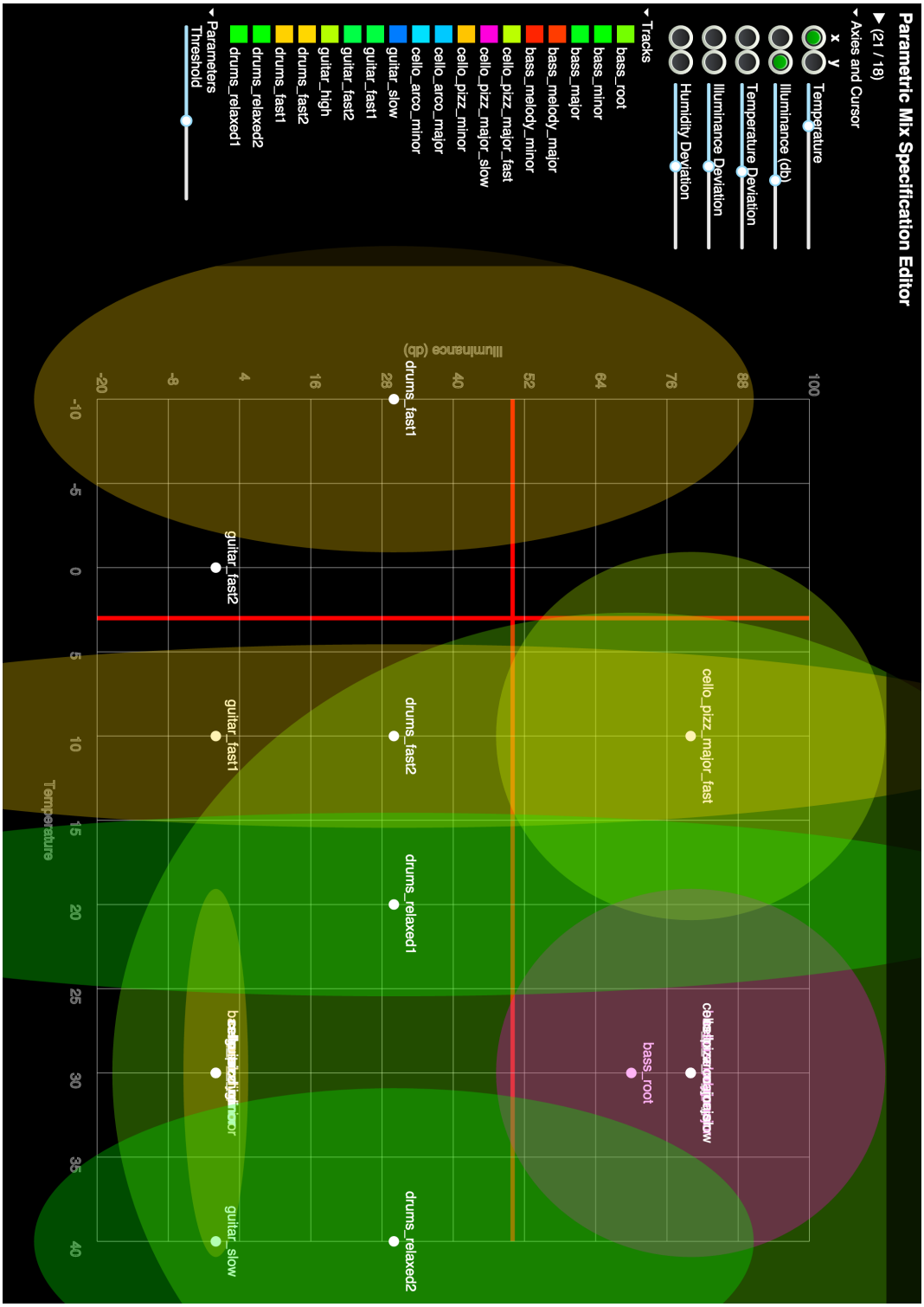

Figure 5.1: The *Dynamix* interface consists of a graph view, axies list, track list, and parameters list. A cursor position in the parameter space is set by the five sliders in the axies list. The graph visualizes each Gaussian distribution as an ellipse in two selected axies. The boundary of the ellipse is an isocontour of the distribution projected onto the shown axies at the hyperplane specified by the cursor position of the non-shown axies. The value of the isocontour is set by the "Threshold" parameter. The position of a selected track in the parameter space may be adjusted graphically or with the parameters pane.

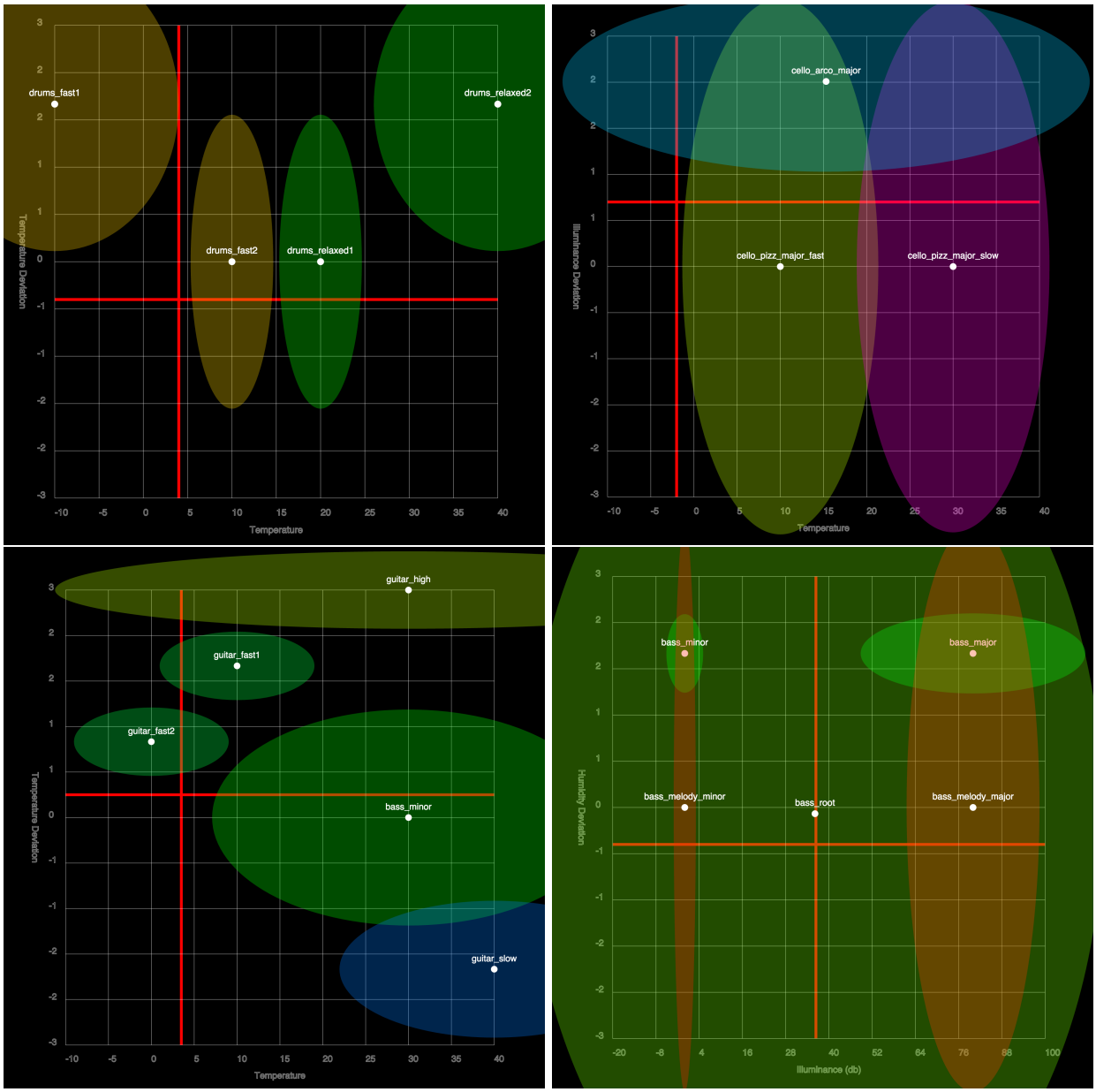

Figure 5.2: These four graphs highlight a few parts of the mapping that makes *The Bog Blues*. The top two graphs show that the rhythmic feel of the piece varies from a fast and anxious on the cold end of the temperature spectrum to slow and relaxed on the warm end. Major tracks are kept on the bright side of the illuminance spectrum and minor tracks on the dark. The lower two graphs show that the many melodic voices each take a small region of the deviation space. For example, if a device is exceptionally warm (two to three standards of deviation above the mean for the site), the "guitar_high" melodic line can be heard. If it is exceptionally cold, the "guitar_slow" melodic line is heard.

11]. A list of tracks appears on the left side, and each associated Gaussian is shown on a graph on the right. The graph is the full n-dimensional parameter space projected onto two selected axies. Above the track list is a list of axies where the shown x and y axies can be selected. A value for each remaining "hidden" axis can be specified with a slider. Each Gaussian is presented as an ellipse whose boundary is the isosurface of the Gaussian at a specified threshold projected onto the hyperplane specified by the values of the hidden axies. This interface allows visualization of the mix specification despite the potentially high dimensionality of the parameter space.

5.1.3 Discussion and Future Work

The Bog Blues is an experiment in making parametric music with instrumental samples rather than more traditional approaches, which might drive harmony-aware sound synthesizers to produce stochastic melodies based on parameters. Parts of the mapping for the final version are presented in Figure 5.2. A few criteria were determined for evaluating the success of this experiment. A good mapping should exhibit:

1. Interest Correspondence - the interesting features of a data set (things that ground its semantic interpretation) should manifest in the salient features of the music it generates

2. Consistency - data sets with related semantic interpretations should generate related music

3. Well Ranged - the scale of variation of a data set should map closely to the scale of musical variation

4. Responsiveness - changes in data should be reflected in music quickly; too much latency makes intuition much harder to develop

As parametric music, *The Bog Blues* excels at all of these criteria. By design, exceptional environmental conditions are interpreted as more exceptional version of the piece. The mapping exhibits smooth transitions between regions of the parameter space so similar conditions always produce similar music. The *Dynamix* interface makes it easy to see if the specified mapping will be active and interesting across the regime of the parameter space which the environment is likely to fall in. And, parametric mixing is essentially an instantaneous reflection of the parameters. Does *The*

Bog Blues give an intuitive sense of the environmental conditions at Tidmarsh; does it achieve augmented presence? The language of music is by no means universal, and every individual's experience with the piece will be different. However, in demonstrating the piece in the 3D virtual environment, after a few minutes, I found myself navigating through the world looking for a particularly humid region by ear despite the humid data being presented as floating text above each sensor. This demonstrates that it achieves this goal at least to some degree and the parametric mixing strategy has potential.

5.2 Case Study 2: Effects-Chain Mapping

This study is a collaboration with Evan Ziporyn, an accomplished composer and clarinetist. Following a tradition dating at least to Camille Saint-Saëns and Sergei Prokofiev, he generated looping tracks of clarinet textures inspired by the birds and frogs of the marsh. The idea behind the piece is to layer and process these textures through various effects to create a large parameterized timbral space. An "instrument" abstraction that produces this texture exposes a set of adjustable parameters that are mapped to linear combinations of semantic parameters as described in Section 3.6.4. The piece is integrated into the virtual environment by generating a chorus of these "instruments," each spatialized to emanate from sensor device.

5.2.1 Texture/Timbre Space

The sound space is parameterized by the following effects:

- Reverb
- Overdrive Distortion
- Layers – Determines how many of the clarinet layers are active.
- Pitch-locked playback speed – Determines how fast the layers playback. The timeshift is offset by a pitchshift to keep the pitch locked to selected ratios of the original. When the timeshift is great enough, the pitchshift will jump to the next approved ratio somewhat like the gears in a car.

- Whistle Interference – This effect introduces highly peaked noise around the most present frequency in the underlying tracks.

5.2.2 Mapping From Correspondences

With this piece, we devised a procedure for mapping the semantic environmental parameters to the instrument parameters of this parametric music based on training examples. First, a set of semantic parameters are randomly selected. Then, the composer or mapper is given the opportunity to adjust the instrument parameters until the sound is most reminiscent of the conditions represented by the random semantic choice. The semantic parameters and instrument parameters are recorded as a "correspondence pair," and the process is repeated until enough "correspondence pairs" have been recorded to specify a full-rank linear transformation from the semantic parameters to the instrument parameters. Future work could experiment with a machine learning approach to this process which might be more flexible or more robust to noise.

5.2.3 Discussion and Future Work

This composition, another experiment in parametric music, is still a work in progress. The timbre space Ziporyn and I created is still in its first incarnation, and in the future, we hope to experiment with a broader range of DSP effects as well as more source material. However, in its current state, it is still dynamic enough that a mapping from environmental parameters feels responsive and is interesting to listen to.

In the future, we intend to add an interface along the lines of Harmonix's "The Axe," which allows the user/listener to improvise an arpeggiated melody on top of the texture in a harmonic and timbre space adjusted by real-time sensing.[1]

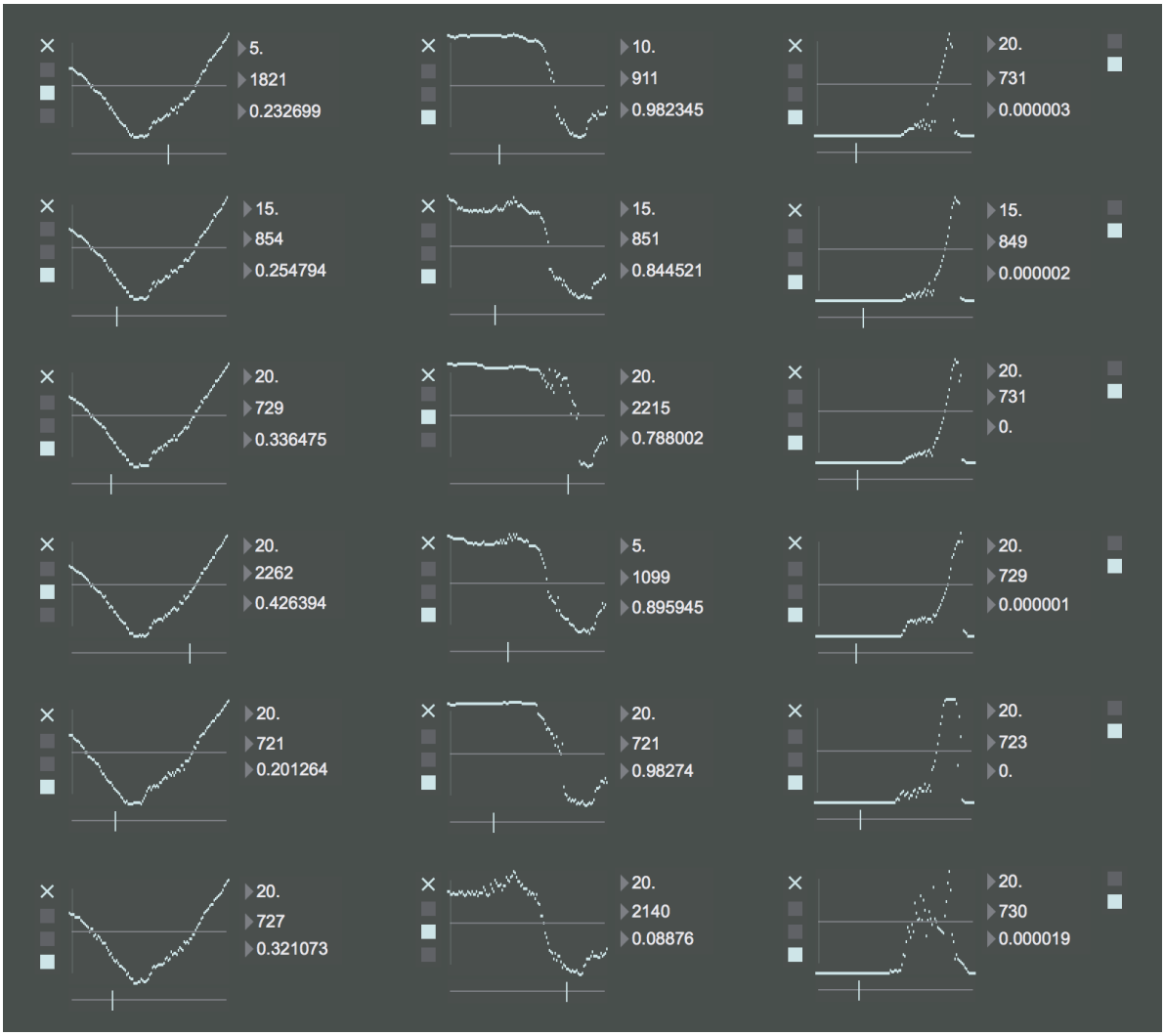

Figure 5.3: Ricky Graham's Interface. Each row corresponds to a sensor device at Tidmarsh. The first column is temperature, the second is pressure, and the third is illuminance. The three tables of each row drives a granular synth patch.

5.3 Case Study 3: Mapping Historical Data

Ricky Graham, an accomplished guitarist and computer musician, was interested in working with data from Tidmarsh's history and developed an interface that allows a listener/improviser to select data ranges to iterate through driving his own granular synthesis patch. This interface, presented in Figure 5.3 encourages exploration of how data sets can drive timbral and temporal changes in electronic music. Using this interface, a piece was created based on the contour which barometric pressure, humidity, and illuminance take over the course of a day on the full-moon. This piece was presented at the Fall 2015 Media Lab Member's Event and will appear as a research presentation at SEAMUS 2016 [12].

5.3.1 Discussion and Future Work

A new version of this piece is in progress which will be presented within the virtual environment. For various points across the wetland, real-time as well as historical data on multiple time scales will be used simultaneously to drive Graham's synthesis patches. Data from the last minute, hour, and day can all be used to drive different layers of the composition which are then spatialized to emanate from the sensor devices and specific regions of the wetland.

[1]The AXE: http://www.harmonixmusic.com/past-games/

Chapter 6

Conclusion and Future Work

In the contemporary world, the rate at which we collect and generate data and information vastly outpaces our ability to process and interpret it. Spaces are increasingly instrumented with sensors, but the insights that we might gain from these data remain mostly untapped. The technologies that mediate our access to digital information give us only small glimpses into this world of dense data. We need powerful new paradigms to render these data interpretable. On the path to these new paradigms, these data present themselves as a canvas for the contemporary artist. We cannot yet walk through a marsh and internalize everything we can measure with sensors about the world around us as easily as we can feel the warm rays of sun on our skin and the moist soil beneath our feet. However, music that speaks to these experiences, which reacts to these data, is a step toward a world where data is interpreted incidentally. In the near future, this project could break out of the confines of the virtual world and make a showing onsite at Tidmarsh, a real auditory augmentation using the head tracking presented by Russell and Dublon [30].

The potential for music creation along the lines of the pieces presented in this thesis is large and mostly unexplored. The primary goal of this project was to create a framework with which to create musical compositions. The process of using this framework in novel ways to create music is only beginning. The case studies presented here represent only the specific inspirations of a few individuals. It is hoped that ChainFlow will be adopted by or inspire more composers interested in the vision of this project who will take it in their own unique directions.

ChainFlow is a good first draft of a versatile interface for sensor networks in Max/MSP. Within minutes of thinking of a musical mapping idea, a prototype can be put together in Max which implements it with real data streams. The device and metric abstractions provide a convenient interface for real-time data, historical data, and aggregate features. However, there are many next steps. The next iteration should consider a the drawbacks of ChainFlow. First, ChainFlow is (obviously) only useful within Max/MSP which makes web deployment (almost a necessity in our time) very difficult. A similar tool could be built for Pure Data which would open up a new realm of portability because of the open-source nature of Pure Data and the libpd C library, which makes it possible to build Pure Data patches into other projects and products.[1] Second, the limits of ChainFlow have not been tested and significant optimization may be required to handle substantially larger deployments.

Each metric measured at Tidmarsh can be thought of as a scalar or vector field which has been sampled by each device. ChainFlow attempts to reconstruct this field by interpolating between the samples with the `[chain.metric]` object. However, this object only returns the interpolated value at a a single point and there is no interface to acquire a representation of the entire field at once. Future work could investigate the musical mapping potential of this kind of field representation.

[1] Pure Data: https://puredata.info

Bibliography

[1] Maryanne Amacher. *Composing Perceptual Geographies*. Course/Workshop Introduction/Description. 2006.

[2] Durand R Begault et al. *3-D sound for virtual reality and multimedia*. Vol. 955. Citeseer, 1994.

[3] Wendy Carlos. *Sonic Seasonings (Libreto del CD)*. 1998.

[4] Juliana Cherston. *Quantizer*. http://resenv.media.mit.edu/quantizer. 2016.

[5] *ChucK*. https://chuck.cs.princeton.edu. accessed January 2016.

[6] Enrico Costanza et al. "SensorTune: a mobile auditory interface for DIY wireless sensor networks". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2010, pp. 2317–2326.

[7] Paul Doornbusch. "Composers' views on mapping in algorithmic composition." In: *Organised Sound* 7 (2002), pp 145–156.

[8] Gershon Dublon and Joseph A Paradiso. "Extra Sensory Perception". In: *Scientific American* 311.1 (2014), pp. 36–41.

[9] Gershon Dublon and Edwina Portocarrerro. "ListenTree: Audio-Haptic Display in the Natural Environment". In: (2014).

[10] David Dunn. "Acoustic ecology and the experimental music tradition". In: *American Music Center, New Music Box* (2008).

[11] *Electron*. http://electron.atom.io/. accessed January 2016.

[12] Ricky Graham. "Sonifying Tidmarsh Living Observatory". Society of Electro-Acoustic Music in the United States. 2016.

[13] *HAL: Hypertext Application Language*. http://stateless.co/hal_specification.html. accessed January 2016.

[14] Thomas Hermann and Andy Hunt. "The importance of interaction in sonification". In: (2004).

[15] John Himmelman. *Cricket Radio*. The Belknap Press of Harvard University Press, 2011.

[16] Nicholas Joliat, Brian Mayton, and Joseph A Paradiso. "Spatialized anonymous audio for browsing sensor networks via virtual worlds". In: (2013).

[17] Paul Kaiser. "The Encircling Self: In Memory of Maryanne Amacher". In: *PAJ: A Journal of Performance and Art* 36.1 (2014), pp. 10–34.

[18] Brandon LaBelle. "Background Noise: Perspectives on Sound art". In: Second Edition. Bloomsbury Publishing USA, 2015. Chap. Ch. 10.

[19] Joshua Lifton et al. "Metaphor and manifestation—Cross-reality with ubiquitous sensor/actuator networks". In: *IEEE Pervasive Computing* 3 (2009), pp. 24–33.

[20] Matthew Lombard and Theresa Ditton. "At the Heart of It All: The Concept of Presence". In: *Journal of Computer-Mediated Communication* 3.2 (1997), pp. 0–0. ISSN: 1083-6101. DOI: 10.1111/j.1083-6101.1997.tb00072.x. URL: http://dx.doi.org/10.1111/j.1083-6101.1997.tb00072.x.

[21] Apostolos Loufopoulos. *Bee*. 2010.

[22] Mark Masse. *REST API design rulebook*. " O'Reilly Media, Inc.", 2011.

[23] *Max/MSP*. https://cycling74.com/products/max/. accessed January 2016.

[24] Brian Mayton et al. *Tidmarsh Living Observatory*. http://tidmarsh.media.mit.edu/.

[25] Tooba Nasir and Jonathan C Roberts. "Sonification of spatial data". In: (2007).

[26] Alison Pezanoski-Browne. "The Tragic Art of Eco-Sound". In: *Leonardo Music Journal* 25 (2015), pp. 9–13.

[27] Larry Polansky. *Manifestation and Sonification. The Science and Art of Sonification, Tufte's Visualization, and the 'slippery slope'to Algorithmic Composition. An Informal Response to Ed Childs' Short Paper on Tufte and Sonification; with additional commentary by Childs, 2002*. 2002.

[28] Marty Quinn. "Research set to music: The climate symphony and other sonifications of ice core, radar, DNA, seismic and solar wind data". In: (2001).

[29] *React*. https://facebook.github.io/react/. accessed January 2016.

[30] Spencer Russell, Gershon Dublon, and Joseph A Paradiso. "HearThere: Networked Sensory Prosthetics Through Auditory Augmented Reality". In: *Proceedings of the 7th Augmented Human International Conference*. ACM. 2016.

[31] Spencer Russell and Joseph A Paradiso. "Hypermedia APIs for sensor data: a pragmatic approach to the web of things". In: *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 2014, pp. 30–39.

[32] Volker Straebel. "The sonification metaphor in instrumental music and sonification's romantic implications". In: *Proceedings of the 16th international conference on auditory display, Washington*. 2010.

[33] Bob L Sturm. "Pulse of an ocean: sonification of ocean buoy data". In: *Leonardo* 38.2 (2005), pp. 143–149.

[34] *SuperCollider*. https://supercollider.github.io. accessed January 2016.

[35] Kendall Wrightson. "An introduction to acoustic ecology". In: *Soundscape: The journal of acoustic ecology* 1.1 (2000), pp. 10–13.

Appendix A

ChainFlow Objects

The following is a summary of each object and patch that makes up ChainFlow.

A.1 Externals

A.1.1 [chain.site]

This object corresponds to the ChainAPI "site" abstraction. In addition to maintaining the connection to the ChainAPI "site" and caching data and metadata, this object also maintains a clock which can be locked to realtime or advance at a chosen speed from a point in history.

Attributes:	
`name` (string)	name of site
`url` (string)	url to ChainAPI site to connect to
`include_nonactive` (bool)	whether to include devices that do not have recent data
Methods:	
`load`	request site summary from ChainAPI URL.
`start`	connect to websocket and begin updating sensor readings
`start [start] [scale]`	begin updating workers with historical data from time `start` at `scale` seconds per second.

A.1.2 [chain.info]

Attributes:	
`name` (string)	name of site
Methods:	
`metrics`	outputs list of all metrics measured at the site
`devices`	outputs list of all included devices present at the site
`near [x] [z] [r]`	outputs list of all devices within r meters of (x,z)
`nearest [x] [z] [n]`	outputs list of n devices nearest (x,z)

A.1.3 [chain.device]

Attributes:	
`name` (string)	name of site
`device_name` (string)	name of device
`autoupdate` (bool)	object outputs on update events
`deviation` (bool)	object outputs data normalized by site meand std
`historical_interval` (int)	if > 0, device will only output at most one value per interval in historical data
`historical_downsample_rule` (string)	mean: value per historical_interval by averaging. interp: value per historical_interval by interpolation.
Methods:	
`bang`	outputs current sensor values
`metric [s metric_name]`	outputs current value for specified metric
`metrics`	outputs list of metrics measured by this device
`location`	outputs device location in Unity meters
`geoLocation`	outputs device location in Latitude Longitude
`data [s metric_name] [i start] [i end]`	outputs list of data from specified sensor from start time to end time according to downsample rule and interval.

A.1.4 [chain.time]

Attributes:	
`name` (string)	name of site
Methods:	
`parse [timestamp]`	outputs unix timestamp corresponding to ISO formatted string
`format [unix]`	outputs an ISO formatted string corresponding to unix timestamp
`now`	outputs current local time at UTC
`historical_now`	outputs historical "now" as set for attached site at UTC
`tod [seconds]`	outputs current time of day at UTC shifted by `seconds`
`historical_tod [seconds]`	outputs time of day at UTC shifted by `seconds` for historical

A.1.5 [chain.metric]

Attributes:	
`name` (string)	name of site
`metric_name` (string)	name of metric
`measure` (string)	interpolation, mean, median, max, min
`radius` (string)	radius of zone (all site if 0)
`interp` (string)	interpolation type from (proximal, bilinear)
`pos_x` (float)	x-coord of center point of zone
`pos_z` (float)	z-coord of center point of zone
`autoupdate` (float)	whether should output when dependent sensor updates
Methods:	
`bang`	outputs value of specified measure at specified point for specified metric

A.1.6 [chain.zone]

Attributes:	
`name` (string)	name of site
`enter` (float)	radius for devices entering zone
`exit` (float)	radius for devices exiting zone
`pos_x` (float)	x-coord of center point of zone
`pos_z` (float)	z-coord of center point of zone

A.1.7 [chain.data]

Attributes:	
`name` (string)	name of site
`interval` (float)	resampling interval (0 for no resampling)
`normalize` (bool)	normalizes data to 0-1 scale
`savefile` (string)	file to load on object load and to save to on autosave
`autosave` (bool)	saves to `savefile` on set
Methods:	
`bang`	output resampled data
`set [data..]`	save data received from `[chain.device]` in memory
`read [filename]`	read data from file on disk
`write [filename]`	write data to file on disk

A.1.8 [chain.map]

Attributes:	
`name` (string)	name of site
`pos_x` (float)	listener x-coord
`pos_y` (float)	listener y-coord (not represented)
`pos_z` (float)	listener z-coord
`ang_azi` (float)	listener azimuthal angle
`ang_ele` (float)	listener elevation angle (not represented)
Methods:	
mouse click	outputs selected device name

A.2 Abstractions

A.2.1 [chain.browser]

This object is a wrapper around `[chain.device]` which incorporates an instance of `[chain.info]` to present UI elements for selecting `device_name` and `metric`.

A.2.2 [chain.cache]

Attributes:	
`name` (string)	name of site
`cache_dir` (string)	directory in which to save cache files
Methods:	
`cache [device] [metric]` `[start] [end]`	looks for cache file for requested data; if it is missing, makes query and saves cache file; outputs path to cache file

A.2.3 [chain.timerange]

Attributes:	
`name` (string)	name of site
`tz` (int)	timezone to measure midnight from in seconds from UTC
Methods:	
`from [time] to [time]`	formats start, end pair for specified time range
`last [time]`	formats start, end pair for specified time range ending now
`next [time]`	formats start, end pair for specified time range beginning now

A.2.4 [chain.itertable]

This object is a wrapper around `[chain.data]` which presents a UI table of resampled data. A range of the data may be selected and an interal metronome can be used to iterate through the data.