Embodied Emergence: Distributed Computing Manipulatives

by

David Bouchard

B. Comp. Sc., Specialization in Computational Arts, Concordia University (2004)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2007

Author ____________________
Program in Media Arts and Sciences
August 20, 2007

Certified by ____________________ [illegible]
Patricia Maes
Associate Professor of Media Arts and Sciences
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by ____________________ [illegible]
Deb Roy
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Embodied Emergence: Distributed Computing Manipulatives

by

David Bouchard

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on August 20, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

Distributed systems and the emergent properties that can arise out of simple localized interactions have fascinated scientists and artists alike for the last century. They challenge the notions of control and creativity, producing outcomes that can be beautiful, engaging and surprising at the same time. While extensive work has been done using computer simulations of such systems in fields like artificial life and generative art, their physically embodied counterparts are still in their infancy, in part due to the complexity of building and deploying such systems.

In this thesis, I will discuss how simple tangible nodes can enable playful and creative experimentation with the concept of emergent behavior. Specifically, I will address how embodied interaction scenarios involving parallel systems can be implemented and how a range of sensing and actuating possibilities can be leveraged to generate novel and engaging experiences for the end users. In particular, the use of sound will be explored as a medium for representation. Finally, I will argue that there is value in making the transition from software simulations to a situated and manipulable instantiation of these concepts, both for the designer of a system and its users.

Thesis Supervisor: Patricia Maes
Title: Associate Professor of Media Arts and Sciences, Program in Media Arts and Sciences

Embodied Emergence: Distributed Computing Manipulatives

by

David Bouchard

The following people served as readers for this thesis:

Thesis Reader________________________________

V. Michael Bove Jr.
Principal Research Scientist
Program in Media Arts and Sciences

Thesis Reader________________________________

Mitchel Resnick
LEGO Professor of Learning Research
Program in Media Arts and Sciences

Contents

Acknowledgments

This thesis would not have been possible without the contribution of many people at the Media Lab and elsewhere whom I must thank before anything else.

First and foremost, my advisor, **Pattie Maes**, for giving me the unique opportunity to work at the Media Lab and for being supportive of my ideas. My thesis readers, **Mike Bove** and **Mitchel Resnick**, for taking the time to provide me with useful feedback, advice and insight.

My colleagues in the Ambient Intelligence Group for their fine friendship and support. In particular, **Sajid Sadi**, for his invaluable help, wisdom and perspective and **David Merrill**, for his infectious enthusiasm and getting me to combine music and electronics. Also, **Enrico Costanza**, **James Chao-Ming Teng**, **Hugo Liu**, **Orit Zuckerman**, **Marcelo Coelho** and **Pranav Mistry** for all the good discussions and collaborations.

Orkan Tehlan, for bearing with my obsessive code organization. **Jeevan Kalanithi**, **Graham Grindlay** and **Owen Meyers** for the epic jam sessions. Also, **Mark Feldmier** and **Josh Lifton** for making said sessions possible. My fellow students as well as the staff and faculty for sharing their passion, knowledge and resources, particularly the **Responsive Environment** folks.

Tom Hayes and **Paul Horowitz**, for being a source of inspiration and for teaching me electronics.

Jason Lewis and **Joey Berzowska**, for helping me get where I am today.

Ma famille, pour leur soutien inconditionnel.

And finally, **Adrienne**, my love, for everything.

Chapter 1

Introduction

"Emergence is what happens when an interconnected system of relatively simple elements self-organizes to form more intelligence, more adaptive higher level behaviour."
– *Steven Johnson* [1]

Emergence is a concept that has been captivating scientists, philosophers and artists alike for the past century. The concept traces back to John Stuart Mills in his 1843 treaty *A System of Logic* [2]. While Mills did not explicitly refer to emergence, he defined the idea as an irreducible cause which cannot be traced back to its individual components. The notion of emergence was originally applied to examples derived from chemistry, but later carried over to philosophy and the humanities, for instance providing a basis for theories about city development or marketplace fluctuations.

There is indeed something fascinating and powerful about the notion that simple elements of a system can somehow produce an effect which is more than the sum of its individual constituents. Nature provides us with a multitude of examples of such systems. For instance, groups of cells, birds or fireflies all exhibit some kind of collective intelligence in which no central coordination actually takes place. These systems have inspired algorithmic models for a range of software simulations and applications.

1.1 Motivation

With the advent of powerful microcontrollers and sensors, that have become extremely small, cheap and ubiquitous, emergence models inspired by nature are slowly making their way out of software simulations back into physical forms. From a computational standpoint, these systems share some of the properties of distributed computing, albeit perhaps not in the traditional sense. While distributed computing is typically defined as a method of processing in which different portions of a program run in a parallel over a network of computers, one can think of emergent behaviour as the program, with its constituents each carrying out their role in a decentralized fashion.

There are a number of practical incentives to explore the viability of these systems from an engineering standpoint. Because of their similarities with distributed computing, they share some benefits: the main one being scalability. Decentralization becomes a strategy for handling the challenges of system expansion. At least in theory, distributed systems can be very agile and can be designed to easily work at different scales. In a tangible hardware platform, this translates into the ability of an application to function with a small number of nodes, such as in a table-top interaction scenario; or as a much larger instantiation, such as a room-scale installation.

Furthermore, if properly designed, distributed systems can be made robust and sustain the failure of individual nodes. This becomes particularly important when taking into account the scaling factor. If a component in a centralized solution fails, the whole system fails. By contrast, a space or application designed with a large number of distributed nodes can keep on functioning so long as a sufficient number of nodes are operational. Perhaps the cost might be reduced performance, but the system will have some output nevertheless.

Emergent behaviour systems also carry the promise to allow the creation of complex behaviour from simple elements. The notion of using emergence as a strategy to manage this complexity is very attractive in an era where technology is becoming ever more complicated.

Finally, the creative possibilities presented by such systems have been explored over the years through a number of software platforms such as StarLogo[3] or the Emergence Engine[4]. Yet, both the notions of emergent behaviour and distributed computing remain neither intuitive nor trivial. We are still in the process of understanding these types of networks and what possibilities that they might have to offer. Therefore, there is a need for platforms which facilitate experimentation with these types of systems while providing users with a better of awareness and understanding of decentralized problem-solving.

1.2 Thesis overview and contributions

Initially, I will survey a number of projects falling under the theme of what I call *embodied emergence*. The survey will examine projects from a computational, educational and design perspective. I will use an expansive definition of the emergence paradigm as the coming together of systems in which no central controller is dictating the overall behaviour. Drawing from this rich and diverse body of work, I will distill some useful design principles for what might constitute an engaging and intuitive emergence platform.

I will discuss how these design principles informed the implementation of a tangible emergence and distributed computing platform situated at the intersection of the works presented in the survey. On the one hand, the existing educational tools shine with their simplicity and intuitiveness: they can be manipulated by the user to enable a deeper understanding of the emergence mechanisms and therefore encourage experimentation. On the other, some of these projects' designs push the limit of what can be accomplished using software simulations. Unfortunately, the user's involvement is often limited to an observation role, with very little direct manipulation. Also, their structure and designs tend to be fairly rigid, as dictated by the artist's intent, yet the nature of emergent behaviour is such that interesting and meaningful output often comes through serendipitous experimentation.

From an educator's perspective, the goal of this thesis is to demonstrate that there is value in combining all those elements in order to create engaging experiences in which users can gain a better understanding of the complex systems and the emergent behaviour principles at work. All the while, I will approach these systems from a designer's perspective with an interest in examining how this exploration can be made intuitive and playful, but also in looking into how we can leverage the system's behaviour to generate compelling audio and visual feedback.

I will then describe the production of a tangible platform, called Sound Mites, which enables a playful exploration of emergent phenomena. Sound Mites are tangible blocks which can communicate with one another. They generate light and sounds to create novel representations of emergent phenomenons. In addition, a software simulation framework used to experiment with decentralized behaviour will be presented, as well as the implementation of two applications written for the blocks. Finally, several other potential applications will be examined and future improvements to the platform itself will be discussed.

In summary, the contributions of this work include:

- A platform for tangible exploration of emergent phenomena, including robust, simple, wireless nodes which are cheap and battery powered;
- A software simulation framework to prototype applications for the platform;
- A set of applications for the nodes; and
- A review of user evaluations from a public installation of the system.

Chapter 2

Foundations

2.1 Embodied Distributed Computing

Distributed processing over networks of computers has been an active area of software research for some time now. However, perhaps due to recent technological advances but also to the promise of even greater things to come, only recently has there been interest in the potential of large systems of physically co-located computers.

The work of the Amorphous Computing Group laid the software foundations for such large systems of small distributed computers. Their work is one of the first instances of a system architecture and algorithms which are specifically targeted towards an embodied hardware platform [5]. The vision of amorphous computing is one of a dense assembly of physical, miniature processing nodes, each running asynchronously but with the ability to communicate locally and self-organize into more complex structures. This self-assembly enables the nodes to collaborate in solving problems which a single node could not otherwise solve.

Bill Butera has built upon these ideas and introduced the notion of a "paintable computer", which he defines as an agglomerate of numerous parasitically powered, finely dispersed and ultra-miniaturized computing particles [6] Under these conditions, the topology of a cluster can change unexpectedly and individual elements are so cheap that they become expandable. His reference architecture ex-

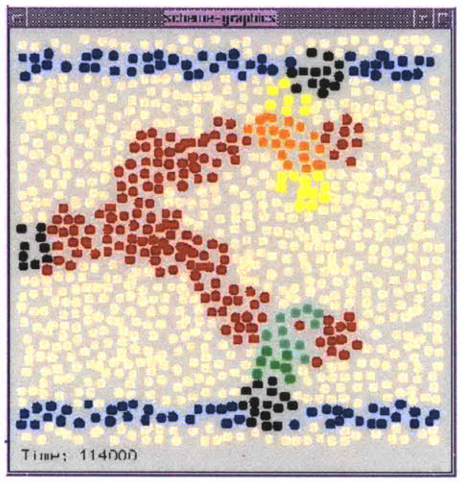


Figure 2-1: Amorphous computing simulation

tended the self-assembly programming model to include "process fragments", mobile autonomous pieces which can aggregate into higher-level processes. The usefulness of these fragments was demonstrated through extensive software simulations.

His work also highlights some of the core requirements and challenges of physical distributed architectures. By nature, such systems will have to be completely asynchronous, since there is no way to reliably synchronize so many clocks. Systems will need to be fault tolerant, since the autonomous and expandable nature of individual nodes implies that some of them are bound to be destroyed or simply fail at some point. Also, the nodes' ability to communicate will be constrained to their immediate neighbourhood. Moreover, the topology of the neighbourhood will be subject to change and cannot be predicted. Finally, nodes will have limited processing resources, which puts an upper bound on the size of the program for an individual node, pushing complexity upwards into the higher level processes. While these observations were made with paintable computers in mind, they apply in the broader sense to other distributed systems, even though they might be operating at different scales. For example, the work of Diane Hirsh on the Smart Architectural Surfaces tiles is a good example of a larger-scale system which addresses some of these issues, particularly asynchronous operation [7].

There are still relatively few actual hardware platforms which attempt to implement these specifications. However, Josh Lifton's Pushpins Computers, a distributed sensor network, follows Buthera's

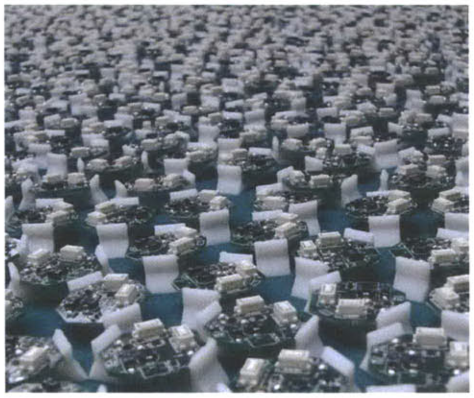


Figure 2-2: A paintable computer prototype

simulation work very closely. The Pushpins form a peer-to-peer sensor network comprised of several identical nodes. Each node supports local communication using a variety of modes (infrared, capacitive, coupling) as well as centralized control via an infrared beacon. Nodes can show their state using an on-board color LED. Expansion modules can add sensing capabilities to the nodes, including light sensors, sonars and microphones. Finally, the nodes are powered according to the pushpin metaphor by sticking them into a layered substrate providing insulated power and ground planes. Applications of the Pushpins have focused on taking advantage of the sensor modules, notably the light sensor, which transforms a group of properly equipped Pushpins into a simple retina capable of distinguishing boundaries between light and dark areas [8].

Figure 2-3: Pushpin computing

Berkeley's Smart Dust [9] is another early and famous example of such a hardware platform. The Smart Dust is comprised cubic-millimeter scale programmable computers complete with bidirectional optical communications, power supply and analog sensor circuitry, and is targeted at sensor network and monitoring applications. Unlike the work presented in this thesis, Smart Dust and the other projects mentioned above are neither designed for the average user, nor are they focused on how people might interact with such a technology.

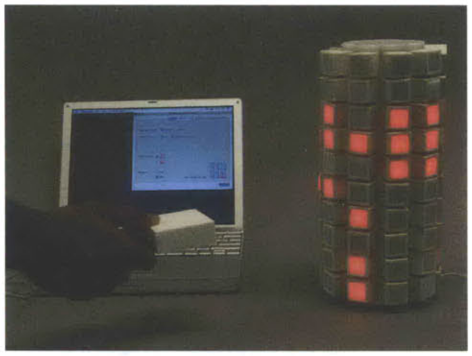


Figure 2-4: SmartTiles

2.2 Educational tools

While it is a software tool, StarLogo [3] has a prominent place in any discussion regarding educational tools for decentralized thinking. Expanding Logo's single turtle to a large group of turtles, the Star Logo language made possible the experimentation with emergent behaviour simulations and inspired much of the subsequent efforts in making these systems tangible.

Most of the educational tools described in this section draw on the theories of constructivism, which is a philosophy of learning founded on the idea that people build an understanding of the world by reflecting on their own experiences, generating new "mental models" in the process. This philosophy was put in practice through the introduction of *manipulatives* by educators such as Frederich Froebel [10]. Following in these footsteps and learning from the experiences of systems such as StarLogo, the Lifelong Kindergarten group at the Media Lab introduced the notion of digital manipulatives[11]–physical manipulatives augmented with computational capabilities to help children explore and discover some properties of the physical world. Their work spawned a number of digital toy projects, including a number with a focus on distributed computing (for a more exhaustive survey, see [12]).

One such example is the SmartTiles [13] created by education researchers at the University of Colorado, a computational tool which enables children to learn about distributed systems and emergence in a tangible way. As the focus of the tiles is to introduce the concepts to children, their capabilities are constrained to a specific behaviour (in this case, Conway's Game of Life [14]), a

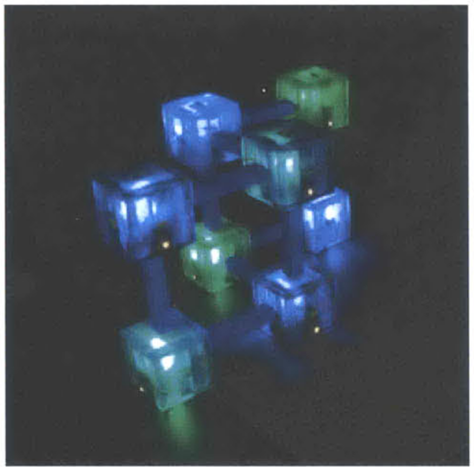


Figure 2-5: Boda Blocks

shape (a grid over a carpet providing power) and an output modality (each tile has a lamp which can be either on or off). The SmartTiles consist of a handful of nodes; yet, the authors refer to them as early prototypes of room-sized artifacts, hinting towards larger, more complex and expressive systems with the ability to scale up and integrate with their environment.

Similarly, Leah Buechley's Boda Blocks is a kit of blocks cast out of resin and outfitted with an LED and a microcontroller [15]. Blocks can be connected to each other to form 3-dimensional structures. A Control block provides a serial link to a computer. It synchronizes the blocks together and allows the user to carry out cellular automata simulation throughout the structure.

A few years ago at the Media Lab, Kwin Kramer explored a related space in a thesis entitled "Movable Objects, Movable Code" [16]. His project, called the Tiles, examined how programs could reside not only in a single computer, but on a network, and the resulting patterns that could emerge out of interactions between programs moving from one tile to the next. While Kramer's Tiles' focus is different than the SmartTiles, they share some design principles: simplicity, intuitiveness and playfulness. Kramer also contemplated that the Tiles could allow for a deeper exploration of the output capabilities of the platform and suggested the development of a more extensive I/O layer for the Tiles.

Oren Zuckerman's System Blocks is a construction kit which allows children to explore and model dynamic systems. The kit is designed in a fully distributed fashion. Blocks are connected to their

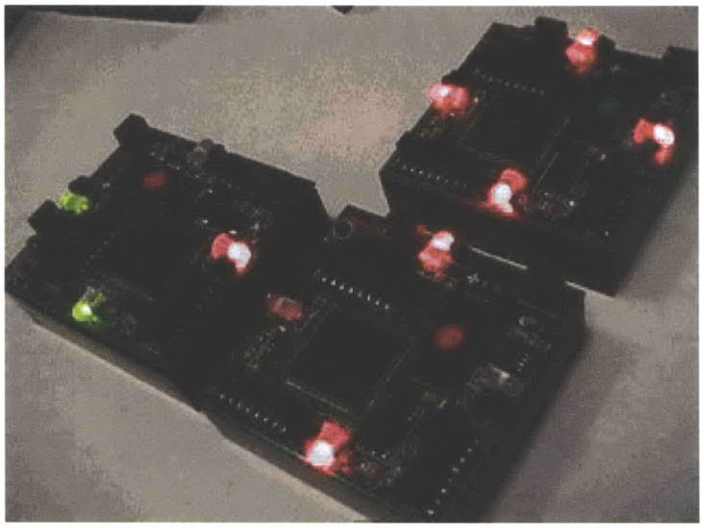


Figure 2-6: Movable Objects, Movable Code

neighbours via cables and, while they are not directly aware of each other, they can accept input, process information, and output the result. This design scales well without making any compromises in performance and provides a robust infrastructure in which each block is assigned a specific behaviour, such as value generation, accumulation or arithmetic operations. Furthermore, this design is easily extensible – simply add a new block capable of a different behaviour. For instance, in order to improve representation of the system, some blocks can provide a rendition of the values passed through them using a variety of output devices such as graph displays and motor actuators. Evaluations of the system with young children showed, amongst other observations, that a physical interface which provides multiple representations (sound, motors, lights) is engaging for the user [17].

Senspectra by Vincent Leclerc is another computationally-enhanced modeling toolkit, designed for sensing and visualizing structural strain. [18] The system works as distributed sensor network comprised of processing nodes and flexible sensing joints; it allows users to assemble 3D models by direct manipulation while giving them real-time feedback about strain on the structure itself using LEDs embedded in each processing nodes. A node connects the structure to a computer where the structure can be visualized as well.

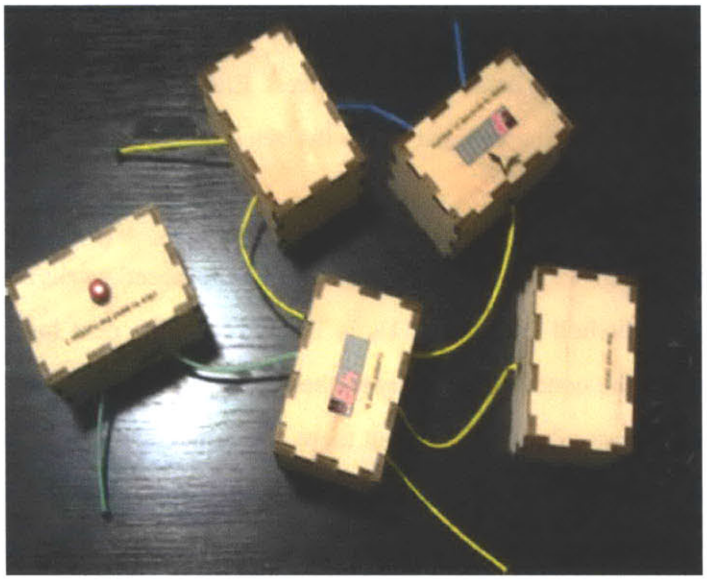


Figure 2-7: System Blocks

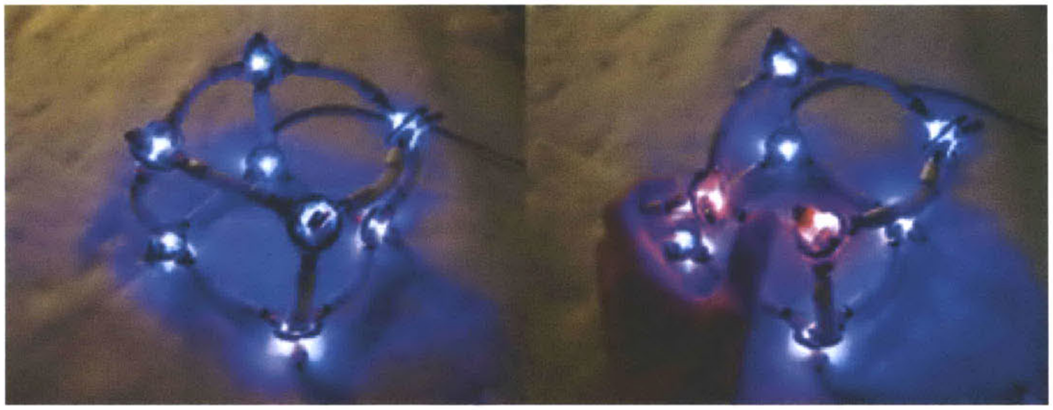


Figure 2-8: Senspectra

2.3 Design and Aesthetics

Artists have also found the idea of emergent behaviour appealing. The concept has been central to fields such as artificial life and its artistic derivatives. Emergence implies something novel and unanticipated, and as such can be thought of as the *reward* which draws artists to explore this bottom-up approach to creation [19].

The last few decades have given rise to a rich body of software-based works such as Brian Knepp's Healing Series [20] or Mitchell Whitelaw's Us is Them [21]. The next logical step involves the transposition of some of the concepts of emergent behaviour into the physical world; not only are the creative possibilities certainly intriguing, but there are immediate benefits in applying the ideas to physical installations in terms of scalability, maintainability and robustness in a time where new media projects are becoming increasingly complex. However, perhaps the most important idea is to leverage emerging patterns to generate surprising and unpredictable behaviours as opposed to scripted action/reaction interaction models.

Some have already begun to explore this area. Their projects set themselves apart from the aforementioned educational platforms from an aesthetic standpoint. These works will be used as references in order to examine their designs and how they approach embodied emergence while intentionally leaving out a discussion of their artistic qualities, which is not directly relevant to this thesis.

Our first example is the Bion project [22]. The installation consists of 1,000 sensor nodes that are programmed to react to visitors and nearby nodes. Visitors trigger wave-like patterns of lights and chirping sounds when approaching the swarm, until the nodes become accustomed to their presence and integrate them into their ecosystem. In their paper [9], the authors highlight the expressive significance of the nodes' spatial configuration as well as the network itself as a product of the nodes and the visitors. Both elements contribute to elevate the piece to an experience beyond what could be previously achieved on a computer monitor.

Emergence is a central component in the work of Ken Rinaldo. For instance, his Autotelematic Spider Bots is an installation consisting of 10 spider-like bots which interact with the public in

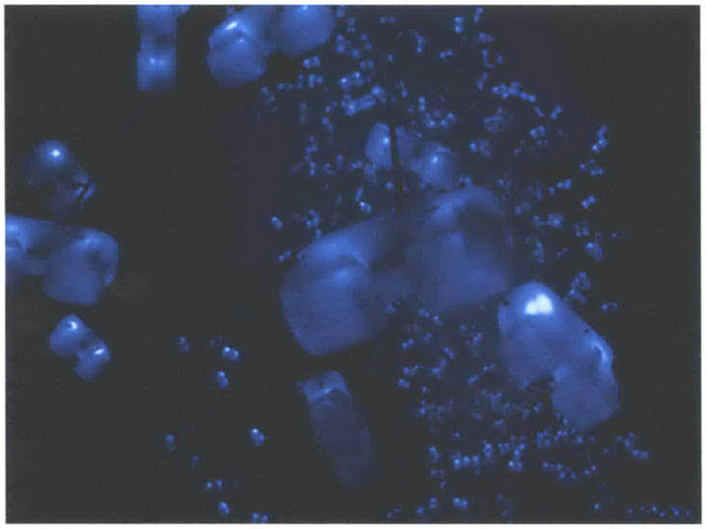


Figure 2-9: Bions

real time and self-modify their behaviour, based on interactions with the viewers, themselves, their environment and their food source. The bots can "see" the participants through ultrasonic eyes at a distance of 3-4 meters. The Spider Bots were developed as an experiment to discover if digital pheromones could allow the robots find their own food source, like in ant colony. In this case the food source is a charging rail and the pheromone a 1HZ infrared beacon signal.

An earlier, more straightforward instance is Bill Vorn's Evil/Live 3 [23]. The project implements a cellular automaton on a 16x16 matrix of halogen lights. Each light has a sound associated with it, such that light patterns also generate rhythm patterns and variations. A computer vision interface enables viewers to toggle the lights by projecting an image of the space onto the grid. The lights are controlled over MIDI by Max/MSP and the soundtrack is played back through a speaker behind the array. This project illustrates how a simple cellular automaton can be magnified out of proportion for a much more dramatic effect on the viewer. The implementation, however, is modeled after a centralized simulation process. This type of system is amenable to expansion or recreation in a truly distributed fashion.

Simon Penny's Petit Mal challenges and critiques the conventions of robotics. This piece is an autonomous robotic sculpture with locomotion capability and a number of sensors. The robot reacts and interacts with viewers in an ambiguously aggressive way. While it is not a distributed system,

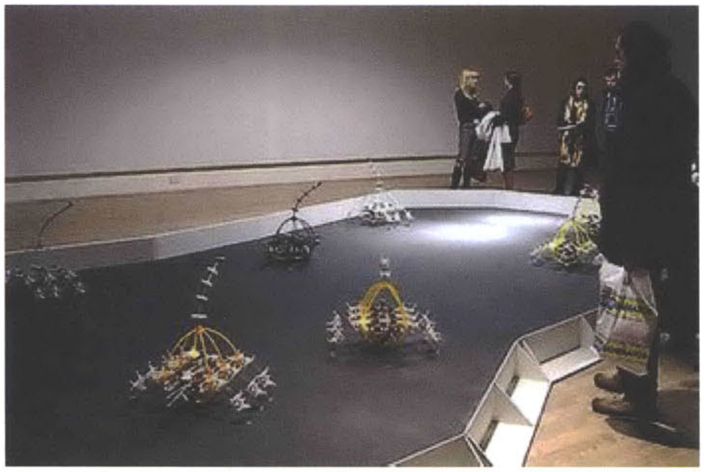


Figure 2-10: Spider Bots

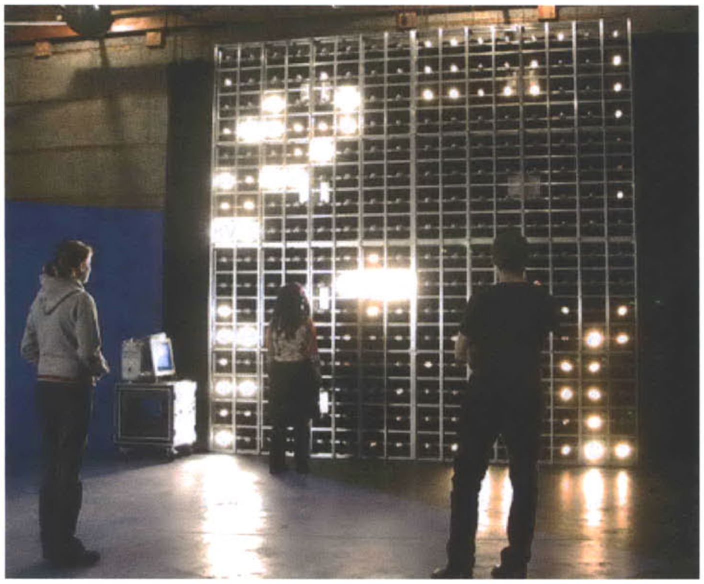


Figure 2-11: Evil/Live 3

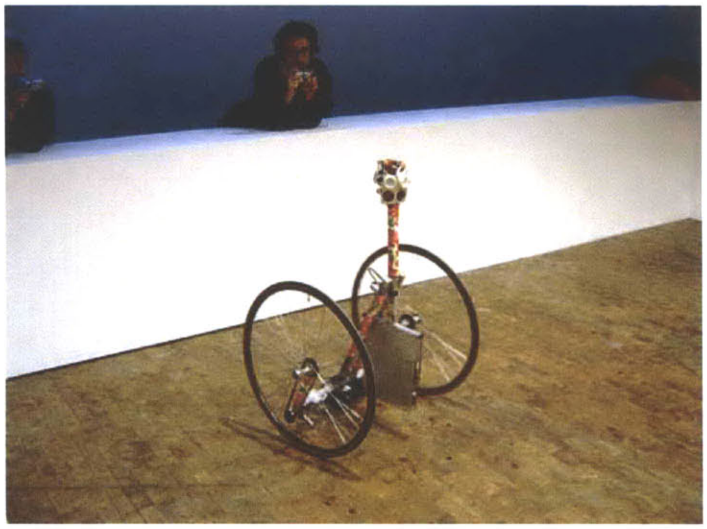


Figure 2-12: Petit Mal

this project embodies the notion of emergence on a different scale. The behaviour of the robot is the result of individual functional units for motion and sensing acting in an uncoordinated way. Penny stated that he intended for the "software to emerge from the hardware" [24].

The Ping Genius Loci project [25] by the Aether Architecture collective is also worth mentioning. The installation features a grid of 300 solar-powered analogue pixels that are used as actuators and sensor nodes. Although the project is not a true distributed system (rather, every node is controlled by a central computer), it suggests an interesting potential for more kinetics based explorations of emergent behaviour. It also demonstrates how simple nodes can be sustainable from an energy standpoint while remaining effective in public spaces. Finally, it highlights one of the desirable advantages of decentralized control, as it could potentially become quite scalable if its nodes were to self-organize and become truly autonomous.

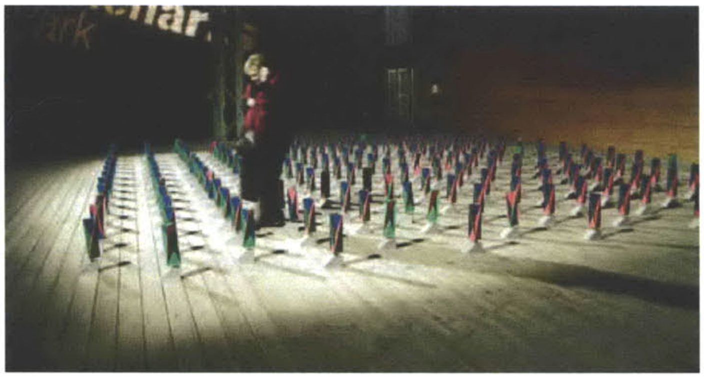


Figure 2-13: Ping Genius Loci

Chapter 3

Design considerations

3.1 Guiding Principles

In light of the foundations reviewed in the previous chapter, I set out to define a set of principles which have guided and informed the design process of the final product. My goal is to define what might be constituents of a playful, intuitive and engaging system to explore the notion of emergent behaviour.

3.1.1 Tangibility

The nodes should embody the principles of distributed computing and emergence in a tangible way. Tangibility has some advantages over software implementations when it comes to intuitiveness and playfulness, because it opens up a world of affordances from the physical world, in terms of the node's shape, materials and textures. It enables a direct manipulation of the system as well. For instance, it is easy to quickly make adjustments to the topology using both hands. Finally, projecting a simulation unto the physical objects also has the benefit of allowing multiple people to interact at the same time around a set of blocks.

3.1.2 Interactivity

The system should be interactive. This is partly derived from the previous rule, which implies manipulation. In combination with their tangible nature, interactive nodes enable coincidental input and output, where feedback occurs on the interface itself thus reducing the level of indirection, as illustrated by a project like Senspectra. Interaction also fosters exploration. By allowing the nodes to sense input from the user, we can create a system which is more responsive and thus more gratifying to interact with. Action / reaction mechanisms can provide some grounding point for more abstract concepts, for example by allowing the user to control the placement and amount of "seeds" which are driving a distributed algorithm simulation. Furthermore, the system should be able to respond in real-time, in order to provide a satisfying interactive experience for the user.

3.1.3 Autonomy

In order to remain true to the spirit of decentralized control, the nodes should be implemented in a fully distributed and autonomous fashion. This means that each node should be independent from one another, and should not be dependent on the presence of a particular neighbour node to perform specific processing.

To strengthen the user's perception of their autonomy, nodes should be able to communicate wirelessly. There is something about having physical connections which suggest that there is also some hidden clock somewhere, or a process running the show, such as in the Boda Blocks. I believe that making the nodes truly autonomous will have a more powerful impact on the user when it comes to the awareness of their decentralized behaviour. While under some conditions connectors can make sense, such as in System Blocks where a connector represents an edge in a flow graph, the connectors can still become more of a hindrance when trying to perform some quick manipulations. Finally, physical connectors tend to fail and can be a source of frustration.

The lack of connectors also implies that the nodes should be self-sufficient when it comes to power. Unfortunately, inductive power remains hard to achieve for that amount of power, leaving a power substrate, such as the Pushpin's, or battery power as viable alternatives.

3.1.4 Simplicity

For a number of reasons, the nodes should remain as simple as possible. One of the most appealing ideas about emergence is that the notion that a simple model of local interactions can generate a complex, higher level behaviour. Nodes should reflect this in their architecture as well as in the complexity of the computation that they can carry out.

Simplicity also means that the nodes should remain low cost. This principle contrasts with some existing distributed hardware platforms such as Sun's SPOTS (which will be discussed in further details in Chapter 4). In order to make a tangible distributed simulation interesting, a high number of nodes should be made available. This is completely impractical if the cost of each individual node is too high. Furthermore, in the context of this project, keeping the nodes simple also means putting an upper bound on assembly labour. This ensures that the manufacturing of dozens of nodes will remain within reach and possible to accomplish in a timely fashion.

3.2 Application design

One of my chief interests in producing this work is to explore how distributed computing systems and emergent behaviour can be leveraged to create applications that are unique and compelling. The visual potential of such systems has been well explored through numerous software programs, and also in physical setups through some of the projects discussed in the introduction. However, there is a realm of output possibilities which goes way beyond blinking LEDs and light patterns. To reflect this, this project uses a combination of light and sound as its output mechanism.

The choice of sound as a medium is informed by past experiences in software simulations. Sound and emergence have come together in a number of successful projects. There are several pieces of software that have shown how emerging patterns can be used to generate compositions [26, 27], granular synthesis, sonic ecosystems [28], and collaborative music composition between a human musician and a multi-agent system [29]. An LED light as output remains present on the device mainly because its implementation is well understood, but also because it contributes to creating a

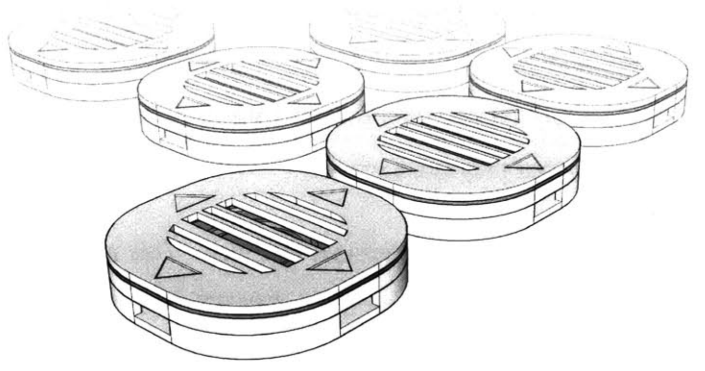


Figure 3-1: Early application sketch

richer experience and can act as a counterpoint to the sound, especially in noisy environments.

While the aforementioned projects are interesting, there is room for a more tangible exploration of sound generation by means of emergent behaviour. This thesis pushes the notion of sonifying emergence a step further by implementing a network of small autonomous nodes, dubbed Sound Mites. The nodes are to be deployed on a surface, such as a large table or a wall. Each node plays a note through a small speaker in accordance to the state of the nearby nodes and a set of programmed rules. The number of possible states and their associated notes are confined to a musical scale to avoid cacophony. The nodes also provide some visual feedback of the underlying pattern movements using the internal color LED reflecting the nodes state. Because the nodes are tangible, the user will be able to reconfigure the node topology and explicitly toggle their state by manipulating them and re-arranging them on the surface. I make the assumption that the resulting sound patterns will have a unique quality which can only be revealed through an embodied implementation where many sound sources are actually physically distributed in space.

While several types of applications for the blocks will be exposed in Chapter 5, this thesis will focus on a discussion of two main categories in greater detail, each illustrated by a specific demo application. The first category is emergent behaviour systems. Here, the concepts at work tend to be more abstracts, such as in cellular automata systems. Behaviour is slightly harder to develop for, because of the random nature of emergent systems. To demonstrate this type of system, the nodes implement a reaction/diffusion algorithm which creates shifting local gradients through neighbour to neighbour communication. Randomness will be minimized by enforcing a set of predictable rules which can be recognized by the users. The second application category is a construction kit system. Here, we will develop a set of mappings which allows a set of identical blocks to collaborate into forming an instrument of sort, a synthesizer capable of producing organized sound loops and patterns.

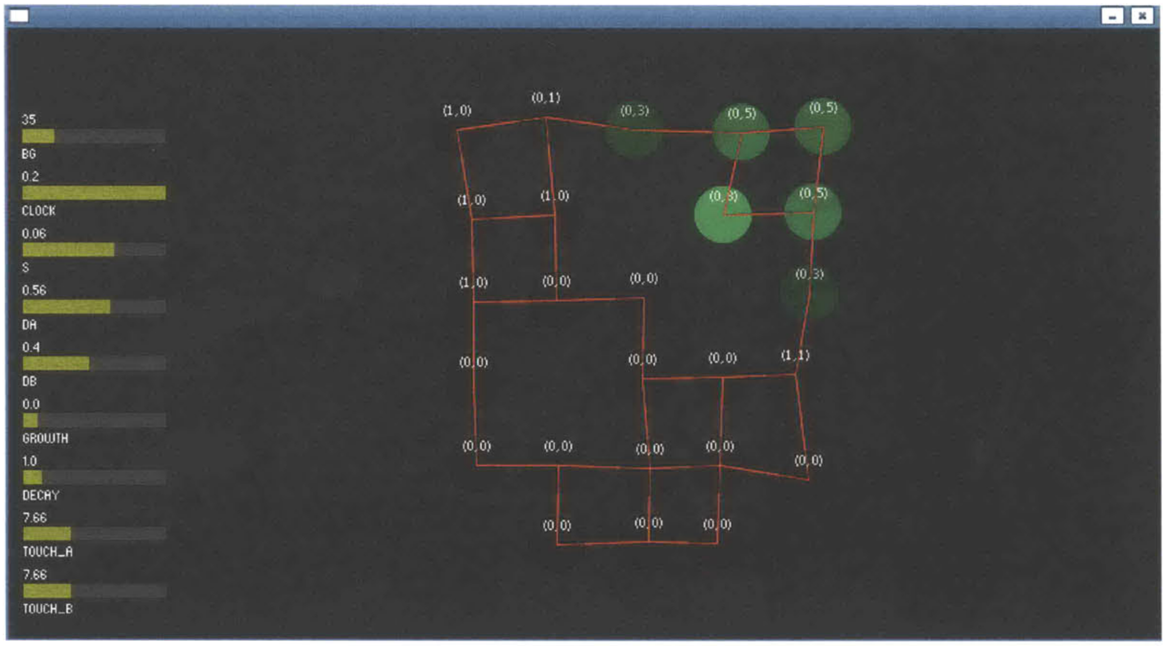
Chapter 4

Implementation

4.1 Software Simulator

One of the drawbacks of implementing a distributed system in hardware is that it significantly complicates development and maintenance of the system. For instance, there are entire research projects devoted to over the air programming and propagation of code to the nodes of a network. In order to sidestep some of these challenges, and make the development of behaviours for the system easier, I developed a software simulation framework which enables me to prototype some ideas before implementing and deploying them as a firmware into each node. The simulation framework, while not an accurate representation of reality, models some of the constraints which apply to the nodes, and allowed me to tweak of the system towards a good approximation of how the actual blocks would behave.

The simulator is written in the Processing programming language [30]. Processing is a Java-based programming environment which encapsulates some useful functions for quickly prototyping graphical applications. The simulator represents each block as an autonomous object with its own internal clock, rendered as an animated pie slice. Blocks are added one by one into the simulator – clicking anywhere in the drawing area creates a new block. This is similar to the way they were envisioned to be used in a physical space. Events, such as neighbour discovery, message passing or internal



35
BG
0.2
CLOCK
0.06
S
0.56
DA
0.4
DB
0.0
GROWTH
1.0
DECAY
7.66
TOUCH_A
7.66
TOUCH_B

Figure 4-1: Simulator

processing occur at fixed clock intervals. However, since blocks are created at random times, their clocks are not synchronized with one another. Local communications are simulated using a simple collision detection algorithm. Each block has a virtual sensor on all four sides which can lock into communication when it gets within the cone of influence of another block's sensor. Communication between blocks is visualized using connecting lines. The collision detection routine also allows blocks to be dragged around and pushed against one another, as it is might happen during a real-life interaction scenario.

When it comes to information processing, each block runs an identical short code snippet, which represents the behaviour. The state of the block is reflected by its color, so I can get an idea of what types of patterns are created by a given algorithm. Constants and delays can be tweaked using an on-screen GUI, as well as the global clock increment speed. There was an attempt to integrate the sonic behaviour of the blocks into the simulator using a multi-channel sound synthesis library for Processing called ESS. However, performance was not adequate and it quickly became apparent that the language was not designed to handle such a large number of individual sound generators. This reinforced my belief that while the visual output was easy to obtain in a simulated environment, the quality of the sonic texture was difficult to prototype in software and would only really manifest itself in the physical incarnation of the system.

4.2 Sound Mites

Sound Mites are the blocks which form the basic unit of this thesis work. They were dubbed as such because of their ability to produce synthesized sounds, yet, like a tangible pixel, they can also emit a full-spectrum colored light. Each block is capable of computation and is equipped with a touch sensor and four infrared (IR) transceivers for neighbour-to-neighbour communications. The blocks are battery powered and fully autonomous.

Existing infrastructures have been considered to fulfill the Sound Mites' core role. Wireless sensor network platforms such as the Berkeley MOTES [31] or Sun's SPOTS [32] come to mind as a good fits for this type of application, because they support a wide range of extensions and are capable of wireless communication. Interestingly, Sun's SPOTS is even being marketed as a platform for creative expression – so far, some of the highest profile applications for the platform came through Sun's first artist-in-residence program as well as their sponsorship of a class at the Art Center College of Design [33]. Unfortunately, while these platforms are attractive because a lot of the implementation work has already been done and they provide a number of useful features, their cost is still fairly high (a few hundred dollars a piece plus development kits), which makes them unreasonable choices for systems such as this one where at least 40-50 nodes are envisioned to coexist on a small surface. Beside the price, they are also perhaps too complex for a system focused on simple interactions within large groups of simple-minded objects. Thus, like Kramer, I reached the conclusion that creating a custom platform was in order.

4.3 Enclosure design

One of the first decisions to make in the enclosure design was the required dimensions. While the current tendency for electronics is to make everything as small as possible, the benefits for doing so are not always obvious. Besides, the aim of this project was not to create something like the

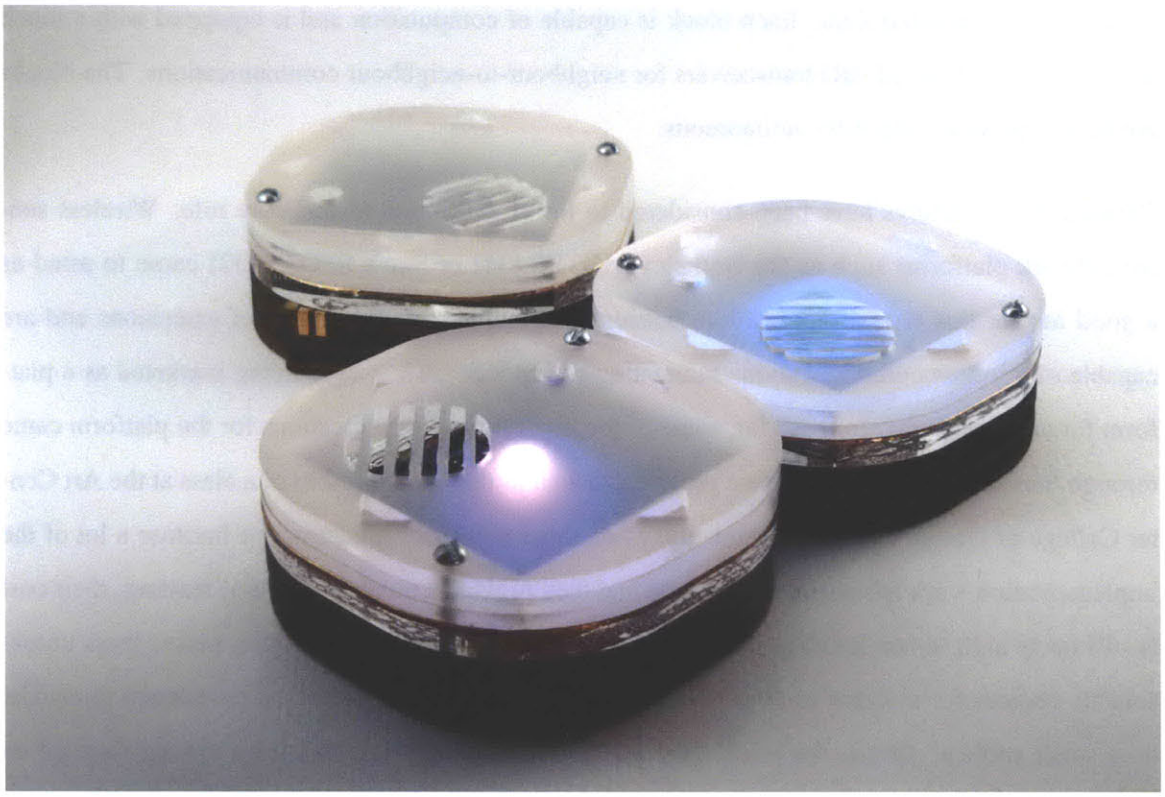


Figure 4-2: The Sound Mites blocks

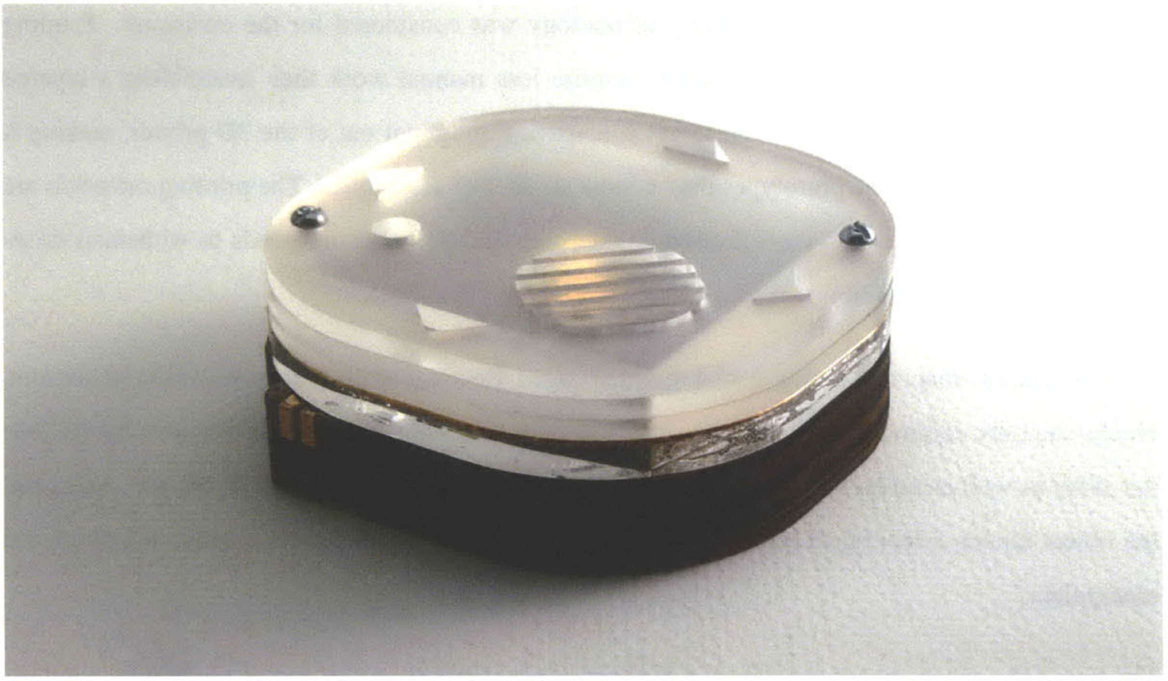


Figure 4-3: Enclosure closeup

paintable computer. The size of each node would have to strike a balance between being small enough to easily manipulate, yet big enough so that they could cover a sufficiently large surface in order for the spatialized sound effect to be felt. I determined that the ideal size for a block was roughly that of a hockey puck, or about 2.5" x 2.5". Size considerations also imply compromises between low cost, discrete components and smaller integrated parts which can be more expensive. The selected form factor allows for enough room on the circuit board for cheap, commonly available parts to be used.

The enclosure is built using a layered structure of plywood and acrylic. Layers are glued together with the exception of the top cover, which is held in place using two screws in order to allow access to the board for repairs or programming. The use of wood as a material not only provides a nice feel and texture to the object but is also cheaper and more environmentally friendly. Acrylic layers were used when required. A layer of clear acrylic in the middle of housing acts as a window lens for the IR sensors. The top layer is made out of frosted acrylic since the material provides good light diffusion even at short distances from the light source.

As an alternative, the use of 3D printing technology was considered for the enclosure. Printing is attractive, and appears at first glance to require less manual work than assembling a layered structure. It is challenging, however, to get consistent throughput out of the 3D printer, making it more suitable for one time prototypes than a large number of assemblies. The printing materials are also much more fragile than their layered counterpart, and the structure needs to withstand drops and relatively rough handling.

The enclosure is shaped as a square with rounded edges. Because of the placement of the IR sensors, blocks can only communicate with each other orthogonally. This constraint is hinted at by the four flat sides as well as arrows on the faceplate pointing in the direction of each IR sensor. However, the object retains a feeling of roundness and smoothness, which makes it pleasant to look at and manipulate.

The capacitive touch sensor consists of a layer of copper tape placed on top of the acrylic window, with a small flap on the inside which has to be soldered to after the circuit board is introduced in the housing. The copper tape is slightly larger than the acrylic layer which it covers, causing it to spill by a small amount all around the enclosure, thus providing a large enough surface for reliable sensing. The placement of the sensor (around the block rather than on top) takes advantage of the fact that form factor of the block affords a certain way of handling it.

The faceplate design also includes a small hole for access to the on/off switch and a set of slots above the speaker for better acoustics. The reset switch has been made deliberately hard to reach to prevent accidental resets and power off during use.

The enclosure also includes a charging connector made out of two thin strips of copper tape which loop around one of the wooden layers. The strips are placed such that they line up with two large pads on the circuit board. They can be directly soldered to once the board is placed within the enclosure, allowing for external connections to a power supply. In order to be charged, blocks are placed on wooden trays. A spring connector holds the block in place when inserted into a slot while making electrical contact. Blocks can be very easily and quickly placed onto or removed from the tray.

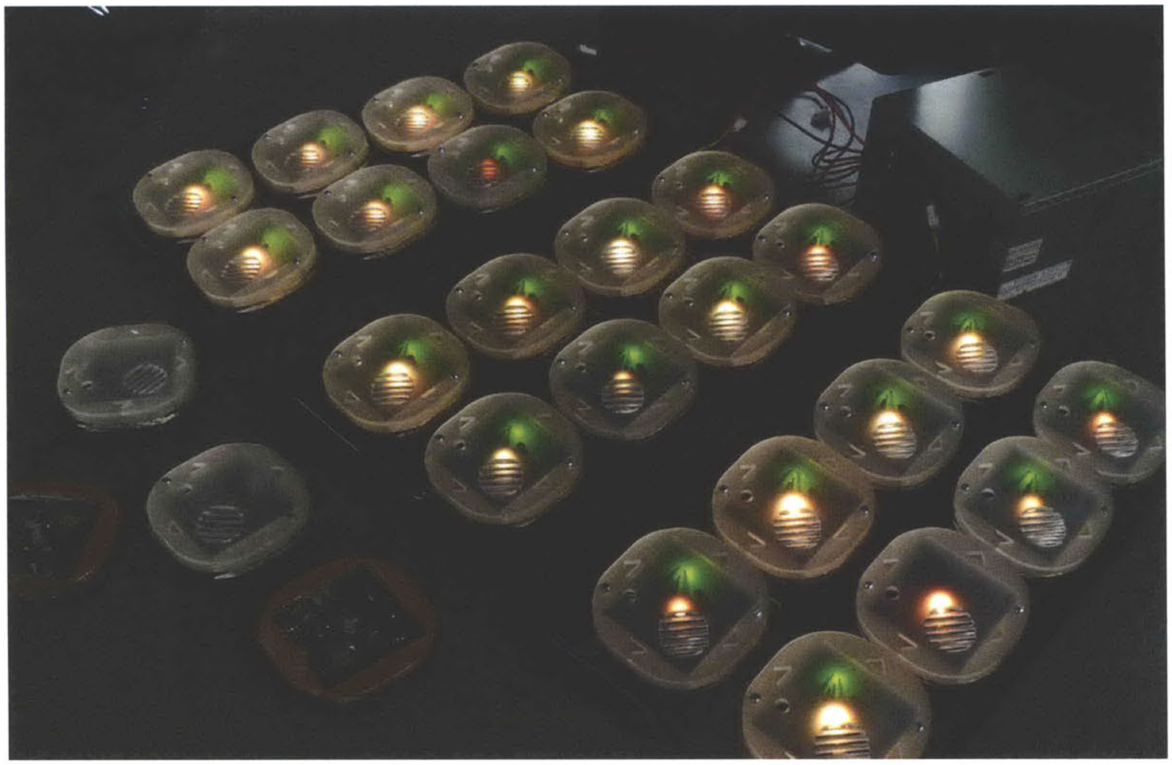


Figure 4-4: Charging trays

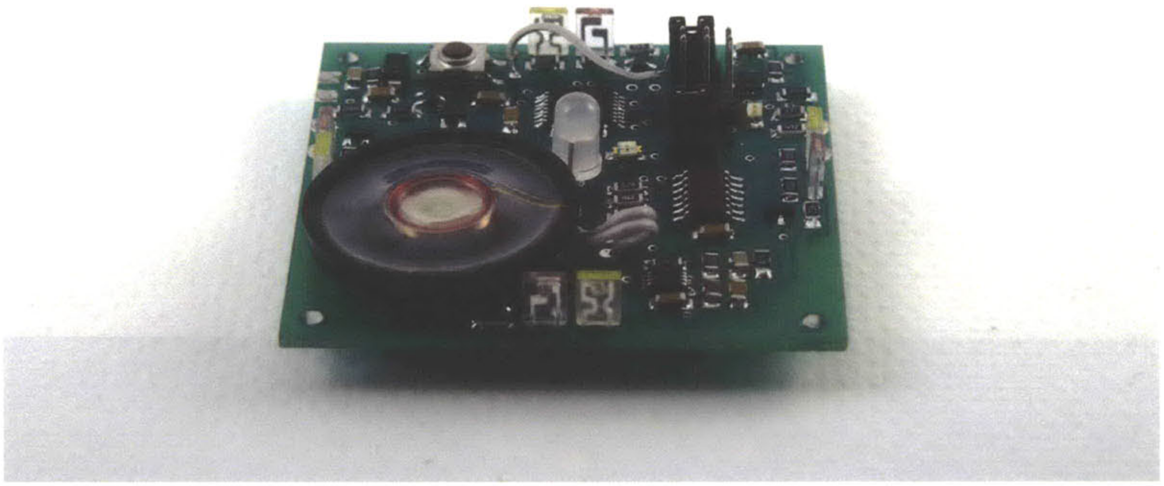


Figure 4-5: The circuit board

Finally, the original design implied a table-top based interaction. However, the addition of four tiny earth magnets press-fitted onto the bottom of the case allow blocks to be deployed on a vertical surface as well (which proved to be a lot more compelling – more on this subject in Chapter 6).

4.4 Circuit design

The original intent for this project was to provide a modular architecture, similar to the Pushpins or Kramer's Tiles, where the output modules are detached from the main processing board and can be replaced for different applications. However, after some deliberation, it became clear that extensibility was not necessarily a desirable attribute for a project of this scale. Dividing the circuit into two distinct boards added to its complexity and introduced the need for a robust connection scheme between the two. It essentially ended up increasing the cost of each block significantly for no immediate benefit. Besides, in all likelihood, the blocks would probably not be reused for a different purpose than this thesis, as hardware evolves very quickly. In light of these considerations, I made the decision to forgo the modularity of the platform and to focus on the outputs and sensors I was actually going to use, saving some money and time in the process.

The circuit architecture, however, retains some of the original design's modularity. The processing and communication (the "brain") module remains decoupled from the I/O module (the "hindbrain"), with each getting a dedicated microcontroller. The brain accesses the hindbrain's functionality by communicating with it over SPI using a simple command-based protocol. In order to save space and parts, both microcontrollers can share the same in-circuit programming (ISP) header by placing a jumper on each programmer's clock lines, allowing one microcontroller to be programmed while the other is disconnected.

The brain module is powered by an Atmel Mega AVR 8-bit microcontroller with 8 kilobytes of program memory and 1 kilobyte of RAM. It is clocked externally using a ceramic oscillator calibrated at 20 MHz. The AVR was chosen because it offers great performance at a relatively low cost but also for the availability of a good free and open-source tool chain to program it. There is also a fair amount of in-house expertise within our own research group with this particular microcontroller, which makes it an obvious choice. The brain is basically responsible for handling communication with nearby blocks, and then for processing this information using the set of local rules dictated by a particular application. Each IR sensor is allocated an output pin for sending data and one of the four available dedicated interrupt pins for receiving data. The hindbrain is powered by an AVR Tiny microcontroller, which has less program memory (2 kilobytes) and RAM (128 bytes) but offers similar performance to his larger brother. It is clocked at the same frequency as the brain. The hindbrain is responsible for controlling the I/O circuitry.

4.4.1 Infrared sensors

The infrared circuit is made out of discrete components. A few integrated solutions were examined at first, but dropped for a number of reasons. Small, IrDA compliant modules could have been a nice solution, but their individual cost was too high to be viable. Infrared receivers for remote controls were also considered. While they are generally cheap, ubiquitous and robust, they have the extra requirement that the data signals must be modulated over a carrier frequency. Multiplexing the PWM module of the microcontroller so that each side could transmit independently added extra components and complexity to the circuit which negated the benefits of these receivers. Fur-

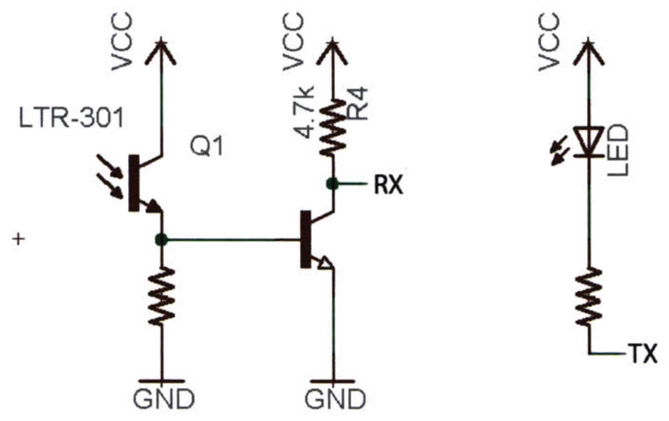


Figure 4-6: Infrared transceiver circuit

thermore, the switching speed of these modules is quite low, and that puts some restriction on the bandwidth which could become problematic if the blocks begin to require frequent exchanges of data. As such, a discrete IR receiver was implemented, made out of an IR sensitive phototransistor in a current amplifier configuration. The sensed signal pulses are then amplified to clean logic levels using a FET acting as a comparator trigger. The transmitter is simply an IR LED with wavelength characteristics matching the receiver. The LED's power is controlled by a resistor in series. Both the transmitter and the receiver are housed in convenient side-looking packages, which are perfect for this application. This solution, while not the most robust in terms of noise immunity, has the advantage of being very low cost and easy to solder.

4.4.2 Touch sensor

The touch sensing circuit is extremely simple and relies on a loading mode capacitive sensing scheme. A single large pull-up resistor (1 Meg) is connected in parallel between the sensing surface and the power supply. The sensing surface is then connected to a pin of the microcontroller. Since the AVR's pins can alternatively act as both inputs and outputs, only one pin is required for sensing. First, the pin is set to output and taken high for a few microseconds, charging the sensor. The pin is then placed in high-impedance input mode and taken low, discharging the sensor. The presence of fingers on the sensor increases its capacitance, which in combination with the pull-up resistor affect

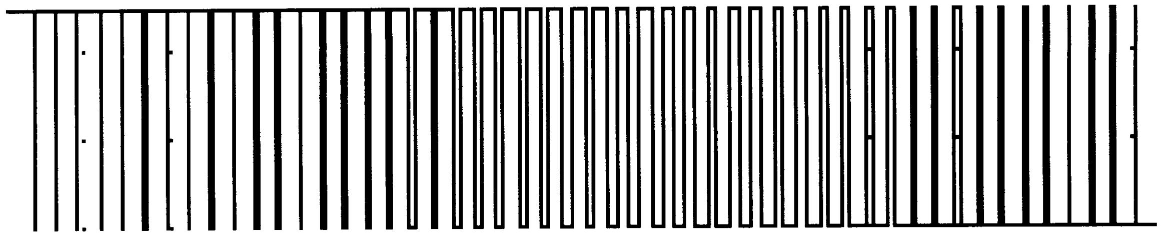


Figure 4-7: PWM signal used to generate a sine wave

the speed at which the voltage will charge on the input pin. By taking a readout of the pin after a fixed time interval (in this case, 4 microseconds), we can determine if fingers are present (pin reads low because it takes longer to charge) or not (pin reads high).

4.4.3 Audio Amplifier

The audio circuit uses a low-pass filter to transform a high-speed pulse-width modulated (PWM) train with variable duty cycle into an analog sine wave signal. The quality of the resulting wave is a trade-off determined by the low-pass filter which must be low enough to minimize ripple but high enough to not attenuate the audio frequencies too much. At the operating clock speed, the fastest PWM signal that the AVR can generate is around 70 KHz. This allows us to use a simple RC filter with a corner frequency of 1.94 KHz, which keeps the ripple at an acceptable level. The sine wave is then fed to a low-power audio amplifier which drives a small inductive microspeaker. The amplifier used is a LM4889 by National. It can provide 400mW of power into an 8 ohm load at 3.3V. This amplifier was chosen because of its rail-to-rail operation, low-power characteristics and minimal amount of required external components. It also features an ultra low-power shutdown mode (0.01 uA), which is useful when sound drops below an audible level or stops completely.

I experimented with a number of speakers for this circuit. The main factors in play were power consumption, cost, availability and frequency response in the operating range (300 Hz ~ 1600 Hz). My intuition was to use a piezo-electric. While they have the advantage of being low power, the frequency response for these speakers was unacceptable when driving them with anything other a square wave. There exists piezo speakers with good frequency responses, but they are hard to find and fairly expensive. After testing with a number of inductive speakers, I settled on a small 8 ohm microspeaker which provided the loudest output for the amount of power consumption at maximum volume (about 50mA). The fact that the speaker was available for extremely cheap at a surplus store was also a plus.

4.4.4 Power

One of the requirements for this project was for the nodes to be battery powered in order to give them some autonomy. The power circuit of the blocks is designed around a lithium-ion battery charging IC, the BQ Tiny 2410 made by Texas Instruments. The IC provides status lines and adjustable, regulated charging current for the battery. A MOSFET switch toggles between DC and battery sources to power the circuit's 3.3V voltage regulator. The charging current was set to 250mA within a range between 100mA and 1A – intentionally low to make sure the charging mechanism could handle many blocks at once.

The regulator, a TPS7633D, was chosen because it features an enable pin as well as a status line monitoring when the regulator begins to drop out of regulation. It also has very low current drain characteristics when disabled, which helps the battery lasts longer when the device is powered off. The enable pin is connected to the main microcontroller and disabled by default. However, a push button can momentarily pull the pin high, turning on the circuit. At this point, the microcontroller can take over and maintain the regulator's enable pin high, providing power to the circuit. Pressing the same button again triggers an interrupt on the microcontroller, signaling a power off request. The microcontroller can also turn off the regulator if the monitoring line signals a regulation drop, preventing the battery from draining beyond the point of recharge.

The blocks are powered by a flat rectangular lithium battery with a capacity of 450mA/h. The charging tray is powered by a modified PC power supply which can provide up to 22A at 5V. Currently, three trays are available with each 8 slots, allowing 24 blocks to charge simultaneously. This sets the power requirements 8A (24 * 250mA), leaving plenty of headroom for additional trays to be connected to the same supply. Blocks glow an orange colour when placed on their trays to indicate charging and turn a green colour when the battery is fully replenished.

4.5 Firmware

Following the modular structure of the hardware, separate firmware had to be developed for the brain and hindbrain microcontrollers. The brain's firmware implements a set of core operating system level functions, such as power management and communications. Blocks are expected to run according to a common main loop structure, in which the processing part varies depending on specific applications.

The IR communication protocol works using a polling mechanism. As far as individual blocks are concerned, communication is mostly one way. Blocks only receive data that have been explicitly requested and push data out whenever they determined such a request occurred, regardless of whether or not block is actually listening or not. This scheme is very robust to random noise and false triggering coming from ambient light conditions in the environment.

On every round of their internal clock, each block performs an asynchronous discovery of their neighbourhood during which their own IR interrupts are disabled. A short polling ping is sent, one side a time. The response can either be an echo ping of the exact same duration or a timeout. If a correct reply ping is received, then a neighbour is present. The block then goes in a receiving state and is ready to accept incoming bytes. On the pollee's side, data is sent using a software UART implementation. Each byte begins and ends with a synchronization bit so that each data bit is sampled approximately in the middle of its pulse's period for better reliability. Error detection schemes are to be implemented on top of this layer and can vary depending on the application. For instance, for applications transferring very short packets (2 bytes), one can simply establish as part of the protocol a systematic re-send of very packet. If the packet and its copy do not match, they can be safely discarded as noise. When bandwidth is not a bottleneck, this simple method has the advantage of taking very few cycles to evaluate over regular CRC checks.

The hindbrain's firmware was designed such that it is decoupled from the main processing module. This allowed the firmware to be developed independently. It provides an API for its functionality in the form simple commands. The packet structure consists of a command byte, followed by a number of bytes determined by the command. This portion of the firmware was implemented and

Command	Byte Value	Message Length
NOTE	0x1	2 bytes
VOLUME	0x2	1 byte
RGB	0x3	3 bytes
FADE	0x4	2 bytes

Table 4.1: SPI Packet Structure

thoroughly tested first so that each hindbrain microcontroller would only require to be programmed once.

The brightness of the RGB LED is controlled by three PWM channels. The pulses are generated in software using a custom algorithm built around the internal 8-bit timer. This eliminates the need for a dedicated PWM controller. The brightness values for each channel are double-buffered in order to allow The PWM generation loop is tightly optimized and was originally designed to support 12 channels (4 RGB LEDs, one for each side). The final design of the board, however, included only one LED to cut down on cost.

The sound synthesis algorithm iterates over a look-up table containing duty cycle values for a full sine wave period. On every iteration, a 16-bit accumulator is increment by a fixed amount. The low byte of the accumulator is discarded and the remaining byte used as an index in the look-up table. This technique allows for a better fine tuning resolution of the output signal frequency. For example, at 20Mhz clock speed and using a look-up table with 8 bit of resolution, the accumulator allows for a frequency resolution of 0.65 Hz per increment. The value contained in the table is then used to set the PWM module's duty cycle for that period. By varying the duty cycle, a low-pass filter can produce analog levels varying between ground and the positive supply. The values stored in the table represent a sine wave with a full swing between the two supplies. In order to generate different amplitudes, 16-bit integer arithmetic operations are used to scale down the reference duty cycle values while maintaining them centered around the half the positive supply.

Chapter 5

Applications

5.1 Reaction-diffusion

The first application explored for the Sound Mites was a simulation of a distributed algorithm which showed some emergent properties. The goal for this type of application is to give users the opportunity to discover the emergent patterns resulting from local interactions by slowly building up an assembly of blocks, starting from a clean slate. Common examples of such algorithms involve cellular automata, such as Conway's Game of Life or the Demon's Cycle. However, cellular automata are a poor match for an actual distributed platform because much of their theory of application relies on the existence of a master clock driving the simulation in a synchronized fashion.

After some investigation, I settled on an algorithm inspired by a reaction-diffusion simulation. Reaction-diffusion is "a process by which two or more chemicals diffuse over a surface at unequal rates and react with one another to produce stable patterns" [34]. The resulting patterns produce a variety of spots, gradients and stripes which often very closely resemble patterns found in nature such as on an animal's fur. Reaction-diffusion simulations have been used extensively in computer graphics to generate such textures algorithmically.

Assuming the existence of a cell with some concentration of chemicals a and b, the following equations to generate one-dimensional patterns was proposed by Alan Turing [35].

$$\triangle a_i = s(K - a_i * b_i) + D_a(a_{i+1} + a_{i-1} - 2a_i)$$

$$\triangle b_i = s(a_i * b_i - b_i - \beta_i) + D_b(b_{i+1} + b_{i-1} - 2b_i)$$

The first part of each equation represents the internal reaction process. K here is a constant value which can be tweaked for different results. The variance of concentration for each chemical is correlated between the two. β_i is a random value which represents small irregularities in cells, typically showing a variation of about +/- 5%. s can be used to control the effect of reaction in the overall process. The second part of the equation represents the diffusion process, by which chemicals leak to neighbouring cells. Here, D_a and D_b are diffusion constants. For this application, D_a was set to 0.3 and $D_b$to 0.14, which means that a diffuses more rapidly than b. This one-dimensional system can be easily extended to a two-dimensional grid by taking into account the four neighbours rather than the two neighbours in the diffusion half of the equation.

In our implementation, blocks poll their neighbours for their concentration of a and b chemicals, then they proceed to adjust their own concentrations according to the rules of reaction-diffusion described above. A typical visualization of the process involve mapping the concentration of each chemical to a different colour, and combining them in order to create textures. For this application, I wanted to extend the visualization to incorporate a sound component as well. Thus, rather than controlling the colour, the chemical concentration within a block was mapped to brightness and sound volume. Blocks can play one of four notes according to the number of neighbour it has. Each note is directly mapped to a colour, from the lowest to highest pitch following a progression along the colour spectrum (yellow, red, blue and green). Touching a block causes an influx of chemical into it, which remains constant so long as the block is being held.

The result is a system where predictable "chord" structures can be built according to the topological arrangement of the blocks. However, users also have to seed and control the creation of patterns and gradients within the system. They can do so by touching one of the blocks, thereby generating an instant charge of its a and b chemicals and causing light patterns and sounds to flow within the structure they assembled.

The notes played by the blocks were selected using Just Intonation, which is a tuning system where

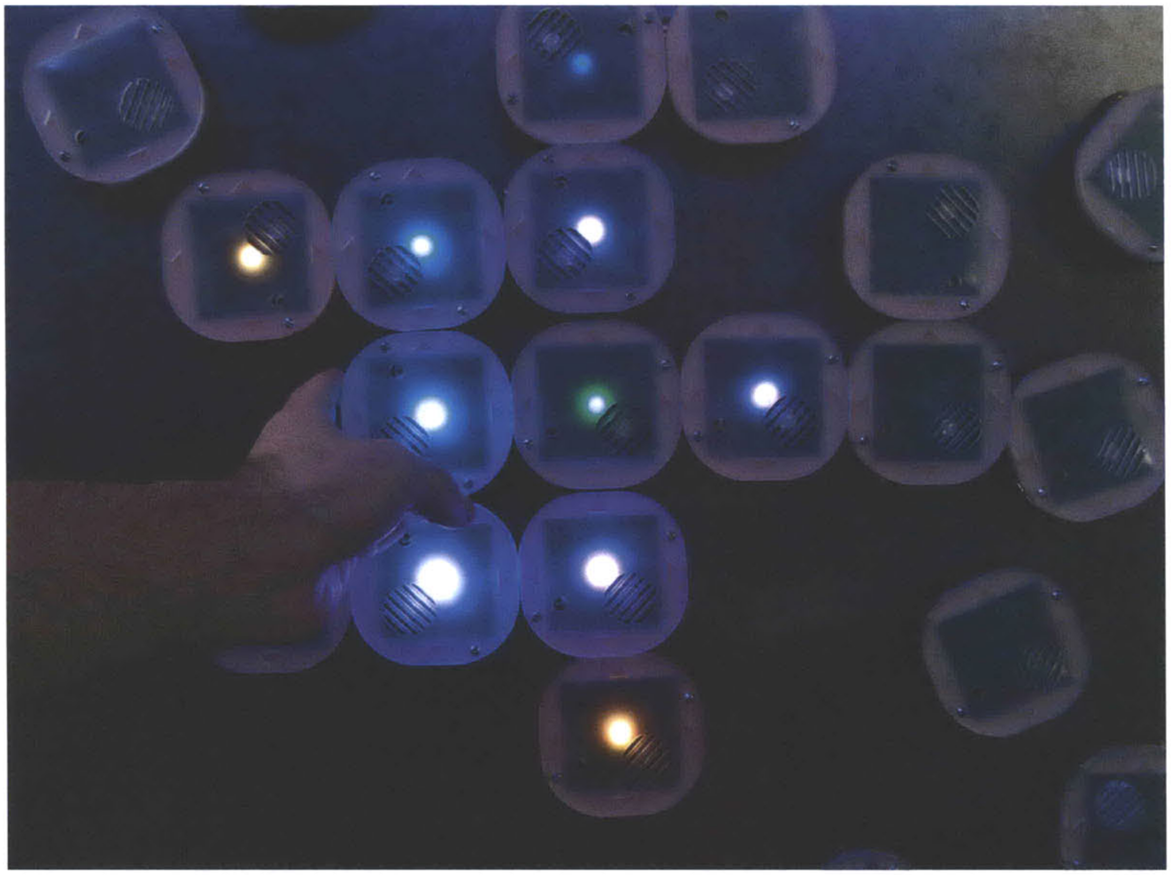


Figure 5-1: Touching a block causes it to charge and diffuse to other blocks nearby

each interval is represented by a ratio of whole numbers (with a strong preference towards small number ratios). The scale is built around a base pitch, which becomes the 1:1 basis. Other notes are then derived from it using ratios, for instance 9:8, 5:4, 4:3, 15:8 and so on. This contrasts with the usual system of modern Western tuning, known as equal temperament, in which octaves are divided in 12 equal intervals. While tempered scales have the benefit of allowing easy transposition between octaves, the human ear is more receptive to the whole number ratios of just scales, resulting in a sound which has a pure and harmonious quality to it [36].

5.2 Block Synthesizer

The second application acts as a counterpoint to the reaction-diffusion simulation. While the latter illustrates the notion of emergent patterns and gradients out of local interactions between blocks, the algorithmic rules at work remain relatively abstract. Some thought was given on how to use the blocks in a more systematic, construction kit application with a simple, clear set of mappings where cause and effect relationships would be more obvious.

A rather famous example of such an application is Block Jam [37], which came out of Sony's Interaction Laboratory a few years ago. Block Jam is a tangible interface comprised of a number of interconnectable blocks which control a dynamic polyrhythmic sequencer. A "cue ball" bounces around from one block to another within a cluster according to the simple rules determined by the functional state of each block. Blocks can display their state using an LED matrix display and toggle between states using a push-button as well as a touch sensitive dial wheel. Each cluster must contain a tethered block known as the "mother block" which is connected to a PC and handles message passing and timing. The bouncing cue ball generates MIDI sequence information which can then be used by a software synthesizer on the PC side to generate sound output.

The Block Synthesizer borrows elements from the Block Jam system while introducing some new ideas. Unlike Block Jam, the Sound Mites do not rely on a nearby PC to generate the sounds or control what the blocks should be displaying. I think this is an important design distinction – while Block Jam is an interface to a MIDI sequencer, the Block Synthesizer is an interface to a distributed system. Also, having the feedback directly on the interface is a desirable feature for a tangible interface. Furthermore, the Mites generate the sequencer behaviour from a single rule set while Block Jam uses specialized blocks (the blocks remain the same, but their behaviour is changed to fulfill a specialized task). However, while following a single set of rules can improve the system's intuitiveness by simplification, it is a trade-off with the complexity of the sequences that can be created.

In this scenario, the local interaction rules are quite simple. Messages between blocks are exchanged either *horizontally* (keep in mind there is no absolute orientation for the blocks) which means to the opposite side from which a message was received, literally "passing the message along" or *vertically*, to the blocks perpendicular to the side from which a message was received, creating a split in the message flow. Blocks can exchange one of three types of messages, which represent a set of actions that the block should follow upon reception.

- **Play**: When a *Play* message is received, immediately send a *Chord* message if vertical neighbours are present. Afterward, play a tone for a short time, then pass along the *Play* message horizontally. If no blocks are left on the opposite side to receive a *Play* message, send a *Rewind* message back.

- **Chord**: When a *Chord* message is received, immediately send a *Chord* message horizontally, then play a tone a short time. If there are no blocks on the opposite side, the *Chord* message ends. The desired effect of a *Chord* message is to trigger the vertical notes almost simultaneously in a rapid succession.

- **Rewind**: When receiving a *Rewind* message, simply pass it along horizontally immediately. If no block is left on the opposite side, act as if the block had just received a *Play* message. The Rewind message basically backtracks to one end of the structure, where the loop can start over.

Touching a block causes it to send a *Play* message to the block on either side in the horizontal direction, creating a new looping cycle. If a block has no neighbours at all, then all message sending attempts just stop, therefore separating the blocks causes the sequencer to reset to a clean state.

The synthesizer application was developed as an experiment after the blocks had been designed and built. As such, we must introduce the notion of a control blocks in order to enable a range of different tones and volumes. The control blocks are special blocks with a different firmware who's purpose is to modify the behaviour of regular blocks. Immediately useful control blocks are Tone and Volume. Placing a control block next to a regular block sends a control message which updates the tone or volume value for the block accordingly. In order to enable the user to select between different values, the control block will iterate over its available range until the two blocks are separated, cause the regular to use the last received value from then on. The Tone block has 7 different tones available, which consists of an extension of the Just Intonation scale used in the reaction-diffusion application. The Volume block provides 5 different volume steps. The range of values available were kept small for the sake of simplicity and ease of use, but could be easily extended.

The introduction of a control block is somewhat of a hack, considering that one of the design challenges put forward by this project is to use sets of identical blocks to create higher-level behaviours. In an ideal world, both applications would require completely different enclosure designs. The synthesizer blocks, in this case, would greatly benefit from a dial mechanism, for instance a twisting motion of the cover plate, a sensitive recess such as the one found in Block Jam or small volume wheel, eliminating the need for the control blocks altogether.

That said, the resulting effect remains that clusters of block can collaborate by relying messages to one another and by following a simple of set of rules in order to create a tangible synthesizer capable of looping and chord structures.

The main limitation of the Block Synthesizer lies in the timing and the speed at which messages can be passed along, particularly the *Chord* messages, in order to maintain a consistent tempo. Perhaps a worthy addition to the system would be to extend the packet structure for each message with timing information as well as a counter, indicating for instance the number of "hops" crossed by a Rewind message. This information could be used to dynamically alter the tempo of the sequencer, reflecting the travel time of messages throughout the system – a small cluster would play faster than a large cluster. It is clear, however, that the system will not scale indefinitely. This is perhaps a more general drawback to this type of structured application. In contrast, in the emergent behaviour-type systems, the behaviour of each block is independent of the behaviour of the system as a whole, allowing pattern formations at virtually any scale.

5.3 Other application scenarios

While the two scenarios described were actually implemented using the Sound Mites, there is a range of potential applications that would be interesting for a platform with similar specifications.

For instance, one larger theme is that of educational games. The magnetic blocks have a very playful quality to them, and their ability to perform computations could enable some interesting and challenging puzzles. For instance, a puzzle inspired by the game of domino but using rules derived from color theory for legal combinations could be interesting. One could also imagine some sort of memory game. Blocks could be randomly assigned a tone when powered up. Touching a block would cause it to play its tone. Placed next to another block, they would both light up an identical colour if their respective tones are in harmony or part of a set of typical music chords, or red if they are dissonant. This system could be an entertaining way to learn about chord formations and music theory through direct feedback and manipulation, training one's ear to recognize note formations while playing around with the blocks.

Another area of application is the visualization, prototyping and also teaching of a broader category of algorithms falling under "nature inspired computation". Reaction-diffusion is one example, but there are other instances where being able to manipulate a system might help enhance the user's understanding, such as neural networks. These algorithms deal with very abstract concepts, and providing toy implementations on tangible blocks could help improve understanding of the basic principles at work for newcomers.

Finally, I think an area with a lot of potential for distributed and autonomous blocks is the creation of rich, responsive architecture and environments. Working at a completely different scale than the projects implemented for this thesis, these applications would shift the focus from the manipulation aspect of the blocks to their decentralized nature. For instance, one can envision a floor, a wall or even a facade built out of autonomous processing building blocks. There are examples of such structures already, but they are mostly built in a centralized fashion, where the blocks become merely an alternative display surface. However, the challenge of managing the control and processing for these surfaces increases as the systems get bigger and more complex. A decentralized architecture solves some of these problems by embedding the control in the blocks themselves. For instance, if a floor surface needs to be extended, one can simply add new tiles with the same behaviour, and the system will grow accordingly. Similarly, if a tile fails, one can simply go in an replace it with a new, identical one.

Chapter 6

Evaluation

A project dealing with emergence is challenging to evaluate because the concept can be hard to quantify. Similarly, the playful, expressive and engaging qualities of a particular design can all be fairly subjective. As such, I will focus my evaluation on two main aspects which I believe are sufficiently grounded and best suited to the scope of this project.

6.1 Technical evaluation

For the first half of this evaluation, I will examine whether or not the project was successful on a technical level. I will reflect critically on my implementation choices and the challenges faced, in light of the overall goals which were established during the design phase.

One of the key elements from a technical standpoint was to make sure that the system's performance enabled good real-time responsiveness. This was particularly important from a user experience standpoint. Since we expect users to quickly manipulate and reorganize the blocks' layout, the system should be accordingly responsive. Today's low-cost microcontrollers offer a lot of performance, which allows them to perform a variety of tasks which used to require dedicated external hardware. They are not, however, ideal for multimedia applications and generally lack useful features such as floating point hardware.

Thus, a lot of effort went into optimizing the output module so that it could handle sound synthesis and the multi-channel PWM light control with acceptable performance. Originally, the sound synthesis was going to have support for up to 3-voice polyphony. The idea was to allow effects such as mapping variables within an application to 3 different notes and onto each RGB channels. Polyphony would also have enabled each block to produce chords individually, opening up interesting possibilities. Unfortunately, running three independent sound synthesis channels is very costly for the AVR. While the look-up table and 16-bit accumulator technique is quite efficient to generate sine waves of various frequencies, the amplitude scaling operations are computationally intensive. Despite this, good performance was actually achieved with 3-voices on top of 3 RGB channels during isolated tests with the hindbrain's firmware, with no discernible flicker on the LED. Once connected to the main microcontroller, however, the SPI unit began missing bytes, rendering the control of the hindbrain extremely unreliable. This was unacceptable, since the hindbrain had to be as solid as possible to minimize the complexity of the debugging the main module, therefore only monophonic sound output is supported in the final version of the blocks.

Another potentially troublesome area was the infrared sensors, having made the decision to implement my own sensing and transmission circuit rather than going for a tried-and-true integrated solution in order to remain within budgetary constraints. The main factors at play were range, noise-immunity and communication speed.

The desired range for the sensors was a few inches, between 2" and 4", and resistor values for the circuit were picked accordingly using a breadboard version. Values around 500 ohms were used for both the amplifier and the IR LED. In the printed circuit board design, the range ended up being much shorter than what was exhibited by the prototype circuit. Resistor for the amplifier had to be doubled and the LED's resistor taken down to a tiny 15 ohms in order to get decent communication at a range of about 2 1/2". Increasing the resistor further on the amplifier meant increasing amplifier's floor value too high, and decreasing on the LED just made power consumption worse. In the end, the obtained range was sufficient for our application, although the blocks require a close proximity to communicate properly given the size of their enclosure.

On the other hand, reliability and speed are adequate. Like with any infrared-based transmissions, ambient lighting conditions will affect the receiver's sensitivity. The system was tested in a number of environments, including office lighting, indoors with proximity to day light and a dimly lit room, and performed acceptably in all cases. Proximity to halogen lamps induced some consistent trigger of the IR sensor, however, false communications were avoided in software using the ping handshakes and precise timings. The IR transceivers also allow the blocks to exchange data at a good rate. Each byte takes 120 microseconds to transfer, resulting in data rates of 83 kbps. The timings used are very conservative, since reliability was more important than speed for this type application. Considering a simple application such as the reaction-diffusion simulation exchanges 4 bytes per side per internal clock cycle, this leaves most of the CPU time on the hindbrain available for local processing or responding to neighbourhood blocks in a timely fashion.

The power circuit performed better than expected. Given that the blocks are prototypical in nature and thus are not required to match the battery life performance of typical consumer electronics, the initial autonomy required was around 3 hours or, in the context of the Media Lab, an afternoon of demos! While circuit design choices were made with low power consumption in mind, the combination of inexperience and some compromises such as the user of power-hungry microspeakers had to made in order to keep the circuit simple could have easily jeopardized this aspect of the project. Fortunately, the final incarnation of the blocks sported a battery life of approximately 12 hours, based on typical use in a public exhibition setting.

The most sorely missed feature in the final design was a scheme for mass re-programmability of the blocks. Currently, firmware updates to the blocks must occur on an individual basis. While the process of updating the firmware is relatively painless, it can quickly become a chore when 50 or more blocks are involved. The lack of mass re-programmability also increases development times for the platform significantly, even when testing with a handful of blocks which affects their medium-term reusability. However, for this project, I deemed more of a priority to focus on developing application, given that viral programmability had already been successfully implemented in a number of projects and thus, while quite useful, was not necessarily a novel feature to develop.

Finally, one of important design goals of this project was to create a platform for embodied emergent behaviour where individual nodes would be as simple as possible. In some respects, this simplicity was achieved. Using a handful of components and some commodity microcontrollers, one can now successfully put together such a platform. However, the circuit could probably use even further simplification. The cost of an individual node is still too high to produce in them larger quantities (~20$), not counting assembly labour. In hindsight, one mistake was probably to include the charging circuit on every block. Off-loading charging to an external devices, such as the charging tray, could cut the circuit's complexity by at least 30%. Furthermore, a simpler scheme to detect low-voltage on the battery could be developed, eliminating the need for a dedicated regulator and further slimming down the bill of materials.

6.2 Qualitative evaluation

The second part of my evaluation was a field test of the system in a public presentation setting. The evaluation took part in the context of the 2007 Boston Cyberarts Festival, which takes place biennially at a number of galleries and museum, theatres and public spaces in the Boston area. The Festival presents a large gathering of artists working with new technologies in a variety of media, including visual arts, dance, music, electronics, web art and public art. Sound Mites was on exhibit at the Collision 11 show, one of the events of the festival, between April 20th and May 3rd at the MIT Stata Center. Collision is a collective of local artists and inventors exploring new technologies. While the collective presents Collision as an experimental art show, the selection of works is rather inclusive and often includes works which do not make an artistic statement, such as this project, but that are interactive, crafty, and relate to technology in a novel and exciting way . The Collision show was also held in conjunction with the first Cambridge Science Festival, presented by the MIT Museum. Because of this, a large portion of the audience was young children, usually accompanied with their parents, especially during the weekends.

The decision to have a public exhibit as a validation mechanism for the project rather than a more formal user study was informed by a number of factors. Perhaps the most important was to maximize exposure and broaden the user base which gets the opportunity to interact with the system. While user studies of this scope will usually see a dozen test subjects, a public exhibit is good way to reach hundreds very easily. The exhibit also provided a varied user base, combining adults, students

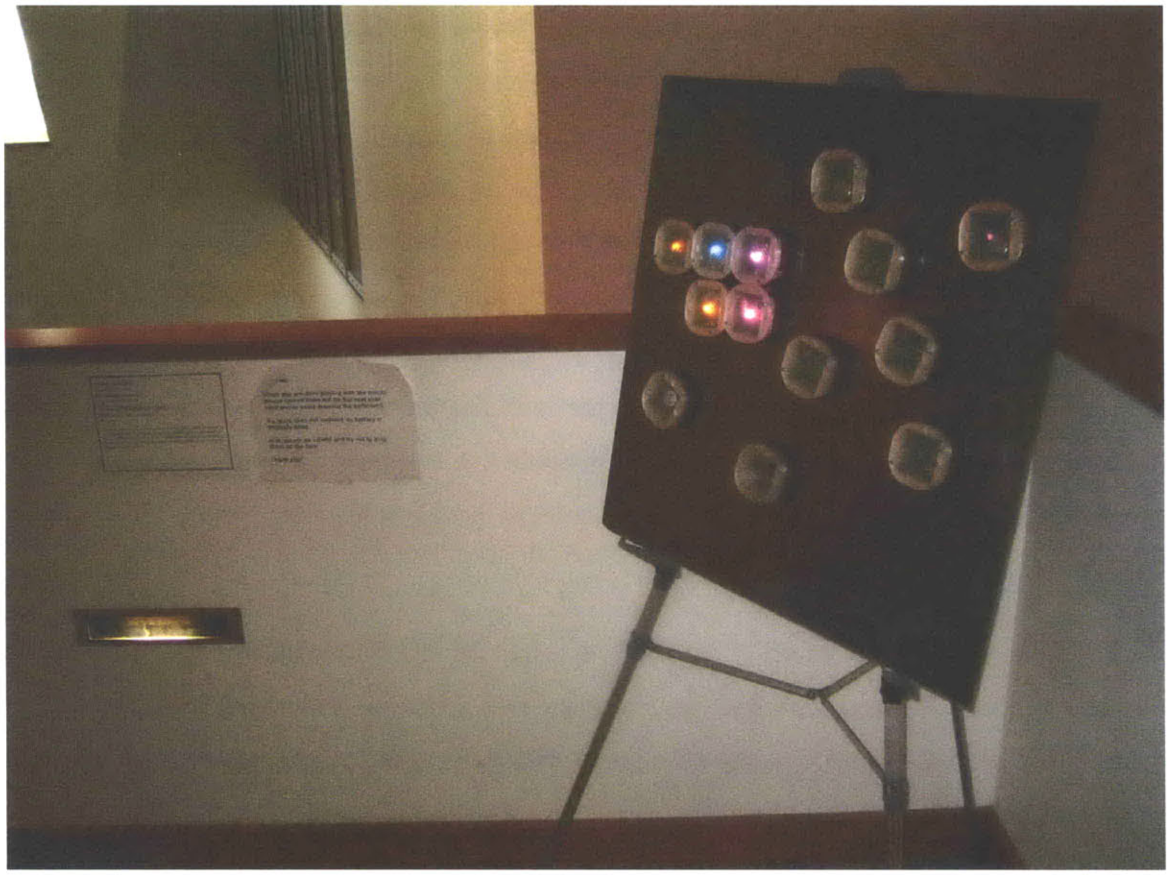


Figure 6-1: Collision exhibit setup

and young children, providing an overall representation which is closer to the intended audience of the system. Finally, a public exhibit is an excellent way to stress the robustness of the system. This is true both from an application perspective, where the software must run without crashes and the hardware without failures, but also quite literally, particularly when the blocks are repeatedly dropped onto a hard cement floor by enthusiastic children.

The evaluation methodology consisted of a number of short, informal interviews with users taken at random times for the duration of the festival. While the bulk of these short interviews took place during the opening night, periodic visits to the exhibit site during the following ten days allowed for a number of spontaneous discussions with a wide range of users. Interaction with the blocks was also videotaped on several occasions to provide material for reflection on the user experience.

The installation setup consisted of a 3" x 3" piece of steel mounted on an aluminum easel. The steel was mounted such that it was at an acceptable height for young children as well as adults, with a slight backwards inclination to stabilize the structure. At the time of the Collision show, 25 blocks had been assembled and were in working condition. Typically, the blocks would be scattered randomly across the steel surface, with no immediate neighbours. In that configuration, blocks emitted a quick light flash in sync with their internal clock cycle. This "screensaver" mode served the purpose of capturing the attention of passersby by providing them with intriguing patterns of asynchronous blinking lights. The usual interaction session usually lasted between one to twenty minutes and consisted of a user walking up to the easel and beginning to slowly assemble blocks together in order to discover the nature of their behaviour. A note placed by the easel prompted the users to break apart their assemblies after use in order to provide subsequent viewers with a similar experience.

One of the most compelling parts of the interaction was the fact that the blocks were mounted on the steel surface using magnets. This seemed to provide a familiar ground for people to interact with, reminiscent of magnetic poetry on a refrigerator door. The magnets provided an easy and quick way to pick up and place the objects. They also allowed for a vertical layout, which in some ways functioned better than an horizontal one for this type of interactions. Because the blocks are directly laid out in front of the user, it was easier to reach a large number of blocks on the plate without having to lean forward or to interact side-by-side with somebody.

Unfortunately, the speakers were a little bit quiet for this public exhibit, especially when surrounded by louder projects or a larger crowd. Thus, the overall experience functioned better when users could spend some quiet time with the piece. Even though they could not always be heard, one could sense that the blocks were emitting sounds through the vibrations caused by the speaker, causing some users to listen more closely. That behaviour was interesting, and provided an unexpected, more intimate way to interact with the work.

Most users would discover the system's behaviour in a bottom up fashion, starting with two blocks put together. Understanding the behaviour became a game of speculation and experimentation, where the users were encouraged to try different combinations. While ultimately the pay-off ceiling for the behaviour was rather low (beyond immediate reaction to the number of neighbours, blocks would only create diffusion patterns), there was a definite sense for users of progression in terms of understanding the system from its most simple rules to the more complex ones. Many users also had a tendency to approach the assembly of structures as a puzzle of sorts, where the goal became to create unusual or interesting layouts. Placing four blocks together caused them to become of an identical colour, since they all share the same number of neighbour. This stimulated experimentation with larger systems, for instance trying to discover structures where every block has exactly three neighbours.

Overall, I think that the number of blocks used in the experiment (25) was a bare minimum to convey a sense of emergent patterns (the block population was later doubled for sponsor presentations, and the results scaled accordingly). Also, because of the abstract nature of the patterns, I think the work had a stronger effect on adults than children, the former being able to conceptualize more easily how the system would scale up as they were discovering how to use it, filling in the blanks introduced by having only a restricted number of nodes.

Chapter 7

Future Directions

This project, like the others it builds upon, has just begun to scratch the surface of the creative possibilities of distributed systems. Therefore, in addition to the potential applications discussed in Chapter 5 and some of the shortcomings mentioned in the technical evaluation, there is a lot of room for future work in this area.

The input modality for the Sound Mites was a simple on/off capacitive switch. The interactive experience would be greatly altered by trying out different, perhaps more complex, sensing mechanisms. For instance, replacing the capacitive sensor by a push-button behind the top cover or a range sensor would result in a very different way of handling the blocks and triggering their behaviour. The range sensor is particularly interesting, introducing a new analog dimension (distance) and possibility to let users "play" the structure they assembled by simply hovering their hand above it.

Also, while this project extended the usual range of output modalities for a tangible distributed system by including sound as a feedback element, there remain many exciting possibilities to be explored. For instance, using actuators such as motors or magnets could result in some very dynamic visualizations of decentralized behaviour.

I would also argue that there is a space of research area for even simpler systems from a technological and computational standpoint than the ones discussed in this thesis. There is room to try and develop elegant, simple systems which would be even more closely inspired by nature and its manifestations. In these scenarios, the behaviour of the system would be tightly integrated with the hardware itself, relying on negative feedback mechanisms to generate patterns. Such systems would have their actions and reactions tied to the sensing and actuating mechanisms. For instance, can we develop a simple device which both emits and senses light, and design it in such a way that it can display some kind of behaviour when combined with several other similar devices? Another example related to sound would be a system where the nodes can play sounds but can also recognize sounds. A system like this would introduce interesting dynamics between the nodes themselves and how they could react to one another without communication taking place in the traditional sense, but also between the nodes and participants, where the latter could interfere with the behaviour of the system by mimicking some tones that the nodes could "understand".

Finally, one of my main personal motivations as a designer for undertaking this work was to experiment with the notion of distributed systems and how I could leverage their properties to create unique interactive audio/visual patterns experiences. In recent years, there has been a lot of progress on quick prototyping platforms for artists, designers and architects, such as the Arduino development boards [38], which empowers them to build their own electronic devices, complete with sensors and actuators, without having to dive too deeply into the complexity of electrical engineering and circuit design. While these platforms usually provide easy integration of a networking module of some sort, the main concern is with communication back and forth with a single PC, for example. I believe that there is a need for similar prototyping platform with a focus on distributed systems. The decentralized approach to system architecture can offer rich interaction models and output behaviours but the cost of entry of developing such applications is currently very high, with almost no options for prototyping systems. While there exists a range of platforms targeted towards sense networks which could potentially fulfill this role, their target audience and intended purpose are very different from the ones required for projects such as the ones contemplated in this thesis, which makes them less than ideal candidates for the task. In short, I hope to see in the future the advent of a platform which will enable me to quickly prototype a project idea such as the Sound Mites and enable me to focus on the exploring different behaviours and interaction scenarios without having to build boards from the ground up.

Bibliography

[1] S. Johnson. *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. Scribner, 2001.

[2] J.S. Mill. *A System of Logic*. Lincoln-Rembrandt Pub., 1986.

[3] M. Resnick. StarLogo: an environment for decentralized modeling and decentralized thinking. *Conference on Human Factors in Computing Systems*, pages 11–12, 1996.

[4] E. Mendelowitz. The Emergence Engine: A Behavior Based Agent Development Environment for Artists. *Proc. Twelfth Conf. on Innovative Applications of Artificial Intelligence (IAAI)*, pages 973–978, 2000.

[5] H. Abelson, R. Weiss, D. Allen, D. Coore, C. Hanson, G. Homsy, T.F. Knight Jr, R. Nagpal, E. Rauch, and G.J. Sussman. Amorphous computing. *Communications of the ACM*, 43(5):74–82, 2000.

[6] W.J. Butera. *Programming a Paintable Computer*. PhD thesis, Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences, 2002.

[7] D.E. Hirsh. *Piecing Together the Magic Mirror: a Software Framework to Support Distributed, Interactive Applications*. PhD thesis, Massachusetts Institute of Technology, 2006.

[8] J.H. Lifton. *Pushpin Computing: a Platform for Distributed Sensor Networks*. PhD thesis, Massachusetts Institute of Technology, 2002.

[9] B. Warneke, M. Last, B. Liebowitz, and KSJ Pister. Smart Dust: communicating with a cubic-millimeter computer. *Computer*, 34(1):44–51, 2001.

[10] F. Fröbel and J. Jarvis. *Friedrich Froebel's Pedagogics of the Kindergarten: Or, His Ideas Concerning the Play and Playthings of the Child.* D. Appleton, 1895.

[11] M. Resnick, M. Eisenberg, R. Berg, and F. Martin. Learning with Digital Manipulatives: A New Generation of Froebel Gifts for Exploring" Advanced" Mathematical and Scientific Concepts. *Proposal to the National Science Foundation, May*, 1999.

[12] E. Schweikardt and M.D. Gross. A Brief Survey of Distributed Computational Toys, 2007.

[13] N. Elumeze and M. Eisenberg. SmartTiles: Designing Interactive Room-Sized Artifacts for Educational Computing. *Children, Youth and Environments*, 15(1), 2005.

[14] J.H. Conway. The Game of Life. *Scientific American*, 223:120–123, 1970.

[15] M. Eisenberg, L. Buechley, and N. Elumeze. Computation and Construction Kits: Toward the Next Generation of Tangible Building Media for Children. *Proceedings of Cognition and Exploratory Learning in the Digital Age (CELDA), Lisbon, Portugal*, 2004.

[16] K.H. Kramer. *Moveable Objects, Mobile Code.* PhD thesis, Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences, 1998.

[17] O. Zuckerman and M. Resnick. System Blocks: A Physical Interface for System Dynamics Learning. *Proceedings of the 21st International System Dynamics Conference*, 2003.

[18] V. LeClerc, A. Parkes, and H. Ishii. Senspectra: a computationally augmented physical modeling toolkit for sensing and visualization of structural strain. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 801–804, 2007.

[19] M. Whitelaw. *Metacreation: Art and Artificial Life.* MIT Press, 2004.

[20] B. Knep. Healing Series. `http://www.blep.com/healing/index.htm`, 2003-2004.

[21] M. Whitelaw. Us is Them. `http://creative.canberra.edu.au/mitchell/usisthem/`, 2006.

[22] Brown A. and Fagg A. Is it alive? Sensor Networks and Art. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 22, New York, NY, USA, 2006. ACM Press.

[23] B. Vorn. Evil / Live 3. http://billvorn.concordia.ca/robography/Evil3.html, 2004.

[24] J. Drucker. Simon Penny. *Art Journal*, 56(3), 1997.

[25] Aether Architecture. Ping Genius Loci. http://www.aether.hu/pgl/, 2006.

[26] P. Reiners. Autonomous Monk. http://www.automatous-monk.com/.

[27] Wolfram Research Labs. Wolfram Tones. http://tones.wolfram.com/, 2005.

[28] A. Dorin. The Virtual Ecosystem as Generative Electronic Art. *2nd European Workshop on Evolutionary Music and Art, Applications of Evolutionary Computing: EvoWorkshops*, pages 467–476, 2004.

[29] Ando D. and Iba H. Real-time Musical Interaction between Musician and Multi-agent System. In *Proceedings of the 8th Generative Arts Conference*, 2005.

[30] B. Fry and C. Reas. Processing 1.0. http://processing.org/, 2004.

[31] Berkeley MOTES. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.

[32] Sun SPOTS. http://www.sunspotworld.com.

[33] The New Ecology of Things. http://people.artcenter.edu/~vanallen/ecology/, 2006.

[34] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 289–298, 1991.

[35] A.M. Turing. The chemical basis of morphogenesis. *Bulletin of Mathematical Biology*, 52(1):153–197, 1990.

[36] E.H. Pierce. A Colossal Experiment in Just Intonation. *The Musical Quarterly, Vol. 10, No. 3*, pages 326–332, 1924.

[37] H. Newton-Dunn, H. Nakano, and J. Gibson. Block Jam: A Tangible Interface for Interactive Music. *Journal of New Music Research*, 32(4):383–393, 2003.

[38] Arduino. url:http://www.arduino.cc/.