MUSICAL SWARM ROBOT SIMULATION STRATEGIES

A Thesis
Presented to
The Academic Faculty

by

Aaron Thomas Albin

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Music Technology in the
School of Music

Georgia Institute of Technology
December 2011

MUSICAL SWARM ROBOT SIMULATION STRATEGIES

[To my family and friends]

ACKNOWLEDGEMENTS

I thank Dr. Gil Weinberg, my advisor, for his guidance and support. I am forever grateful for the opportunity to work with him in this field. His wisdom and insights have helped me immensely.

I also thank Dr. Jason Freeman for his guidance and support. I cherish the opportunity being his student and aspire to one day teach as well as he does.

Dr. Magnus Egerstedt introduced me to the world of multi-agent control theory which has been incredibly helpful for my research. I am thankful to him for the opportunity to collaborate and for his guidance and support as well.

I wish to thank the students of the Georgia Tech Center for Music Technology, especially those who participated in this research study.

I thank Jonathan Murphy, my roommate and friend, for his support and help with proofreading. I am grateful to have been able to share my ideas with him.

Additionally I thank my friends, Dr. Kaveh Ardalan, Farsheed Hamidi-Toosi, and Joyce Chang for helping with proofreading of the thesis.

Finally, I thank my mother and father for their love, guidance, and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Adjacency Matrix

An $n \times n$ diagonal matrix containing the degree of each agent along the diagonal

Agreement Protocol

A method of multi-agent coordination where a collection of agents are meant to agree on a joint state value

Boid

A single agent in a swarm that uses the flocking behaviors of separation, alignment, and cohesion, as described by Craig Reynolds

Connected

A quality of a graph where for every pair of vertices in the graph there is a path that has them as its end vertices

Degree

The degree of a vertex is the number of vertices that are adjacent to that vertex; in other words, the degree of agent A is the number of connections made from agent A to the other agents in the network

Degree Matrix

An $n \times n$ diagonal matrix containing the degree of each agent along the diagonal

Digraph

	Directed graph; a graph in which the connections between vertices have direction associated with them
Directed Cycle	
	A closed loop of vertices within a digraph
Edge Laplacian	
	The difference between the diagonal degree matrix consisting of only in-degrees (the number of connections going into a vertex) and the Adjacency Matrix
Epistemic action	
	Action that a person performs to change his or her own state of thinking
Flocking	
	A type of movement behavior commonly seen in natural swarms like birds or fish. Flocking involves three parameters of separation, alignment, and cohesion
Graph	
	An abstraction for describing the interconnected states of vertices (agents)
Graph Laplacian	
	The difference between the idegree matrix of a network of agents and its adjacency matrix
Pragmatic action	
	Action that brings an individual closer to his or her goals
Rooted Out-Branching	

A quality of a graph which contains no directed cycles within it and it has a vertex *a* such that there is a path from it to every other vertex

Stigmergy

A concept describing how swarm agents make sense of their environment by leaving traces in it, similar how ant colonies leave pheromones for others ants

Strongly connected

A type of connectedness in which for every pair of vertices there is a directed path between them

Swarm

A group of similar agents that are influenced by each other in some way, typically by motion

Weakly connected

A type of connectedness in which the edges of the digraph were replaced with two way connections, that is to say, turned into a disoriented digraph

SUMMARY

Swarm robotics for music is a relatively new way to explore algorithmic composition as well as new modes of human robot interaction. This work outlines a strategy for making music with a robotic swarm constrained by acoustic sound, rhythmic music using sequencers, motion causing changes in the music, and finally human and swarm interaction. Two novel simulation programs are created in this thesis: the first is a multi-agent simulation designed to explore suitable parameters for motion to music mappings as well as parameters for real time interaction. The second is a boid-based robotic swarm simulation that adheres to the constraints established, using derived parameters from the multi-agent simulation: orientation, number of neighbors, and speed. In addition, five interaction modes are created that vary along an axis of direct and indirect forms of human control over the swarm motion. The mappings and interaction modes of the swarm robot simulation are evaluated in a user study involving music technology students. The purpose of the study is to determine the legibility of the motion to musical mappings and evaluate user preferences for the mappings and modes of interaction in problem solving and in open-ended contexts. The findings suggest that typical users of a swarm robot system do not necessarily prefer more inherently legible mappings in open-ended contexts. Users prefer direct and intermediate modes of interaction in problem solving scenarios, but favor intermediate modes of interaction in open-ended ones. The results from this study will be used in the design and development of a new swarm robotic system for music that can be used in both contexts.

CHAPTER 1 INTRODUCTION

Motivation

Swarm robotics is a field in which multiple robots are used to accomplish tasks that a single robot might not be able to do. Examples of some applications include search and rescue, chemical concentration mapping, and surveillance [1,2,3]. Even though swarm robot systems have not yet seen widespread use for such tasks, studies in controlled scenarios could potentially lead to widespread use of such systems to accomplish these different tasks more effectively and efficiently. One example under development is the Propsero farming hexapod robot, whose mission is to do crop planting, coordinating with other robots to achieve an efficient use of farming space [4]. While all of the above are interesting research areas, this thesis aims to explore the use of swarm robots for a different goal: making music.

Musical robotics is a field that seeks answers for many different types of research problems. Sometimes the goal is to design a machine that can play similar to a human[5]. Other times the goal is to play with virtuosity and perhaps even outperform what humans can accomplish [6]. Recently, several researchers have focused more upon human interaction with robots [7,8,9]. The motivation for merging the fields of musical and swarm robots is to create novel forms of music that cannot be created otherwise as well as novel forms of human-robot interactions that have different aesthetic outcomes.. A swarm robot platform could allow for new modes of human interaction with robots that will also lend to different ways of thinking about algorithmic composition. As a simple example, consider ten snare drums attached to mobile robots that obey an arbitrary rule: robots may play their drum at a fixed interval only if they are within close proximity to another robot; otherwise they do not play at all. If the robots were initially isolated and then converged to a single location, resulting music would initially be silent and then evolve into a repeated rhythmic pattern. If the robots started out within proximity to one another and then separated so that they have no neighbors, the music would start off with a repeated rhythm and then get quieter depending on how each robot moves out of proximity to other robots. What this example illustrates is that swarm robotics has the potential to allow for exploration of algorithmic compositions through the coordination motion of agents, using the relationships between mobile agents as means to traverse a variety of musical spaces.

Design Constraints

In order to inform the design of a swarm system, it is helpful to make an assumption about typical users of a swarm robot system for music. I expect that such users will have some degree of experience with interactive music systems, music composition, and some degree of computer literacy. Users might interact with the swarm robots in an installation setting or perhaps they might be composers who would want to make algorithmic music with the robots. Given that this is the intended audience, I make additional decisions regarding a specific design philosophy and musical aesthetic.

Constraint 1: Acoustic Sound with the Robots:

Acoustic sound is a priority for the swarm robots. Many musical robots are designed to emulate human playing of real instruments. Weinberg and Driscol are proponents of acoustic sound over synthetic, claiming that the latter cannot capture the richness of sound present in real instruments [8]. In addition, since anticipatory gestures have been proven to aid in synchronization tasks, acoustic sound could also help users localize robots that are playing [10,11]. For example, if the swarm robots were to have mallets or strikers as opposed to the sound only coming from speakers, users will more likely be able to discriminate between a group of robots clustered together. Implicit in this design constraint is that the robots themselves should make the music. The advantage to imposing this constraint is that spatialization is inherent to the design. The users can listen to the music from a variety of different angles, even from within the swarm itself if they were to be surrounded by the agents.

Constraint 2: Rhythm with Sequencers.

The swarm robots must be capable of metrical timing in the form of sequencers. Sequencers are one of the most commonly used forms of maintaining rhythmic timing. Most users that I expect to interact with the swarm robots will be familiar with this form of music creation since they are prevalent in many different types of professional recording tools such as Reason, Logic, Fruity loops, and Ableton Live. The use of sequencers quantizes the rhythmic space to a set of 2^n possibilities where n is the number of steps in the sequencer. The sequencer allows for the use of Boolean operators and other types of musical functions that can operate on arrays. This also provides an easy method by which robots can share sequencers with each other, either by sending the array contents, or the parameters of musical functions used to populate the array. Certain types of rhythmic patterns might be more difficult to achieve in a sequencer approach; for example, a pattern in which the time interval between successive hits decreases logarithmically would be almost impossible to represent unless n is extremely large. However, a size with n equal to 16 is good enough to provide syncopation and complex rhythms. Additionally, the robots should have the ability to synchronize with each other and should be tied to a common clock; in all of the recording tools described, it is assumed that all sequencer tracks are of the same length and start at the same index of the sequence. The robots will each contain a sequencer of the same length. The interval between each element of the sequencer will be the same among all robots as well as the play head index of the particular sequence. Later, if I wish to have the robots keep their own separate meters or even separate timing intervals, it will be trivial to accomplish once the ability to synchronize is obtained.

Constraint 3: Motion Causes Music

Swarm motion should be the primary means by which the music is changed. Recalling the ten snare drum analogy, it is certainly possible to compose a musical piece with ten snares that emulates the type of music that robotic swarm system could make. However, unless the snare drums are moving, one would lose a sense of exactly how that music came to existence. New methods of composing music for ten snares may not arise simply by having them remain stationary. Observing the changes in motion affects one's perception of the music and can also give new creative ways of music composition. Swarming music is a field of algorithmic composition that tries to accomplish this goal in a real-time and decentralized manner specifically through the motion of swarm agents [12]. Algorithmic computer-based swarm music typically deals with a fixed number of simulated agents whose properties are used to create sound. One major source of inspiration for swarming music comes from Craig Reynolds' boid model; this type simulation outlines simple rules for swarm agents to follow that emulate natural motions of flocks of birds or schools of fish. As Reynolds notes, "perhaps most puzzling is the strong impression of intentional, centralized control. Yet all evidence indicates that flock motion must be merely the aggregate result of the actions of individual animals, each acting solely on the basis of its own local perception of the world" [13]. Swarming music makes use of this motion and uses these agents to move through a virtual space. A swarm robotics platform could also produce algorithmic music of this nature. The motion of agents could be "sonified", providing a musical aesthetic that a single robot could not convey effectively, or even a human marching band could not convey as effectively in real-time. There are problems, however, in directly using previously designed boid simulations for swarming music. The strategies taken for how to map motion to music are not always applicable in a robotics approach. For example, many swarming music simulations allow agents to pass through one another, move in three-dimensional space, and move at speeds that are very difficult to achieve in the real world. Therefore it is necessary to design a constrained boid simulation that reflects real world parameters, taking into account robots moving in a two dimensional plane on the ground and moving at realistic speeds. Additionally of concern is how previous swarm music simulations use absolute positioning. The use of absolute positioning to direct the motion of agents is not altogether unwarranted; some multi-robot systems utilize top down camera system to track the positions of individual agents [14]. The dilemma with using absolute positioning parameters in swarming music arises when they are used to directly affect musical parameters; this is problematic because it becomes unclear whether the agent is playing music or if instead the space through which they move is what affects them. As an example, consider two robots positioned in the center of a very large room with the following movement behavior: the first robot is stationary, while the second orbits around the first at a fixed radius. Now consider the robots positioned at any arbitrary location in the room other than the center, but with the same movement behavior. Mapping the absolute position of the robots in these cases would produce different musical output for the exact same time of motion behavior. To avoid this problem, a better approach would take into account the relative distance and proximity of robots. One way to achieve this is to use multi-agent control theory, which was inspired in part by the work of the boid algorithm. Multi-agent control applied to robots creates coordinated movement using only what a robotic agent can discern about its own environment [15]. Using multi-agent control theory as more formal approach to understanding swarm motion, I can design a better robot simulation that is constrained to use parameters from the perspective of individual agents, taking relative distance and proximity into account. Motion, therefore will be a constraint of the swarm robot system, provided that the parameters of motion are only valid from a multi-agent control theory perspective. This will hopefully result in rhythmic music that is well-coordinated with the swarm motion.

Constraint 4: Interaction with Humans

Human interaction is an important feature of many swarming music systems and installations [16,17,18], similarly, humans should interact with a swarm robot system. A human interacting with a swarm could make a guided type of algorithmic composition in which they traverse a rhythmic musical space. In this way, the resulting music may be more entertaining and engaging than if the robots were simply left to move by themselves. The problem is to determine the nature of the control that humans should have over the swarm. Multi-agent control theory can be used to determine free parameters that can help in creating different modes of interaction that vary according to degrees of control. I define the axis of interaction based on how a user interferes with robot motion. A very direct form of interaction might involve forcing them to move to specific locations or in some relation to the user. An indirect approach to interaction might involve issuing commands to robots that make them move without any relationship to the user; in this way the user still affects the motion, but the swarm robots are not aware the human at all. An intermediate approach might involve initially direct actions such as interfering with robots' motion by picking up and moving them to different locations or configurations, and then allowing them to move autonomously afterward when placed down again.

"Inevitability," as defined by Machover is a hallmarks of good instrument design and this principle should also apply to the use of swarms for musical ends [19]. The musical mappings and modes of interaction of a swarm robot system should take this into account. Coming to the robots for the first time, human operators should not need to have intimate knowledge of the inner workings of the robots in order to create music with them. Users should be able to direct the music of the swarm robots by influencing the motion in a straightforward manner.

Hypothesis

Given the design constraints established -- using acoustic sound, rhythm and strict timing with sequencers, motion of the swarm agents to affect the musical output, and requiring that humans interact with the robots in some way as if the swarm robot system were a new instrument -- I hypothesize that motion to mappings that are more understandable as determined in a problem solving context will also be the ones that are more preferred by users in a free form context where the mappings are known. Conversely, users will choose to spend less time with mappings that are difficult to understand, even if the mappings are explained. Additionally, regarding the modes of human interaction, I hypothesize that modes of direct and intermediate action on the robots will generally be preferred to indirect choreographed methods. To justify my reasoning behind this hypothesis, I consider interaction with a musical swarm as a series of epistemic and pragmatic actions. Epistemic actions are defined as those that a person performs to change his or her own state of thinking, whereas pragmatic action brings an individual closer to his or her goals [20]. Kirsh and Maglio explore this in a task involving the game of Tetris. Users achieve higher scores if they can change the positions of falling game pieces rather than if they were only allowed to plan out their moves first before interacting with the controls. Rotating the pieces help players see how different pieces fit and thus, their level of virtuosity, or skill, increases. Human-swarm interaction could also be seen as a mixture of pragmatic and epistemic actions. Since music-making with robotic swarms is not a common musical task, it is reasonable to assume that people working with a swarm for the first time might come to use the swarm from a problem solving perspective or a position of curiosity. Under these circumstances, users might be more likely to prefer motion to mappings that are more legible especially if they should decide to compose with those robots. An alternate scenario would be one in which users are informed about what a musical swarm can do. In this situation, with advanced knowledge of the system, perhaps they would still prefer to use mappings that are more understandable.

Keeping these definitions in mind, I test the claims of my hypothesis in a user study with a swarm robot simulation in two different scenarios. In the first scenario, the goal for the users is to try and determine the musical mapping; the mapping with the least amount of time taken to guess correctly is the most easily understandable. In the second scenario with a different set of subjects, there are no time constraints. The users interact with the swarm for however long they wish, as long as they go through all mappings and interaction modes at least once. I evaluate all of the motion to music mappings as well as interaction modes using the time spent in each to determine legibility of the mappings in the first scenario, and then general interest in the second. The data are then supplemented with three questions asked of each participant, where they are asked their preferences with regards to the motion to music mappings and the modes of interaction in each scenario. If the more understandable mappings from the first scenario are also those preferred by users, then I can say that legibility might be an indicator for enjoyment or appreciation. Additionally, determining their preferences for these modes of interaction tell whether direct or indirect modes of interaction are preferred in both problem solving and free form interaction contexts.

Contribution

The contribution of this research lies in a number of areas. I developed a real-time interactive simulation in the Processing environment [21]. This simulation is novel in the application of multi-agent control theoretic methods to a musical task. Specifically, the so-called Euclid function is used with parameters derived from multi-agent control theory to fill in sequencers for each agent [22]. The potential benefits to the field of multi-agent control a means for reporting on the state conditions other systems that simulate multi-agent control, such as micro swarm satellites, or multi-sensor networks; listening to the changes in music could be used as a supplementary indictor for changes in the agents. This research also contributes to the field of interactive music by developing new ways of thinking about music for decentralized systems. Potential benefits to this field include applications of agent-based control to other areas such as generative music systems. Another novel aspect of this work is the creation of a boid-simulation constrained to match expectations of a swarm robotic system for acoustic music; it also uses sequencers as a primary means of music creation and uses the Euclid function to map motion parameters to fill in the sequencers of each robot agent, something which previous swarm simulations have not done. This simulation offers potential benefits to the fields of interactive and swarming music by spurring interest in formal approaches to musical parameter mapping based on multi-agent control theory. The simulation contributes to swarm robotics, offering a new goal for swarm robot systems to accomplish creative musical tasks in ways that single musical robot systems might not be able to accomplish. Potential benefits to this field could include spurring interest in the development of a musical swarm robot "killer-app" for swarm robotics, a means for commercializing swarm robots as entertainment robots.

In the next chapter, I give an overview of musical robotics, highlighting key aspects of human robot interaction. Then in Chapter 3 I give an overview of the boid algorithm and then swarming music, examining musical mapping strategies as well as different methods of human interaction. A brief introduction to multi-agent control theory follows in Chapter 4, focusing on the consensus equation as a means for coordinated motion. Chapter 5 will describe the multi-agent control simulation that implements the consensus equation, explaining how I sonify the motion of agents in terms of pitch and rhythm with sequencers. I then describe in Chapter 6 the swarm robot simulation that takes into account the considerations of acoustic sound and other limitations of the real world, using mapping strategies and interaction modes derived from the previous multi-agent simulation. What follows in Chapter 7 is a description of a user study with the swarm robot simulation to evaluate the legibility of the motion to music mappings, and then determine user preferences of mappings and modes of interaction in both problem solving and open-ended contexts. I evaluate and discuss the results of the user study in Chapter 8. Finally in Chapter 9 I conclude the thesis by providing a glimpse of the development of a real swarm robot system that will incorporate the findings in this thesis.

CHAPTER 2   SWARM AND MUSICAL ROBOTICS

The design objective of the thesis is to simulate swarm robots that make acoustic rhythmic music influenced through their coordinated motion and through human interaction. First I will give a brief description of swarm robotics, their use in a few common tasks, and two cases that represent first attempts at musical swarm robots. Then I will provide a description of musical robots, focusing on a few particular robots that highlight ways humans can exert control over the robots and the resulting music.

Swarm Robotics

Swarm robotics typically involves the use of many small and simple robots to accomplish tasks that a single robot might not be able to achieve. The robots are usually designed to be capable of autonomy and have sensing capabilities [23]. Swarm robots and the algorithms them are also inspired by biological systems. Concepts such as stigmergy, which is the process of agents affecting their environment in order to direct other agents to accomplish a task, can also be employed [24]. An example of stigmergy can be found in ant colonies that leave behind chemical pheromones for other ants to follow. The coordinated motion of robotic agents is often of concern especially when considering how robots ought to achieve specific tasks. Many examples of future swarm robotic applications involving coordinated motion include search and rescue [1], mapping dangerous chemical concentrations [2], or helping humans achieve a specific task such as providing support in navigation and safeguarding humans in a firefighting scenario [25]. Applications for human and swarm robot interaction are relatively new, but when considering movement of the robots, such applications often try to balance autonomous and guided forms of control [26].

Attempts at Swarm Robots for Music

Music has not been a major focus in swarm robotics. There currently is no research with regards to how musical swarm robotics could address what a single musical robot could not accomplish. There are only two examples of musical innovation with swarm robots. One is an installation called “Bd” that involves small bug-like robots that are tracked by their LEDs with a camera. The music is produced by a central computer system that tracks the robot positions. Because of the fact that the robots do not communicate with each other or apply any sense of coordinated motion, they would not be categorized under common definitions of swarm robots [27]. However, they do deserve a mention as a type of multi-robot system for music that exhibits uncoordinated swarm-like behavior.

The second example is James McLurkin’s Swarmbots created at MIT in cooperation with iRobot [28]. These robots are not intended to be used for music, but McLurkin used music as a means to test different components of the system. Each robot has a Java synthesizer and a 1.1 Watt audio system. With these swarm robots he makes a choir, using it as a way to test out a temporal synchronization algorithm. In his demo there are three phases that are mediated by a human, though the algorithms and behaviors do not require human interaction. The robots are given a musical piece represented by a MIDI file. Each robot picks an instrument to play according to the list of instruments used. Then a leader is elected using one of the gradient algorithms described in the paper. All other robots synchronize with respect to this leader, allowing them to play in time. Finally, a clustering behavior moves robots that are playing the same instruments into groups. In this scenario, motion is not used as a means of control over robots; the only effect that motion has upon the resulting music is in localization of similar sound sources. However, in another scenario with the same robots used to test a counting gradient, human subjects pick up and move robots near to each other. For a video of the counting behavior see [29]. As new robots are added to the scene, the sequencer pattern of the robots varies both in terms of voicing and in the number of hits per repeated sequence. Although the robots are not moving by themselves, humans directly interact with the robots, changing the robot's proximity to each other, and thus directly changing the music. Given the capabilities of these robots and how the music is made to reflect the state of the robotic swarm, they could also be used to reflect motion dynamics; unfortunately, there is no significant published research in this area.

I will now explore the field of musical robotics, specifically focusing on those that adhere to the constraints of acoustic sound and human interaction.

Musical Robotics

Musical robotics has a long history; some of the oldest musical robots are keyboard based, such as Fourneaux's player piano. There are many different types of musical robots, including percussive, bowed, plucked, and wind. For a comprehensive history of musical robots see [30]. There are also many design variations among these different types of robots. One such example, the Waseda flute robot, is designed to mimic human behavior precisely to understand the dynamics of human motor control [31, 32]. Drexel's METLAB makes use of robots from the Korea Advanced Institute of Science and Technology (KAIST) to better understand human dynamics and eventually perform musical tasks with a level of repeatability that cannot be attained with human subjects [7]. There are two examples of human robotic interaction that have relevance to my work, based on the design constraints for my swarm robot system: Pat Metheny's Orchestrion, and Haile and Shimon from the Georgia Tech Center for Music Technology (GTCMT). Both of these camps share the desire to use acoustic instruments to give a richness of sound production that cannot be achieved by computer simulation. However, their approaches to human robot interaction are quite different. I categorize robot interactions with humans based on direct control, in which humans are primarily responsible for the music, and indirect control, in which the robots themselves influence the direction of the music. Intermediate control I define as human robot interactions that use both direct and indirect approaches. I explore if the approaches taken in these musical robotic examples might work from a swarm robot perspective, considering that motion is the primary means of affecting music output.

Robots as Extensions of Humans

The Orchestrion project consists of many different types of robots in the ensemble, some created by League of Electronic Musical Urban Robots (LEMUR), including pianos, marimba, vibraphone, orchestra bells, basses, guitarbots, cymbals and drums, blown bottles, custom-fabricated acoustic mechanical instruments, and a very large percussion installation that serves as the background and rhythm section [33]. Metheny cites a number of reasons for commissioning the construction of the Orchestrion robots, including the creation of platform for musical composition, improvisation and performance, the development of ensemble-oriented music using mechanically controlled acoustic musical instruments, and a redefinition of the idea of “what constitutes a solo performance by a single musician” [34].   Metheny emphasizes full control of the human musician over robotic sound generation. He composes the parts for each instrument by using his guitar as a MIDI instrument. The instruments play certain rhythms or melodies that are prerecorded or they mimic the parts that he himself plays with his guitar [35]. Additionally, he makes use of foot pedals that trigger different musical sections or modes using an Apple G5 with Digital Performer as a means of keeping time[36].   In this single human to multiple robot scenario, Metheny treats the robots as advanced MIDI instruments. He describes them as being “completely agnostic. They don’t care whether they’re getting an instruction from the guitar, from some kind of keyboard input, from some kind of digital paper like Sibelius or Finale or whatever. They’re just waiting for instructions.” The robots are not autonomous in nature. The music that results is meant to be an extension of his own playing and is typically described as such in reviews of the Orchestrion album. I categorize this type of approach as a means of very direct control.

Robots as Musicians

In contrast to Metheny's approach of direct control over musical robots, there are two examples that focus extensively on two-way human-robotic interaction. Haile and Shimon, created at the GTCMT are designed to be robotic musicians. As outlined in their work with Haile, Weinberg and Driscol seek to explore robotic musicianship as an answer to the limitations of computer simulated music [8]. In their opinion, computer based interactive music systems do not provide players and audiences with physical and visual cues needed for expressive music interactions. Additionally, they are limited by electronic reproduction of sound, which in their opinion does not capture the same richness as acoustic sound. Haile is a robotic drummer designed to address this problem bringing both computational power and rich acoustic sounds through physical means. While the robot is capable of imitation and accompaniment with MIDI files, other modes of interaction give Haile more creative control; for example, Haile can take in user input and stochastically transform the output rhythm. It can also perceptually analyze the amplitude and density of human playing and modify its own playing to play along inversely, allowing humans to perform solos, or in a direct relationship where the louder the human plays, the louder the robot plays as well. Haile listens to live human players, analyzes perceptual aspects of their playing in real-time, and uses the product of this analysis to play along in a collaborative and improvisatory manner. I categorize this mode of Haile's interaction with humans as intermediate form of action; the human initializes the musical process, but the robot responds to human input in ways that the human user can't necessarily predict.

Improving upon the goals initially set forth by Haile, GTCMT researchers developed a newer marimba playing robot Shimon that adds the extra dimension of pitch

[9]. Additionally, it was designed to provide expressive interpersonal musical cues through the use of an embodied robotic head containing a camera. The purpose of the robot is not merely to play better than a human could but also to play with humans and inspire them to perform musical works. Like Haile, Shimon is also capable of many different modes of human interaction. In a mobile phone application called ZOOZbeat, Shimon is capable of repeating a musical pattern based on user input [37]. However, it can also weight its response to human input using Markov chains based on the works of famous jazz musicians such as John Coltrane and Thelonius Monk. The resulting music can have mixed characteristics of both the human's initial input, or it could be based completely off from another composer's style. In this way, Shimon has the ability to play directly, in terms of imitation, indirectly, by weighting the Markov chain output more than the human's input, or intermediately by using a combination of both the human and probabilistic models.

Application to Swarm Robots

Applying Metheny's approach to human robot interaction in music would involve considering the swarm as an extension of the human:, the human controls the music and the robots simply obey. Adhering to the constraint that the motion of the robot swarm should cause the changes in rhythmic music, a direct human and musical swarm interaction would involve taking control over the robots' motion, either collectively or individually, in similar ways that Metheny does with his robots. This could mean forcing the robots to move to particular locations, directing them how and where to move with respect to the user. For an intermediate type of interaction, as seen in Haile's and Shimon's transformation of human user input, an analogy for indirect swarm robot motion control might also be transformative with respect to the motion. For example, a human could provide input to the system by setting initial positions of the robots directly or picking up the robots and interfering with their motion momentarily; after placing the robots back down, the human could indirectly observe the robots autonomous behavior in their new location. In the way that Shimon has the ability to play indirectly by ignoring user input, indirect modes of interaction with a musical swarm might have choreographed movement gestures. The robots could move without any regard for human interaction at all. The human in this case becomes an observer, giving the robots control over the creative musical process. While all of these theories of interaction are interesting, I have yet to establish how the rhythms produced by the swarm would change. The choice of mapping from motion to music is open. However, one particular area of computer music composition that deals with the sonification of moving agents is swarming music. It is necessary to look at what swarm music in simulation has been able to achieve thus far and what strategies they use for sonification of swarm motion and human interaction.

CHAPTER 3 THE BOID MODEL AND SWARMING MUSIC

I define swarming music as a type of algorithmic composition in which there are a number of individual agents that move through a simulated space. Typically the algorithm to describe the motion of agents is governed by simple decentralized rules, such as those based on the Reynolds boid simulation. There is a large body of work on the use of decentralized systems for music. For example, one subset of decentralized systems is cellular automata, which include such well known examples as the Game of Life [38]. In these situations, elements of a typically two-dimensional array obey simple rules that result in complex life-like behaviors. Examples of musical applications of cellular automata include Wolfram Tones, which generates musical sequencers of different genres [39], and Chaosynth, which is an approach for sound synthesis using a neuronal based rule-set [40]. For a history of cellular automata approaches as well as evolutionary computational music, see [41]. Whereas cellular automata are abstracted and quantized models of artificial life, the attraction of the boid simulation lies in its decentralized approach to simulating very convincingly the natural behaviors of flocks of birds and schools of fish in both two and three dimensional settings. I will explain the basic form of the boid simulation and then lead into swarming music. In particular I will examine three interactive systems by Blackwell, Hsu, and Bisig in the field of swarming music. All three of these systems use variants of the Boid model and provide interesting modes of human interaction with the virtual swarm; however, they differ in terms of their approaches to musical mappings. I will look at the choices of musical mappings in terms of their straightforwardness, or legibility, and comment on their applicability to a swarm robot system.

The Boid Model

Reynolds' definition of boids is informed by the concepts of decentralized systems, in particular, Papert's work with Logo Turtles [42]. The Logo Turtle was initially a robot used as an education tool to help children learn about geometry, arithmetic and programming. It was a mechanical robot that crawled on sheets of paper upon a classroom floor, drawing figures by dragging markers along the paper while it moved. The paths taken by the turtles are very similar to those taken by the boids. The boid algorithm takes into account a radius of effect, which creates the decentralized behavior. Each agent behaves according to three basic behaviors of separation, alignment and cohesion; all of these behaviors affect the velocity vector, a combination of the heading and speed of the agent.

Both separation and alignment are complementary functions. Separation takes into account the desire of an agent in flock to avoid collisions. Members of the flock want to fly without running into each other and thus steer away from impact. Collision avoidance attempts to ensure that there is a minimum separation distance; it requires knowing the position or the relative distance of nearby neighbors. Alignment, also called velocity matching, is a type of predictive collision avoidance; if an agent matches the velocity of its neighbors (specifically the heading), then it is unlikely that they will collide.

Cohesion refers to agent's tendency to move closer to the center of the flock. While in previous works this may be taken to mean the center of all of the flock members, what it means from an agent based perspective is the center of nearby flock-mates since each agent has only a local perception of the world around it. An agent residing in the center of the entire flock is affected homogenously by other agents around it; in this case, the resulting centering force will be small. Conversely, an agent on the boundary of the flock has a centering force pulling it towards the overall centroid of the flock. Using this approach also allows for the flock to go around obstacles more easily, as real flocks sometimes have to do this; a single agent only cares about its own local neighborhood.

All three of these behaviors can be used simultaneously, as they each result in a new velocity vector to be added to the robot's current velocity. In order to find the best combinations of the velocities that each behavior provides, they can simply be averaged together or weighted to give different dynamics to a swarm. These three behaviors result in flocking. Reynolds notes that "each boid must reason about each of the other boids, even if only to decide to ignore it" [13]. Computer programs that simulate boids oftentimes give each agent access to global information such as their neighbors' absolute position, to determine proximity and other characteristics.   Reynolds' work with this decentralized approach to simulating flock behaviors inspired research in other areas dealing with swarm robotics, especially in the areas of decentralized multi-agent control, which I describe in the next chapter. The flocking algorithms are also used in animation, most famously in the stampede scene in the Disney animated film, *The Lion King* [43].

The strength of the boid algorithm is such that by influencing motion through the addition of velocity vectors, modes of direct, intermediate, and indirect interaction with the agents are possible simply by varying the weights of the flocking behaviors. For example one additional feature of Reynolds boids includes seeking towards a target. The flock moves to a target location while simultaneously taking into account the relationships between their neighboring agents. Swarming music exploits this behavior, giving users direct and indirect control over flocking agents in order to guide musical processes.

Swarming Music

One of the first to publish in the area of swarming music is Tim Blackwell. He uses an implementation of the boid simulation and visualization both for real time music generation as well as interaction with humans. SWARMUSIC is an improvisational music system created for his master's thesis [44]. The simulation program has a number of independent systems that are each responsible for its own swarm. Each system has a capture phase to get MIDI events from the swarm itself or outside sources, an animation phase which draws the swarm on a screen and updates positions, and interpretation phase by which the positions are mapped somehow to music. There are parameters that can be adjusted that allow for real time conduction or left fixed for autonomous behavior. In the animation phase, rules similar to Reynolds' boid algorithm are used to direct agents in a three dimensional virtual cube. The interpretation phase maps the coordinate axes of the virtual space through which the agents move to pitch, pulse, and loudness of the music. Pitch and loudness refer to MIDI note numbers and to the MIDI velocity respectively. Pulse refers to a time interval between preceding note events; there are some arbitrary limits set on the range of this mapping so that the swarm will not play too fast or too slow. If the swarm moves along one side of the pulse axis it will play more frequently and conversely more slowly if on the other end. There is no guarantee that the agents can synchronize. In the capture phase, the pulse pitch and loudness events created by other swarms, or generated by outside events such as a human user input, are recorded and mapped as targets for the agents in the 3D virtual space. The placement of these targets in the virtual space provides stimuli for motion, resulting in changes in the swarm's music. This phase also makes use of scripts to allow for different modification of targets, as well as making a number of agents sound at the same time to form chords. A number of pieces make use of conducted improvisation of the swarms, autonomous swarm music creation, and autonomous swarm and human interaction. Additional components of the SWARMUSIC system provide other modes of interaction where a vocalist can sing and create capture targets for the swarm through pitch tracking. The swarm hovers around these pitch targets, thus making it sound like the swarm was improvising around the vocalist's input [45]. Blackwell's work uses swarms to primarily explore pitch mappings and both autonomous and guided interactions with swarms. The approach Blackwell takes is a very straightforward one with respect to the mapping of music parameters. However, the mapping of absolute positions of agents to musical output is problematic with respect to robotic swarms since he absolute position of a single robot in a room has no inherent meaning with respect to the robot's motion. Mapping pitch and rhythmic interval along an arbitrary direction in a room does not sonify the agents but rather the room itself.

A different take on swarm based music by Hsu uses an inverted approach in which the music of a human performer causes the motion of particle simulations that exhibit swarm-like behavior[46]. In a piece called *Interstices*, up to hundreds of thousands of particles are manipulated with graphics tablets, multi-touch devices, and audio input. Other components in the simulation such as attractors, repulsors, and fluid-like simulations are also added to affect the particles. Each individual agent is not sonified. Instead the human gestures used to influence the swarm motion are sonified as they affect the swarm simultaneously. Performers' gestures with the interfaces can be used to stir the particles, moving them into clusters. The sounds made by the simulation are "synthesized by specifying high-level sound synthesis parameters such as duration, loudness, brightness, amplitude modulation etc. In a particular section, large physical gestures may result in loud, bright sonic gestures of long duration; in another section, sonic gestures may be restricted to shorter durations with very low brightness." Hsu decided to avoid "straightforward" mappings in this piece. An example he gives of a straightforward mapping is one in which the brightness of the particles corresponds their position. As an opposing example, he describes a situation where the "onset and continuation of a slow and loud sonic gesture may trigger a large tidal current in the animation; if the roughness of a sonic gesture is maintained above a threshold for a minimum time, a particle cluster will be triggered to coalesce into an image." Since straightforward mapping is not a goal, the audience cannot necessarily be expected to grasp the purposes of the author. It is clear that when he performs some gestures on the tablet interfaces, some change in the particles occurs; nevertheless, it is difficult to correlate the precise changes in the sound with the gesture as it is performed. Hsu uses sample based approaches modified by granular synthesis approaches to sound generating and other high level synthesis parameters in order to create his sonic gestures. While the legibility of the mappings from an audience perspective in this case is not an explicit goal, he does use the swarm to create certain recognizable shapes and configurations as part of his pieces, for example, having the particles formulate a human skull. This can be categorized as a type of very direct interaction with the swarm because he forces the agents to move to specific locations as defined by his images. Using agents to form recognizable shapes can be an interesting tactic for swarm robots; however, establishing the meaning of those shapes is problematic. For the work in this thesis,, the coordinated motion is the primary interest, not the arbitrary shapes that agents might make. Additionally the constraint I set is that the motion of swarm agents causes the music, not the other way around. However, Hsu's work provides a counterexample to the assumption I make that straightforward understandable mappings would be preferred with a swarm robot system. It is possible that a swarm robot system could still be of interest to users if the motion to music mappings are nebulous.

Unemi and Bisig make a unique way to interact with a simulated swarm through the use of a computer vision system [47]. By capturing the motion of humans using frame differencing, areas of high motion are used as targets toward which the agents flock. Similarly to Blackwell's simulation, the agents play MIDI instruments using pitch and velocity information. Agents respond by moving through the virtual space as well as by user interaction. Whenever the swarm is attracted by the user motion it plays a primary instrument, otherwise as it behaves autonomously it will play a softer secondary instrument. Instead of using one axis to determine time interval between notes, each swarm agent has a certain probability to play notes and the x axis is used for panning; in this way, stereo position of the output corresponds to the visual position. Loudness is controlled by the z position with agents moving closer to the screen being louder than those that are further away. Bisig and Neukom also outline several categories of sound mappings that they make possible through the Interactive Swarm Orchestra (ISO), a swarm toolkit for music and interactive dance and art installations [48]. The first category they describe is parameter mapping, whereby agent properties are mapped directly to musical parameters. An example of this would be an additive synthesis approach – mapping y position to frequency, z to amplitude and x to panning. This approach has the same problem associated with it as Blackwell's, where the space is being sonified as opposed to the agents themselves; however, mapping the panning parameter takes into account the perspective of the user with respect to the screen, helping them to localize specific agents. Proximity based events form the second category described by Bisig and Neukom. As an example, sample triggering can be combined with placement of sound sources in virtual space; as the agent moves nearby to the sound source, agent parameters affect the sample unit parameters, causing it to play or be modified. The third category, called procedural patching, is an idea where a patch is constructed on-the-fly as a result of swarm interactions. One example of this is modulation synthesis; as the swarms come together their positions are mapped to different frequencies of oscillators, and are patched only when they are in a neighborhood. Hence clusters of agents will have more modulation, whereas sparse or solitary agents will sound more like additive. The ISO is capable of changing musical behavior of the swarm based on their proximity to one another and can also use other parameters like the velocity of the agents. Where Bisig's method for mapping differs is in the direct use of proximity for affecting the music. This type of mapping is more relevant to a swarm robot context since it is defined in relation to another robot as opposed to an absolute position.

There are common problems with swarm music strategies discussed here. One cannot differentiate between the spaces through which agents move from the agents themselves. It is unnecessary to build swarm robots to sonify a physical space when simulation ought to accomplish this. Additionally, the swarm robots are expected to play acoustic sound as opposed to extracting information about their movement using a separate computer to create the music. With this constraint, these swarm simulation approaches to making music and creating mappings may not work well. For example, mappings of volume may not make sense for a robotics platform that moves along the floor when people can view the robots from a variety of different angles. Granular synthesis or other sample based methods are certainly not applicable. Additionally, if my constraint is to use physical/mechanical means to create sound, it might only be feasible to play one or two notes per robotic agent. Creating a swarm robot that can play a variety of notes would be a more difficult task. The previous simulations did not have to take into account the constraints of physical sound production, nor do they even have to impose constraints on realistic movement speeds for the agents.

In summary, when we look back at these examples of swarm music simulations, the positive aspects we can note in a swarm robotics system are the unique interesting modes of interaction with the simulation. Some of those almost always include taking direct control over the swarm's trajectory, making them seek specific targets. While the typical musical choices tend seem to focus on pitch events, there is not much guarantee that events will have any sense of meter. Blackwell explores synchronization, but no mention is made of metrical structure. Sequencers have not been explored in any of these boid-based simulations. Since sequencers are one of the most commonly used timing tools in music making I will explore their use in the next chapter with my simulations. Swarm simulation music does result in music that is affected by the motion and interaction of agents. However, the motion itself is not actually being sonified in all of these cases, but rather the absolute position of the agents within an arbitrary virtual space.

I want the swarm robots themselves to make the music, irrespective of their environment. Position, velocity, and proximity are explored by all these simulations to some varying degree, but no single simulation uses a formal methodology for describing the most essential parameters of motion for swarm agents. If swarm motion is the primary means by which robots change music, regardless of the viewing perspective, then it is necessary to find set of parameters that are specific only to the agents themselves and independent of the environment. Multi-agent control theory addresses this problem. In the next chapter, I will show how multi-agent control theory can provide us with a formal way for analyzing and discovering these parameters.

CHAPTER 4 MULTI-AGENT CONTROL THEORY

In this chapter I give a basic introduction to concepts from multi-agent control theory. I will use this as a more formal means to determine musical mapping and interaction parameters that succinctly describe swarm motion. Incidentally, Reynold's boid model is one of the major influences that stirred research in this area. Control theorists found a more formal approach based in nonlinear control theory to try and replicate the behaviors seen in the boid model. In order to design a swarm robotic system, it is useful to look at the theoretical means by which we can coordinate agents without the use of absolute positioning. A basic problem concerns groups of similar robots that have no means of knowing their own absolute position, but could determine relative distances. One such application involves creating controllers to cause these types of robots to move to specific formations [49]. Multi-agent control theory addresses how these robots can coordinate their motion. A more complete reference on the history of control theory as well as how to understand consensus requires an introduction to graph theory and some basic linear systems theory. I refer the reader to [15] as a reference from which all the equations and explanations of multi-agent control theory in this chapter are derived.

Agreement Protocol and Consensus

The agreement protocol for coordinating multiple agents is a fundamental problem in multi-agent control. It does not necessarily have to deal with swarm robots; in general, the idea is that a collection of agents is meant to agree upon some value. The agreement protocol involves *n* agents where all are interconnected in some manner. By interconnected, I mean agents have a means to obtain information about each other. We can represent the state of the network of agents G with a graph. The graph of G is called connected if for every pair of vertices (agents), there is a path that has them as its end vertices. If this is not the case, say for example if one vertex is unconnected to the rest, then this graph is called disconnected. A directed graph, or digraph, is one in which the connections between vertices have direction associated with them. What this might mean in the case of a swarm robot network is that agent A may be able to detect agent B, but agent B may not be able to sense A. Hence the directionality of the connection, or edge, would be described being from A to B. A digraph is called strongly connected if for every pair of vertices there is a directed path between them. A weakly connected digraph is one that is connected if the edges of the digraph were replaced with two way connections, that is to say, turned into a disoriented digraph. The degree of a vertex is the number of vertices that are adjacent to that vertex; in other words, the degree of agent A is the number of connections made from agent A to the other agents in the network. A degree matrix is an $n \times n$ diagonal matrix that contains the degree of each agent along the diagonal. An adjacency matrix is an $n \times n$ matrix that describes the degree relationships of the network. Each column in the matrix corresponds to a connection from one agent to the others; the diagonal element does not matter as that represents a connection of the agent to itself. The graph Laplacian is the difference between the degree matrix of a network of agents and its adjacency matrix. The Laplacian can also be defined for digraphs in a similar manner, though the diagonal degree matrix is modified to only contain the weighted in-degree of a vertex v, meaning the number of connections going into an agent. Now, given this terminology, it is assumed that the rate of change of each state is governed by the sum of its neighboring states. The rate of change for entire system of agents can be described as

$$\dot{x}\ (t) = -L(G)x(t) \qquad (1),$$

where L(G) is the Laplacian of agent's network G, and x(t) is a vector representing the state of each agent. An example of the agreement protocol is in the consensus problem, where a collection of mobile robots is to meet at a single location. However, they can only measure relative displacements. The resulting equation,

$$\dot{x}_i(t) = -\sum_{j \in N_i} (x_i(t) - x_j(t)) \qquad (2),$$

gives a situation in which all the robotic agents will move towards some location. Each $\dot{x}_i(t)$ represents the velocity vector that an agent i must set for itself and the set N_i represents the set of connections to other agents that the agent i can see. Under the right conditions of the connections between agents, the location that the robots move towards is the centroid of the robots. Specifically, this condition is when the directed graph that describes the connection between agents contains what is called a rooted out-branching and is balanced. A rooted out-branching means that the graph contains no directed cycles within and it has a vertex a such that for every other vertex b there is a path from a to b. A directed graph is called balanced if for every agent, the in-degree and out-degree are equal. When the consensus equation is run, these agents will meet at their centroid. An easy way to make a balanced graph is if the adjacency matrix is symmetric. This does not ensure connectedness, meaning that not all agents may meet at the centroid; however, the centroid of the network of agents at any time with a symmetric matrix will never drift from the initial centroid. If I wanted to impose some constraints on the motion of the robots, I can modify the consensus equation. For example, if I wish that the robots should maintain a minimum distance between their interconnected agents, then the consensus equation can be modified as follows:

$$\dot{x}_i = -\sum_{j \in N_i} w(x_i - x_j) * (x_i - x_j) \qquad (3).$$

where w is given by,

$$w(x_i - x_j) = \frac{\|x_i - x_j\| - \delta}{\|x_i - x_j\| - \varepsilon} \qquad (4),$$

δ is the minimum separation distance specified, and ε is some small value close to 0. As another constraint, if I want to make the agents take a rotational pathway towards their intended target, then the consensus equation can be modified as follows:

$$\dot{x}_i = -\sum_{j \in N_i} (x_i - x_j) * R(\theta) \qquad (5),$$

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \qquad (6).$$

Finally, one last modification I can make concerns the relationship of the adjacency matrix. If we use a distance Δ to indicate the range that an agent can see another, then this will cause a connection between the two agents. This is called a Δ-Disk proximity graph; an edge is defined between two vertices if the relative distance between them is less than or equal to some distance delta.

Combining all of the equations together can give us a result

$$\dot{x}_i = -\sum_{j \in N_i} w(x_i - x_j) * (x_i - x_j) * R(\theta) \qquad (7),$$

where w and R are as described earlier. With this formulation of the consensus equation as well as additional parameters to control such as the rotation, separation distance, and the Δ-Disk proximity distance, there are many types of coordinated motions possible. I will illustrate this with a Processing simulation that uses the degree and the velocity vector for musical mappings and the free parameters of rotation, separation distance, and the Δ-Disk proximity distance to allow for real time interaction. With this simulation I can begin to explore motion to music mappings that both adequately describe the motion of robotic agents and provide additional control to influence the motion.

CHAPTER 5 MULTI-AGENT CONTROL SIMULATION

In an effort to achieve robust musical mapping and interaction parameters for use in a swarm robot simulation for music, I now develop a simulation in the Processing environment that utilizes the parameters from the consensus equation, as described in the previous chapter. I experiment with motion to music mappings as well as interaction parameters in order to help inform the design of the swarm robot simulation program.

Motion Design Features

The simulation is designed in Java using the Processing integrated development environment because of its ease of use in creating animation. Sound is produced using the Promidi 1.0 library and the deluxe Java Sound API soundbanks, which provide 128 different types of musical instrument options [50]. The simulation only uses one instrument at a time.

The simulation can be initialized with an arbitrary number of agents, drawn with a circle on the screen with an ID number to distinguish one from the other. The initial positions can be set randomly or manually selected both before running the simulations and also in real time. By default, the simulation starts off with a group of ten agents so that the adjacency matrix can be displayed reasonably well while the simulation runs without obstructing the view of the agents too much. The adjacency matrix describing the connections of between agents is displayed on the top left corner of the screen. Each element i, j corresponds to a directional connection from agent i to agent j.

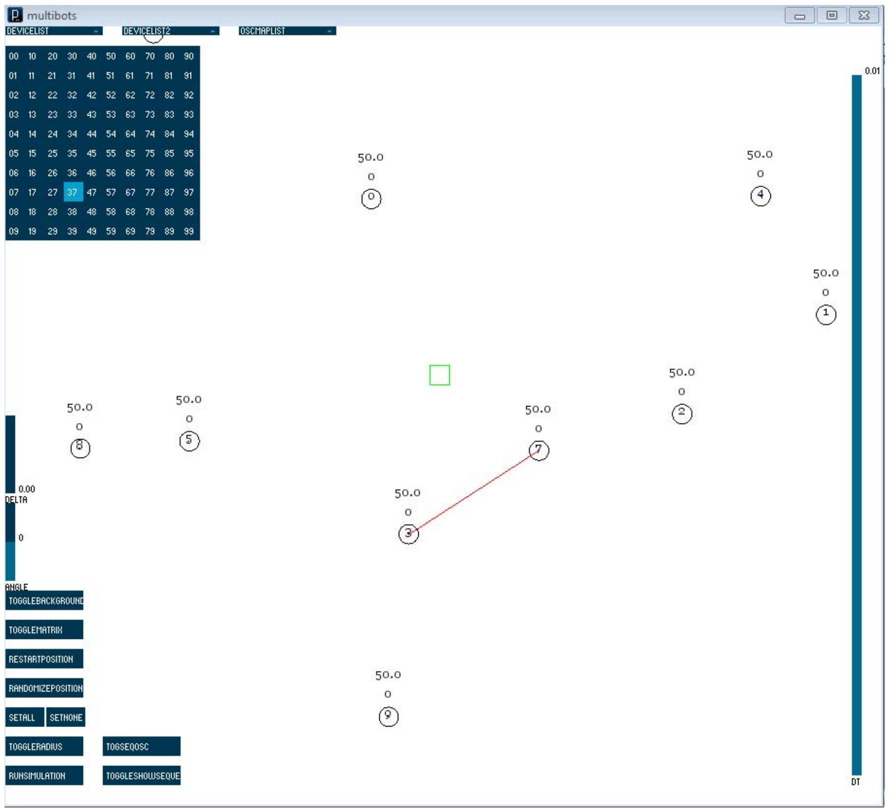


Figure 1: Directed connection from one agent to another

As shown in Figure 1, the red line indicates a one way connection as described by the adjacency matrix in the upper left corner of the screen. In this example, the column for agent with ID 3 indicates that agent 3 knows the relative distance to agent 7.

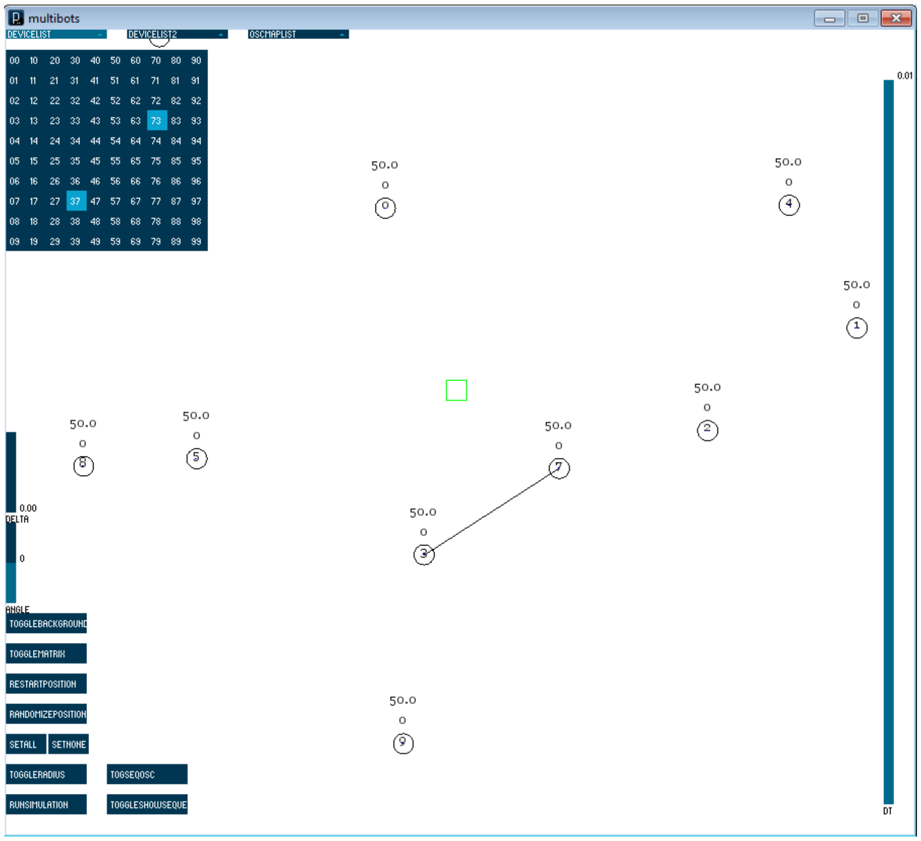


Figure 2: Two way connection between agents

If both agents 3 and 7 have a connection, the line drawn between them is painted black as shown in Figure 2. This provides an easy way to visually determine whether or not the system is a digraph.

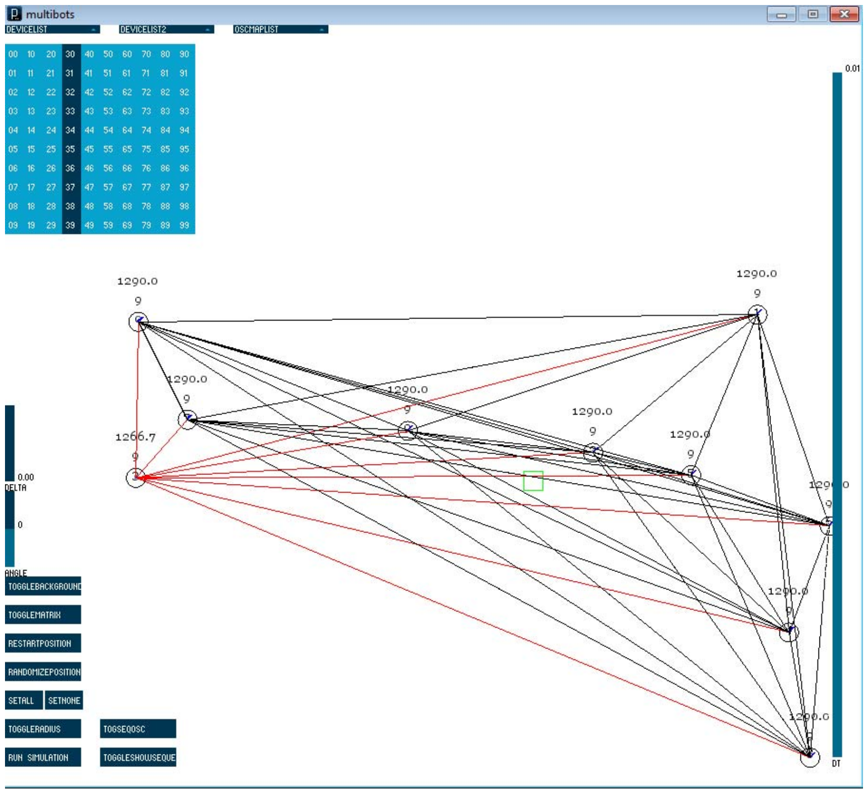
The simulation implements the consensus equation as described in equation 7 of the previous chapter. The equation is evaluated at a time step dt which can be adjusted with the vertical slider on the right side of the screen. Additional features of the program include adjustable parameters for the rotation, and minimum separation distance. The disk graph can also be toggled with a button and then a slider will appear to adjust this parameter. The application of these movement parameters will be described through examples running the consensus equation upon the agents.

Running Consensus

The consensus equation runs by pressing the “Run Simulation” button on the lower left of the screen. The basic form of the consensus equation in equation 2 occurs when the rotation and minimum separation distance parameters are both set to zero. Agents will only move based on the sum of distance vectors of their connected agents.

Movement to Specific Locations

As an extreme example, if an agent i knows nothing about all other agents, being fully disconnected, then it won't move; the velocity vector it will add to itself is zero. Figure 3 shows a zeroed out column of the adjacency matrix corresponding to agent 3. All the other agents are connected to it, meaning that they know the relative position, thus the red lines.



00	10	20	30	40	50	60	70	80	90
01	11	21	31	41	51	61	71	81	91
02	12	22	32	42	52	62	72	82	92
03	13	23	33	43	53	63	73	83	93
04	14	24	34	44	54	64	74	84	94
05	15	25	35	45	55	65	75	85	95
06	16	26	36	46	56	66	76	86	96
07	17	27	37	47	57	67	77	87	97
08	18	28	38	48	58	68	78	88	98
09	19	29	39	49	59	69	79	89	99

Figure 3: Unconnected agent 1

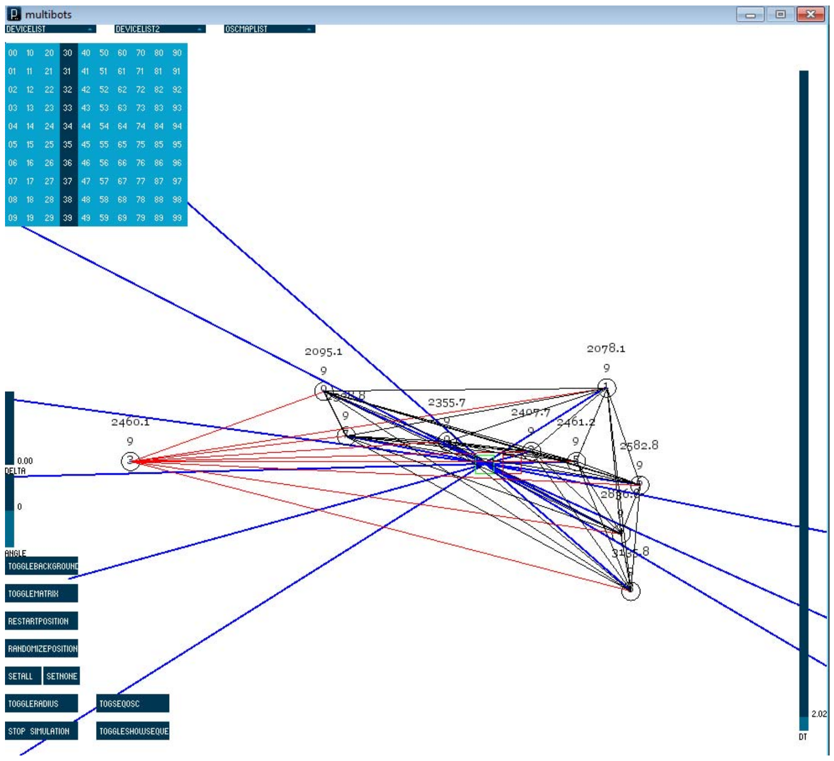


Figure 4: Unconnected agent 2

If consensus is run in this situation where the graph describing the adjacency matrix is not balanced, the centroid of the entire agents will shift over time as shown in Figures 4 and 5. All of the agents will move towards the agent with the zeroed out column, in this case, agent 3.

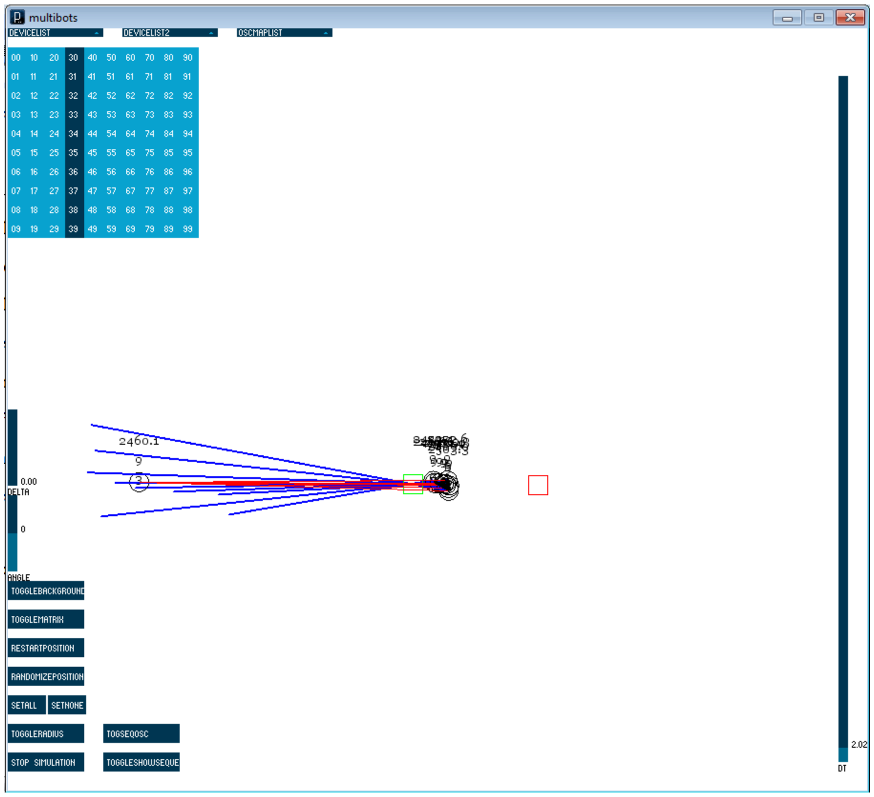


Figure 5: Unconnected agent 3

The red square seen in figure 5 indicates the initial centroid of the whole system, whereas the green square represents the current centroid as the agents move. Eventually, the green square will align on top of agent 3. This is an example of an unbalanced digraph.

In the simulation, just as I can set the initial starting points of the agents with the mouse, I can also do this in real time, forcing an agent to stay still. When this happens, the initial centroid is shown in a red square, and a green centroid is drawn showing the current centroid of the system. This has the same effect as in figures 3 thru 5 in a fully connected network, temporarily zeroing out one of the columns that corresponds to the stopped agent. I can move all of the agents to any arbitrary position on the screen. In this way I can interfere with the movement dynamics of the system momentarily and then watch as the simulation continues to run as a result of my interaction.

In a fully connected network, all of the agents move to the centroid in a straight line. The program is capable of drawing traces of these pathways as shown in Figure 6.

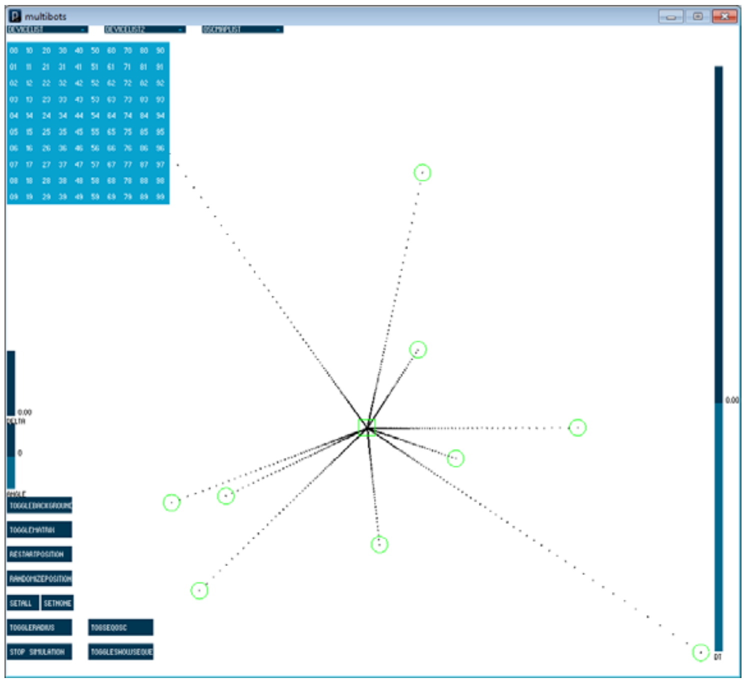


Figure 6: Fully connected network - direct path to centroid

When the graph that represents the adjacency matrix is balanced, but not fully connected, agents still move towards the centroid, but they might not take a linear path. As an example, Figures 7 and 8 show a chained two-way connection between the agents.

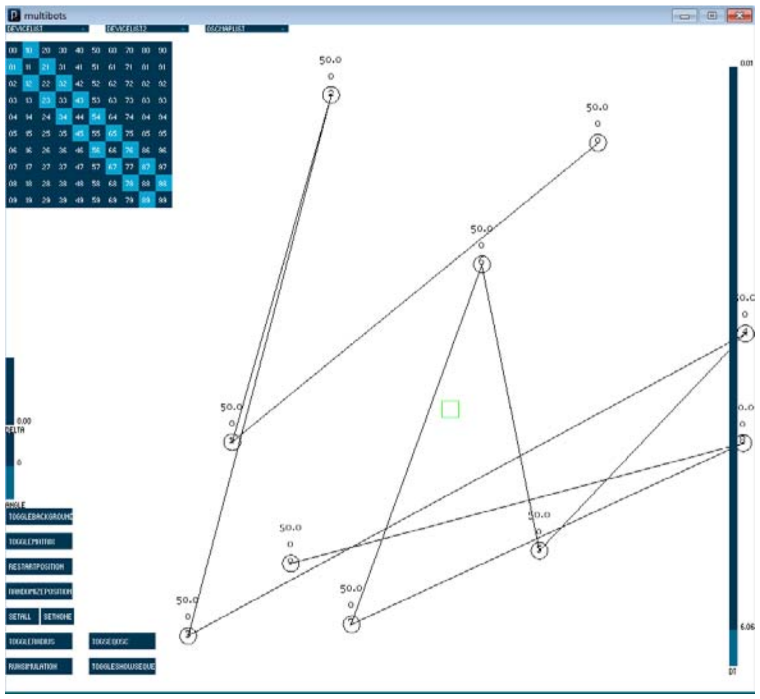
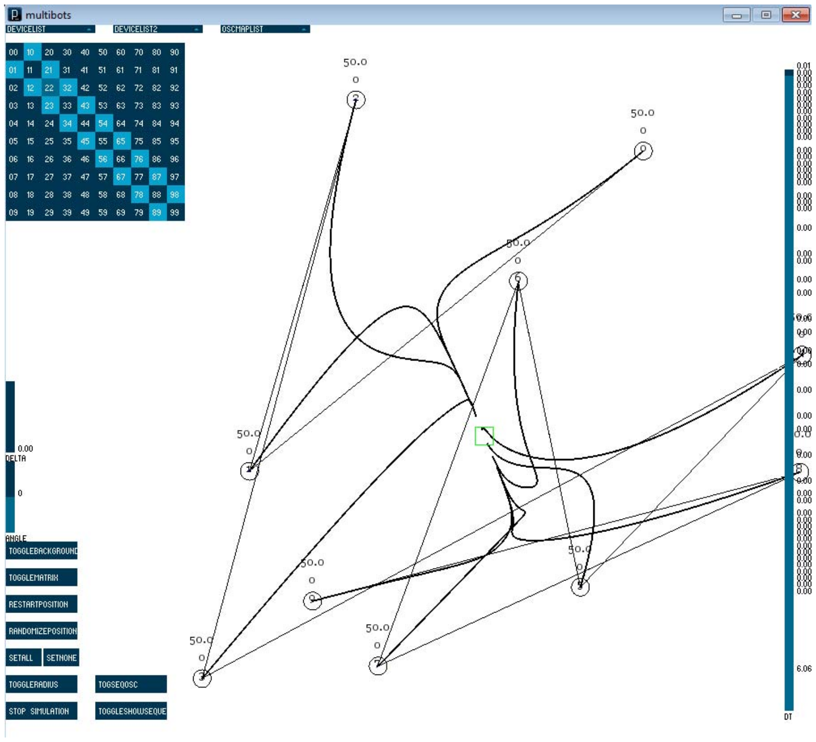


Figure 7: Symmetric connection 1

The pathways that the agents take to the centroid are more curved in nature; it also takes longer for the agents to reach the centroid than if they were fully connected.



00	10	20	30	40	50	60	70	80	90
01	11	21	31	41	51	61	71	81	91
02	12	22	32	42	52	62	72	82	92
03	13	23	33	43	53	63	73	83	93
04	14	24	34	44	54	64	74	84	94
05	15	25	35	45	55	65	75	85	95
06	16	26	36	46	56	66	76	86	96
07	17	27	37	47	57	67	77	87	97
08	18	28	38	48	58	68	78	88	98
09	19	29	39	49	59	69	79	89	99

Figure 8: Symmetric connection 2 - curved path to centroid

<u>Maintaining Separation</u>

The simulation allows for modification of the minimum separation between connected agents, as described by the modified form of consensus in equations 3 and 4. This separation distance can be adjusted in real time as well. The same behavior also exists for symmetric connections, though the minimum distance will only be maintained between agents that have a connection.

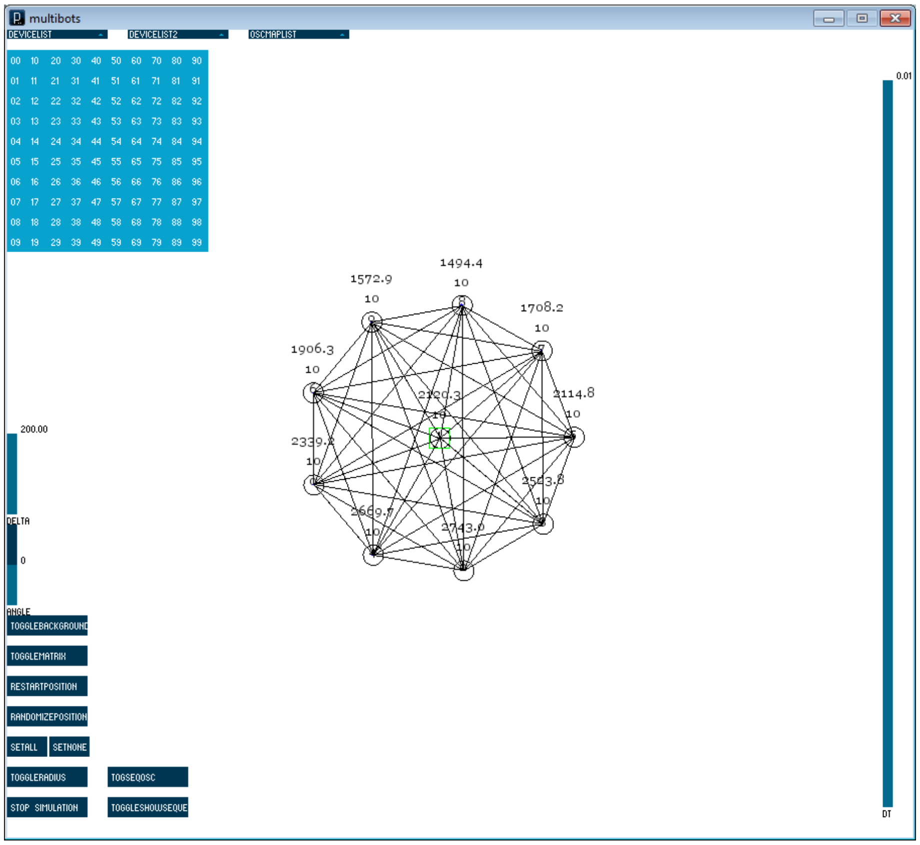


Figure 9: Fully connected network with minimum separation distance

Figure 9, shows a minimum distance separation of 200 pixels lengths, where this parameter is set using a slider bar on the left hand side of the screen. In a fully connected structure, this type of configuration with ten agents automatically forms when the distance constraint is set. It is important to note that in this idealized view of multi-agent control, the agents can only maintain a minimum distance of separation with agents that it can see. This simulation treats agents as abstract entities; as such they will appear to pass through agents for which there is no connection as described by the adjacency matrix

Rotational Movements

In a fully connected network, all agents move in a straight line toward the centroid as shown earlier in Figure 6. However, if the consensus equation is modified as in equations 5 and 6, then each agent can also rotate towards the centroid at the specified degree. The amount of rotation can also be set with a slider located below the minimum distance slider on the left hand side of the screen. When the angle is less than 90 degrees in the case of a symmetric matrix (connected disoriented digraph), the agents will spiral in towards the centroid. Figure 10 shows the pathways taken by a fully connected network if the rotation parameter is set to -54 degrees (clockwise rotation). Similarly, counterclockwise rotation is possible if the angle was positive. Any angle that is greater than 90 or less than -90 causes the agents to spread further apart from each other.

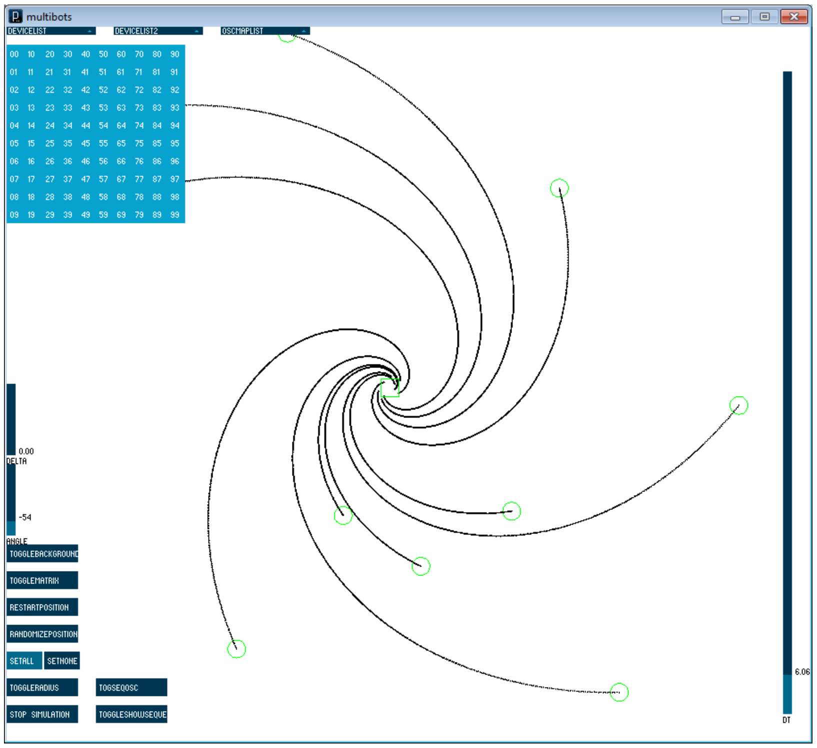


Figure 10: Fully connected network with rotation parameter at -54 degrees

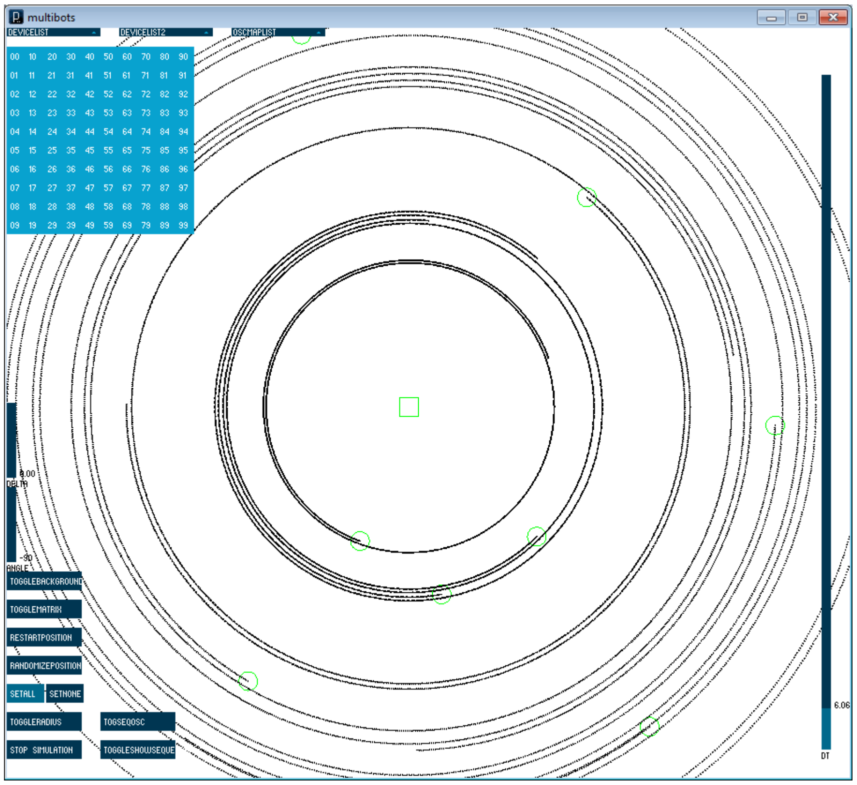


Figure 11: Fully connected network with rotation at 90 degrees

When the rotation angle is exactly 90 degrees, the agents will start to slightly expand further apart over time. Figure 11 shows what happens when the angle is set to 90 degrees. Incrementally, the pathways of each agent will get larger since at every time step the agents try to move in a tangential orbit.

In the case of a weakly connected digraph with the adjacency matrix as shown in Figure 12, the agents will maintain a stable rotational orbit if the rotation angle is π/n, or in this case with ten agents, 18 degrees.

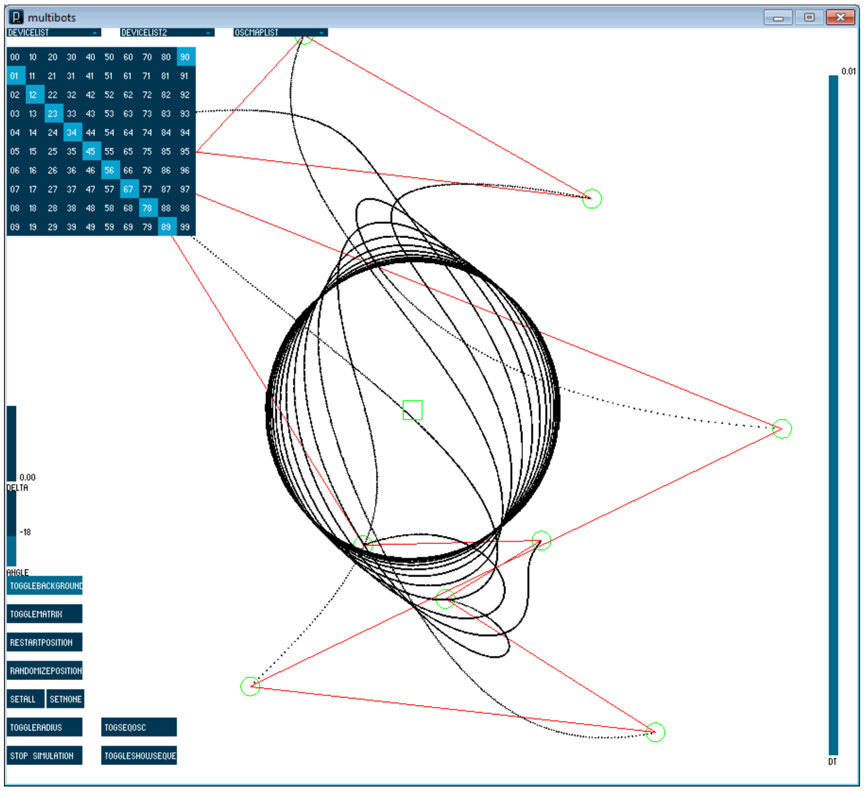


Figure 12: Directed network orbiting at fixed distance

Figure 12 shows the initial starting conditions of the agents, their connections, and the pathways taken that eventually lead to a circular orbit around the centroid.

<u>Proximity Based Motion</u>

The simulation also allows for the setting of the Δ-Disk proximity distance. This is again represented with a slider that can be adjusted on the right hand side of the screen. To enable this, first the "use radius" button must be selected so that the adjacency matrices are set by the disk as opposed to being manually selecting the connections. This disk allows for the connections of the network to vary in real time. A two-way connection will form between agents if they come within range of one another.

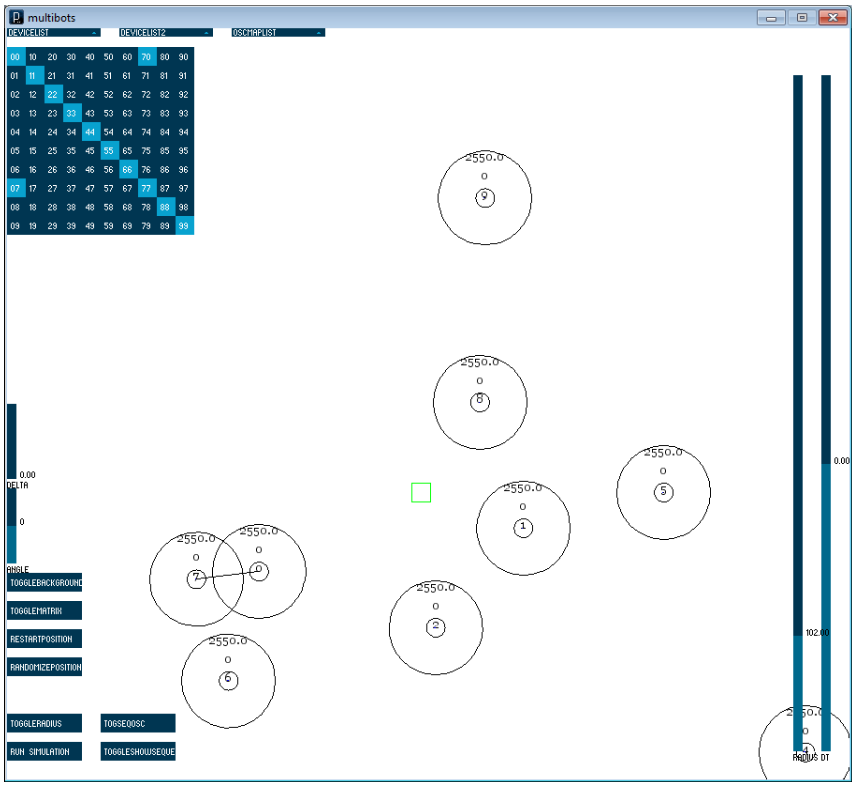


Figure 13: Network with small radius for connection distance

Figure 13 shows a group of agents with the radius representing the connection distance set to a small value. The resulting adjacency matrix shows that there is only one symmetric connection between agents 0 and 7 and all other agents remain unconnected. If this simulation were to be run, only agents 0 and 7 would move. As the radii are increased for all agents, as shown in Figure 14, the adjacency matrix reflects the increasing number of connections between agents. In this simulation, the adjacency matrix will always be symmetric when a disk proximity radius is used since the radii are the same for all agents.

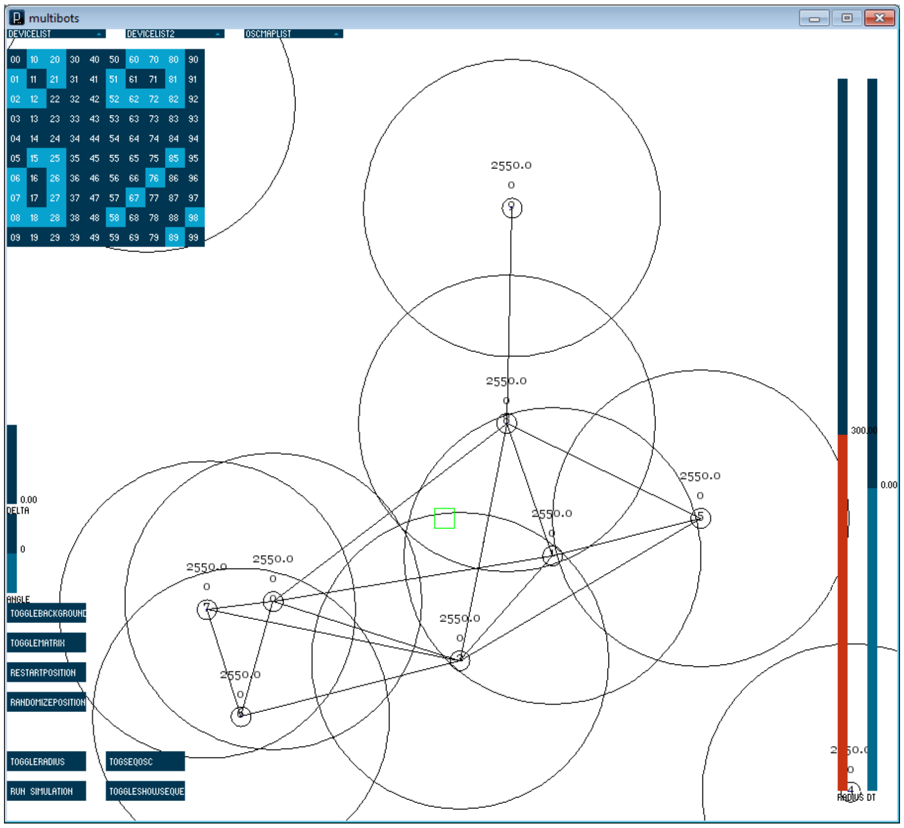


Figure 14: Network with larger radius for connection distance

Though this simulation presents an idealized view of ways to coordinate motion, there are certainly similarities with the boid model. If I consider the neighborhood as the area that a robot can see, this delta distance graph is what each agent uses to determine whether or not it should move closer together. Boid simulations require a threshold distance in order to determine whether or not neighboring agents should be considered in the overall flocking calculations. Now that I have described the motion capabilities of the simulation, I will describe a strategy for using motion parameters for music, and then explain musical design decisions.

A Strategy for Sonification

In an analysis of a serialist musical piece by Pierre Boulez, György Ligeti provided a method by which one could analyze and also compose such works [51]. First the creator of the work chooses what musical parameters he wishes to generate using a particular algorithmic process. Then the composer can alter other aspects that are not governed by the process to aesthetically edit output of the algorithm. Borrowing from Ligeti’s approach and taking into account real time musical generation and interaction, I can create similar three step process to apply to this simulation. The first is to choose a musical aesthetic, a means of creating sound that has editable musical parameters. The second step is to choose a mapping of parameters from the consensus equation to those of the musical aesthetic. Finally, those parameters that are not explicitly mapped to sound generation can be used in real-time to control the process and subsequently the musical output. I choose these parameters in such a way that they will describe the motion of the robots which will affect the music. The parameter mappings from consensus to music are as follows: the magnitude and angle of the velocity vector $\dot{x}_i$ as given by equation 9 are mapped to pitch, though not at the same time, and the degree of each agent is mapped to sequencer events. While I could have decided to map all three simultaneously, for example mapping the magnitude of the difference vector to another parameter like the amplitude of the sound being played, I could make an aesthetic decision not to modify the amplitude of the sound. I only wanted to focus on pitch and rhythm, not dynamics. The reason is there already exists some measure of dynamic range if one agent is playing alone versus having ten agents play altogether; the degree of each agent implicitly controls a measure of amplitude. The free parameters of minimum separation distance, rotation, and delta-disk graph distance can be modified in real time because they influence the difference vector as well as the degree, thus, affecting the movement behavior of the entire system of agents. The mapping choices are set using menu bars shown on the top screen in all of the figures.

Musical Design Features

Recalling the constraints established in the introduction, I will use sequencers as the main format for creating rhythmic music. In this simulation, each agent has its own 16 step sequencer, each one being tied to a common clock. One reason to use this number of steps is that it can easily make a common time musical signature with four beats, each having four steps each; many popular forms of music are in common time. The time interval between each step of the sequencer is 125 ms, which corresponds to 120 beats per minute. The reason for choosing this tempo is that I preserve the ability to play sound files. Using the Minim audio library in the Processing environment which uses JavaSound, this is the fastest rate at which my computer and other machines can play a sound file without latency issues [XMinim]. A straight, non syncopated rhythm would only have the first element of each of the four beats active. 16 steps give enough rhythmic possibilities for syncopated rhythms. Each agent also has one value for a MIDI note that is played each time one of the elements of the sequencer is filled in. Thus, the task I have is to determine how I am going to both set this note value as well as fill in the sequencer.

Exploring Pitch

I explore two different pitch sets. One uses a full range of MIDI note numbers, and the other takes the set of all pentatonic notes in the key of C Major. Choosing a pentatonic set ensures that the resulting melodies are harmonious, since the pentatonic scale does not have much dissonance. I can remove elements from both of the sets that are too low to hear or that are too high as per my preferences. Then I map the magnitude values to a MIDI note linearly from 0 to maximum, to the index of the pentatonic set, from the first (lowest note) to the last (highest note). When an agent moves from a fast speed to a slower speed as it converges to the centroid, the pitch will decrease from the highest index to the lowest until it stops moving. The limits of the simulation have to be chosen somewhat heuristically as some of the pitches are too low to be heard and some arbitrary magnitude has to be set to create a linear map. I could constrain the simulation to a maximum velocity to deal with this issue, but in this simulation there is no upper limit on speed. The mapping bound is chosen heuristically in this simulation. Imagine a fully connected network in which each agent rotates about the centroid. While the magnitude of the difference vector may be a good choice for a convergent set of robots, the same choice would not be as effective in a rotation scenario. The magnitude of the difference vector stays constant and thus there is no change in pitch in that case. Using the angle of the difference vector can completely utilize the entire pitch set since the values are bounded from $-\pi$ to π. While it works well for rotation, it is not necessarily the best choice in a scenario where the robots move towards the centroid taking a direct path. The more curved the pathway, the more varying the pitch choices will be; otherwise, the pitches will remain mostly dependent upon the initial starting conditions. The output from the linear mapping function *f*, used for both the magnitude and the angle is given by

$$f(x) = a + (b - a) * \frac{x - c}{d - c} \qquad (8),$$

where *x* is the input value, *a* and *b* are the min and max index of the output set respectively and *c* and *d* are the bounds of the input respectively. The result from *f* is truncated to an integer in order to select the index of the output set.

Exploring Rhythm

The last remaining parameter is the degree of the agent. When using the delta distance, the degree gives us a sense of the connectedness of the whole system of agents. Therefore, I wish to use this to describe rhythmic density. In particular I use the so-called Euclid algorithm since it provides a means by which I can create natural sounding rhythms using sequencers, where the number of hits per sequence corresponds to the degree of the agent. As described in [22], the Euclid algorithm can generate many metrical structures found across different cultures. The Euclid function takes two arguments: the number of hits, and the total number of elements in the sequence. I first represent a list of 1s and 0s, with 1 representing the number of hits, and the combined set of 1s and 0s representing the total number of elements. The goal of the algorithm is to distribute the 1s as evenly as possible, interspersing them with 0s. As a simple example, $Euclid(3,8)$ will start off with 3 1s, and 5 0s: 111, 00000. Each 0 is then assigned to each 1 until there are no more 0s remaining. Thus we have for the first step: 10, 10, 10, and remaining 00. Then these two remaining zeros are assigned to the first two sets: 100, 100, and 10. When the sets are recombined in this order the pattern is: 10010010. In this simulation, the second parameter is fixed at 16.

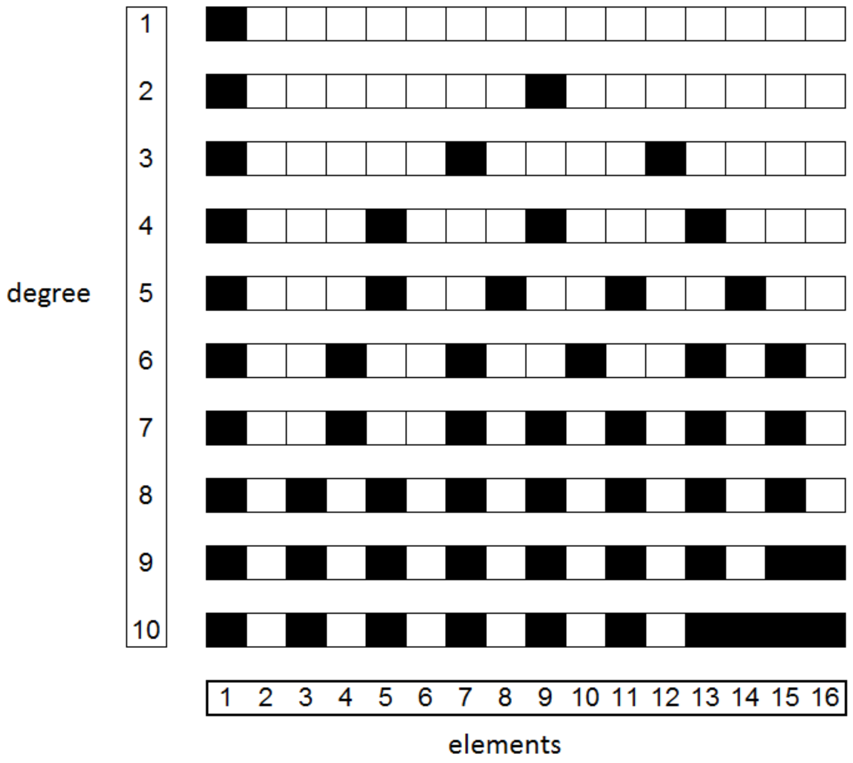


Figure 15: Possible Euclidean rhythms for ten agents

Figure 15 shows all the possible rhythm mappings for this system with degree as the first parameter to the Euclid function and 16 as the total number of elements. Assuming the same time interval between steps, 32 hits would certainly give more options for syncopation; however, it becomes problematic with regards to the Euclid function. Comparing $Euclid(x,16)$ with $Euclid(x,32)$, the time interval for low values of x becomes so long that it is hard to achieve a sense of rhythmic stability. This is another reason why 16 steps is a good choice for the sequencer. When small and odd values of x are used, the resulting rhythms feel off-kilter. For example, $Euclid(x,16)$ feels somewhat off-kilter even though the period of the sequence is not changing. This is because the three hits are not spaced with equal timings: $Euclid(x,32)$ is even more off-kilter because of the long duration.

Description of the Musical Output

The relationship between meter and degree is quite a versatile one. With a set of agents with varying degree, the meter becomes polyrhythmic, with those agents with a higher degree being more rhythmically "active". Those with no degree are silent. In a fully connected network structure, the rhythm for all agents is the same, which gives a sense of strict unity among the agents. The rhythm thus seems to reflect some characteristics of the adjacency matrix quite well. Although in this simulation I do not map angle and speed to rhythm, they prove to be quite interesting when mapping to pitch. In a situation where network agents are rotating, each agent’s orientation moves throughout all the radians of the unit circle. A linear mapping of pitch scale, when rotating in one direction, causes continually increasing pitches; conversely rotation in another direction would give continually decreasing pitches, an almost Shepard-scale like quality to the musical result. Speed mappings are most noticeable when stopping the motion of agents, causing them to suddenly change their direction, and also when they take curved pathways toward the centroid. Adjusting the separation distance parameter in particular is one way to cause the agents to increase or decrease in pitch. Videos of these simulations and the resulting music can be found online [52].

There are some important points to note about this multi-agent simulation. The purpose of the simulation is to use it as a tool to discover and explore the parameters of swarm motion; however, it is not meant to be a simulation of a realistic robot swarm. As stated earlier, even though I enforce a minimum distance constraint, this only applies to agents that are connected. Agents that cannot sense other agents would move through each other. Furthermore, I do not impose an upper bound on the speed limit of each agent; thus, the agents move at very unrealistic speeds and also omni-directionally. Finally, while I explore pitch mappings with this simulation, it would be very difficult to design a robot that could play all the possible MIDI note numbers under the constraint of acoustic sound. The linear mapping for the angle and speed to pitch values is fine in simulation, but it should instead be evaluated with rhythm. In the next chapter, I describe how I take the findings from the multi-agent simulation and apply them to a constrained boid model simulation that takes these points into account.

CHAPTER 6 DESIGN OF A ROBOTIC BOID SIMULATION

With the parameters of degree, orientation, and speed, along with other parameters for interacting, I can now design a boid-based swarm robot simulation program, taking into account the constraints of the real world, and attempt to make rhythmic swarm music. This simulation will be used in a study to evaluate the claims of my hypothesis regarding musical mappings and modes of interaction in musical swarm robot systems

Design Features

This second simulation is also created with the Processing framework, using Daniel Shiffman's implementation of the Craig Reynold's boid model as a starting point [53]. I modify the boids by ensuring stricter behaviors on collisions and also delineating an arena inside which the boids can move around. I also add other functions to allow for moving to specific targets as described by Reynolds, orbiting around specific locations, and a number of different interaction modes. The robots are viewed from a two dimensional top down perspective. They do not move omni-directionally, but rather holonomically as in Reynold's original boid simulation; there is one controllable degree of freedom. The robot is represented as a circle with an arrow indicating directionality of motion. Furthermore, the maximum speed and turning force of the robots are set to reasonable values assuming that the boids are about the size of small introductory robotics kits like the Parallax Boe Bot [54]. Each boid has a sequencer, as in the previous simulation, that is tied to a common clock. It is reasonable to assume that real robots are capable of synchronizing this way provided that they have some means of wireless communication with one another. The robot agents flash, giving a visual cue to go along with the sound; this is reasonable to assume if LEDs are used on the robots or if they use instruments like mallets that have some measure of anticipatory motion. Additionally, in order to help users of the simulation to localize the sounds emanating from the robots, each robot is panned depending on its position in the screen. The robots simulate the playing of small drums. I use various acoustic drum samples from Freesound.org [55]; different timbres are used to help differentiate which robots are playing. Samples are played using the Minim audio library for the Processing environment. Upon initialization of the program, each agent will play different types of drums with varying timbres in order to distinguish when two agents are playing at the same time. The decision to use different timbres was to avoid unnatural phasing effects and amplitude doubling that might occur if the same sample were to be used.

Musical Mappings

The degree and velocity vector succinctly describe the motion in the multi-agent control simulation.   Therefore in the robot boid simulation I use number of nearest neighbors, the orientation, and the speed. The number of neighbors is specified using a small radius around the robots, similar to Δ-Disk proximity graph in the multi-agent simulation. While this distance could be modified in a swarm robot system, it is difficult to visualize this boundary. Instead, I fix this neighborhood boundary to a diameter of 100 pixels around the agent. I choose this neighborhood boundary heuristically because this allows the robots to naturally cluster into small groups but also have the ability to break away from each other. These parameters are capable of being implemented on a real robotic system, both with and without an absolute positioning system. If an absolute position system is used to locate the robots, then the neighborhood is trivial to define. Such a system would not be used to map location parameters to music, but instead, would simply let the robot know when a neighbor has come into its view. If no absolute positioning system is used in the design of the robots, a single robot agent could still determine its neighborhood by using infrared or ultrasonic rangefinders, or with a local camera system.

The orientation of a robot could be determined with an absolute positioning system if there were markers on the robot indicating the front side. This requires some computer vision techniques if a top down camera system is used. Another way to obtain the orientation would be to use compass; this does not require an absolute positioning system at all. The speed of a robot could also be determined by an absolute positioning system merely by taking the derivative of position information. Another approach to determining speed would simply be to use the control system used to turn the motors; a more exact estimate of the speed of the robot could be gained with accelerometers or by using encoders on the wheels of the robot.

Since the design constraints establish that the robots use acoustic sound, realistically it might be difficult to allow for multiple pitched instruments on a small robot. Therefore, pitch is not considered in this simulation at all; instead, each robot agent only plays one sound sample. The sound samples are chosen to be percussive drum sounds because small drums are fairly easy to design with solenoids or small motors. .

While the previous multi-agent simulation only used the degree in making rhythmic mappings with the Euclid function, in this simulation I also make rhythmic sequencer mappings using the orientation and the speed; again I use the Euclidean function in the same way to fill in the 16-step sequencer. I evaluate these mappings separately instead of trying to combine them like I did in the previous simulation. All three of the mappings use the Euclid function with the second parameter of the Euclid function set to the number of sequencer steps. Parameters are mapped to the Euclid function described in the multi-agent simulation with one modification: the equation is $Euclid(1+x,16)$, where x is the input mapping in question. This is an aesthetic decision to ensure that there is no silence. While using neighbors is an easy task with the Euclid function because it is an integer value, angle and speed need to be bounded use the linear mapping function described in equation 8 in the previous chapter. The angle's input values range from $-\pi$ to π. The speed ranges from 0 to the maximum robot speed as defined by the simulation. The output values range from one to seven after the value is truncated to an integer. I choose one as the lower bound to ensure that there is no silence. Seven is chosen as the maximum range because in the boid simulation, the typical maximum number of neighbors a boid can have is around this number. One robot can only realistically have a set number of neighbors given the size of the 100-pixel neighborhood boundary. If I were to choose a maximum range that was too high or too low, then the mappings between neighbors and angles would result in different sets of Euclidean rhythms; it would not be useful to make comparisons between them.

Modes of Interaction

This robot swarm simulation implements five different modes of interaction, based on parameters found in the multi-agent simulation. Recalling the description of direct and indirect actions as described earlier in chapter 2, modes of direct human and swarm interaction ought to interfere with the movement dynamics of the robots. Modes of indirect interaction are those in which the robot swarm acts with more autonomy. Intermediate interactions are those in which the user interferes momentarily and then watches for the robots' reaction.

Intermediate Interaction: Pick up

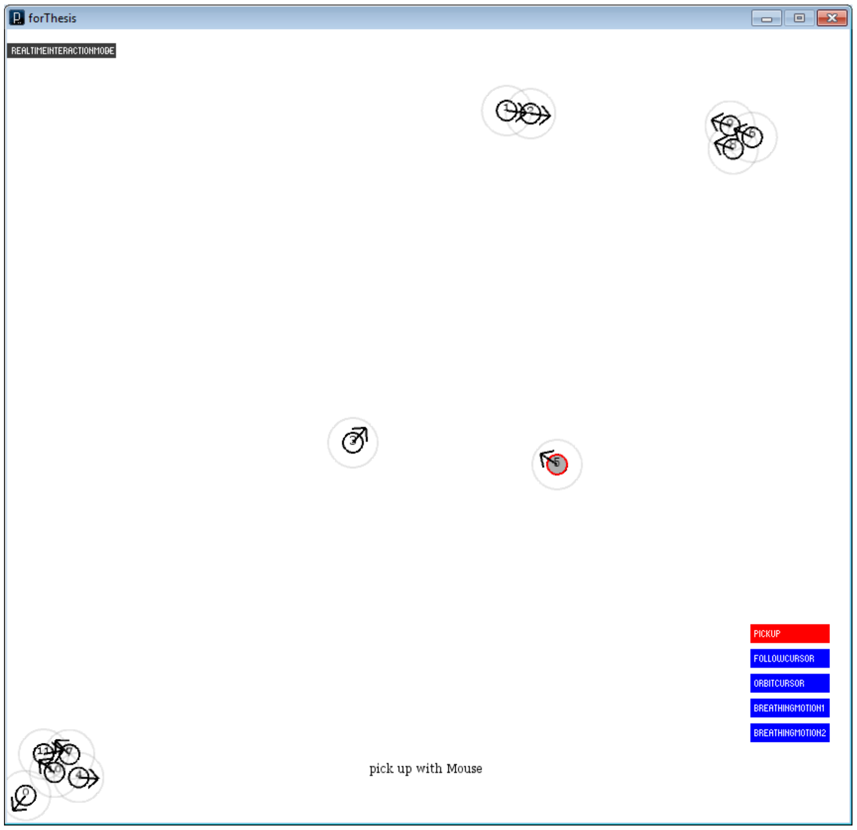


Figure 16: Boid simulation pickup mode 1

In the previously described multi-agent simulation, I can pick up the agents and move them to different starting locations. When consensus is selected, the resulting music sounds different depending on initial conditions. Additionally, if agents use a radial distance to set the connections between agents in the adjacency matrix, the agents stop moving when they are isolated from their fellow members. I could interfere with the dynamics of the system by providing some human input and then observe as the system compensates. Therefore in the robot simulation, the first mode, called “pickup”, allows for a user to pick up robots by clicking and holding the mouse and then moving them around to different positions while the mouse is pressed. When users let go of an individual robot by releasing the mouse, it will stop moving unless it comes within proximity to a neighbor; when that happens it will move again according to flocking parameters. In order to prevent the robots from escaping the arena, I assume that they can sense the boundary and will steer themselves away.   Figure 16 shows one agent highlighted in red that is being picked up by the user. It is also possible to grab multiple agents in succession and move them to different parts of the screen as shown in Figure 17. When released, because the agents are within proximity to one another they will immediately start to move according to the flocking rules. This interaction mode is a type of intermediate action since the user intervenes in the movement behavior of the robots, but then the robots will adjust and use their own flocking rules for movement.

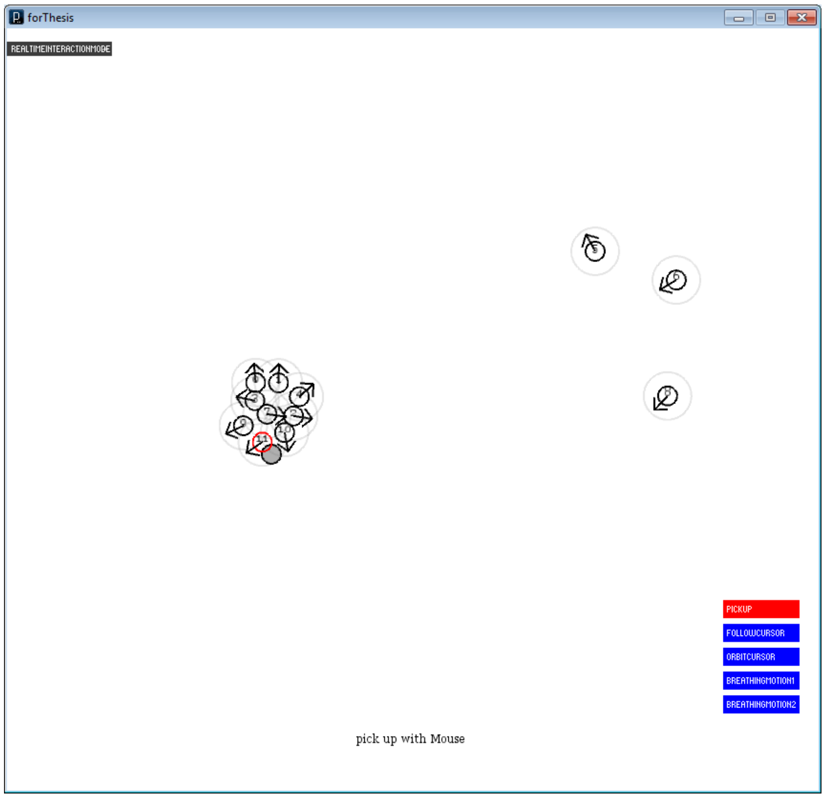


Figure 17: Boid simulation picking up several robots

Direct Interaction: Follow

In the multi-agent simulation, zeroing out the columns of the adjacency matrix would force the agents to follow the agent corresponding to the zeroed column. The user could drag one of the agents in real time, thus manipulating how all of the agents connected to it would move. Therefore, in the boid simulation, the second mode of interaction is called “follow”, making the boids follow the mouse. The boids will try to follow the mouse as best they can, grouping as they get near each other, still obeying their flocking rules of separation, alignment, and cohesion. This is a direct method of interaction since a user puts himself or herself as a target of interest for the robots. The boids attempt to maintain their motion dynamics in terms of their flocking behaviors. Since the user is actively overriding the robots movement with the mouse, the robots will often bump into each other.

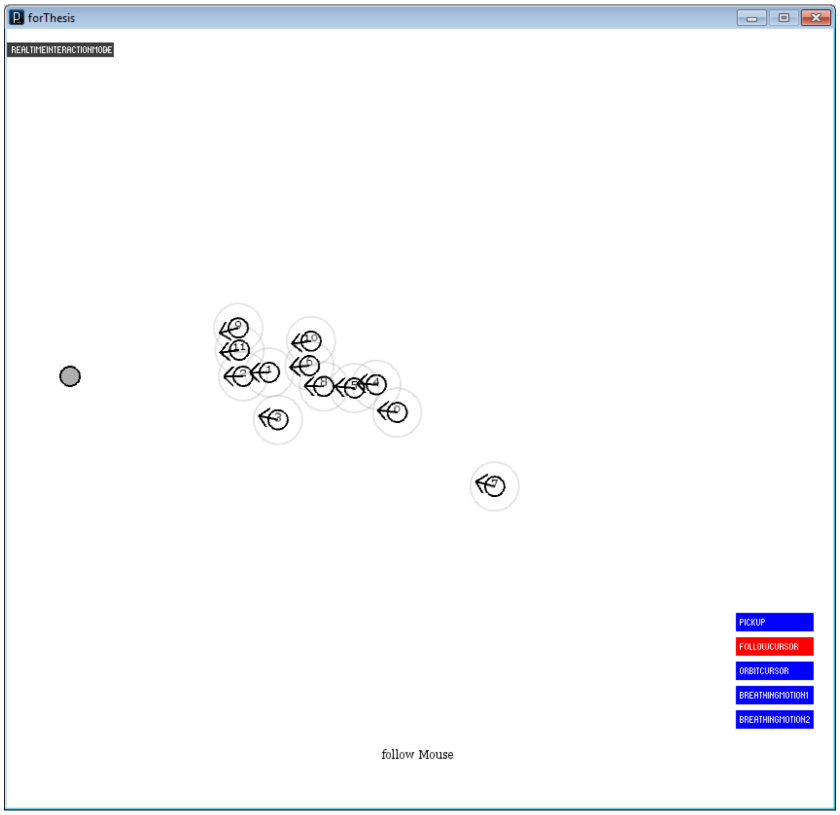


Figure 18: Boid simulation following mouse

Direct Interaction: Orbit

In the previous simulation, manipulating the minimum separation distance as well as the rotation parameter makes the agents rotate at a fixed radius around the centroid. Thus, in the robot simulation, “orbit” mode is created in which all agents orbit around the mouse at a fixed distance. I characterize this as a type of action as direct since the human user interferes with the robots motion. Comparing this interaction mode to “follow,” the user does not have to continually make a rotational gesture to cause rotational motion in the robots. . An example of this movement is shown in Figure 19.

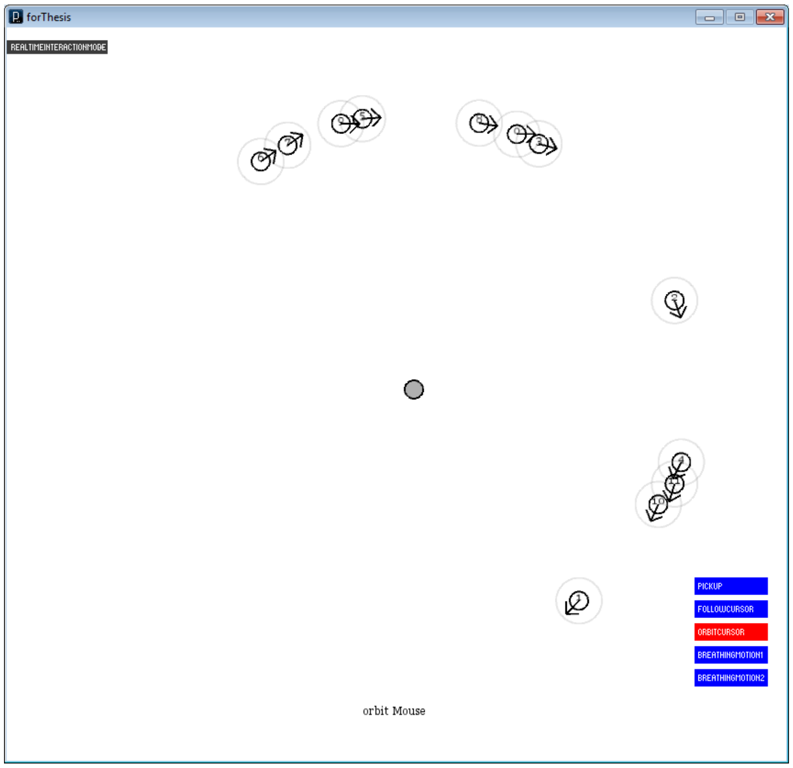


Figure 19: Boid simulation orbiting mouse

This particular movement also will have the same jostling as seen in the follow mode; robots will get closer to each other and sometimes bump into each other while trying to maintain a certain distance apart, obeying the rules of flocking. If the mouse is located near the boundary, the robots will still try to orbit as best they can, rubbing against the wall as they attempt to move around the mouse in a clockwise manner.

Indirect Interaction: Breathing Motion 1

In the multi-agent simulation, by manipulating the minimum separation distance between agents, moving it between zero and the max separation value, a user can make the agents converge and then separate repeatedly. This type of gesture has a profound impact upon the musical output. To make an analogous gesture in the robot simulation without requiring the use of sliders, I make use of timers. The robots move towards the center of the screen for a fixed number of seconds. An example of the first stage of this process is shown in Figure 20.

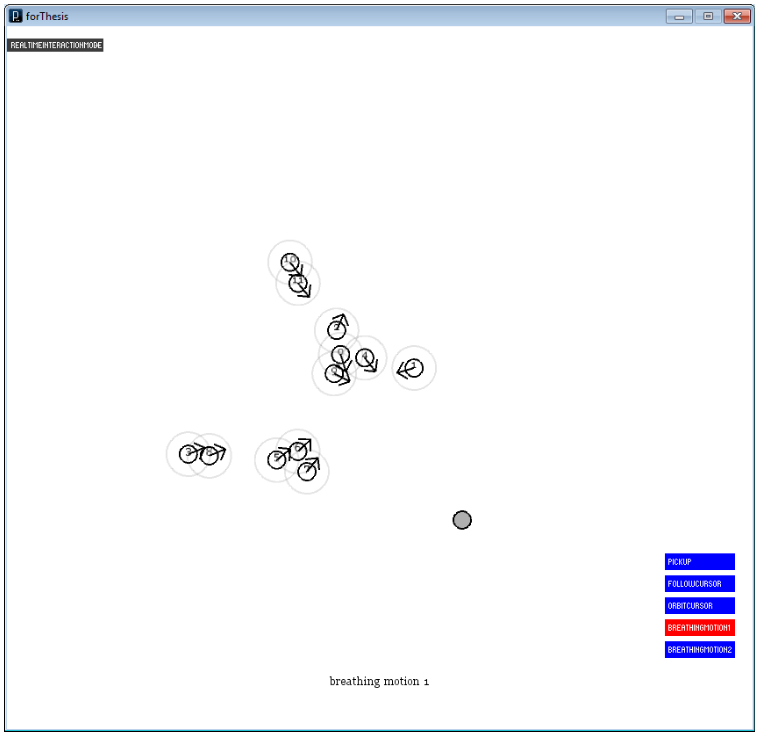


Figure 20: Breathing motion 1 phase 1 – gathering towards the center

After some amount of time, the robots attempt to move away from each other, maintaining some minimum distance from each other, while still attempting to move towards the center. The second stage of this interaction is shown in Figure 21 After another time interval, the minimum separation distance will reset back to the default value and this behavior of the robots will begin again. This results in a type of choreographed "breathing" motion; agents gather together and cluster in proximity and then spread apart. I use a predefined time as opposed to using a slider because I wanted to create a scenario in which the user would have less control than the previous interaction modes. This is a type of indirect type action because the user can only cause this behavior to occur and watch it unfold. No other form of interaction other than the selection of this mode is allowed

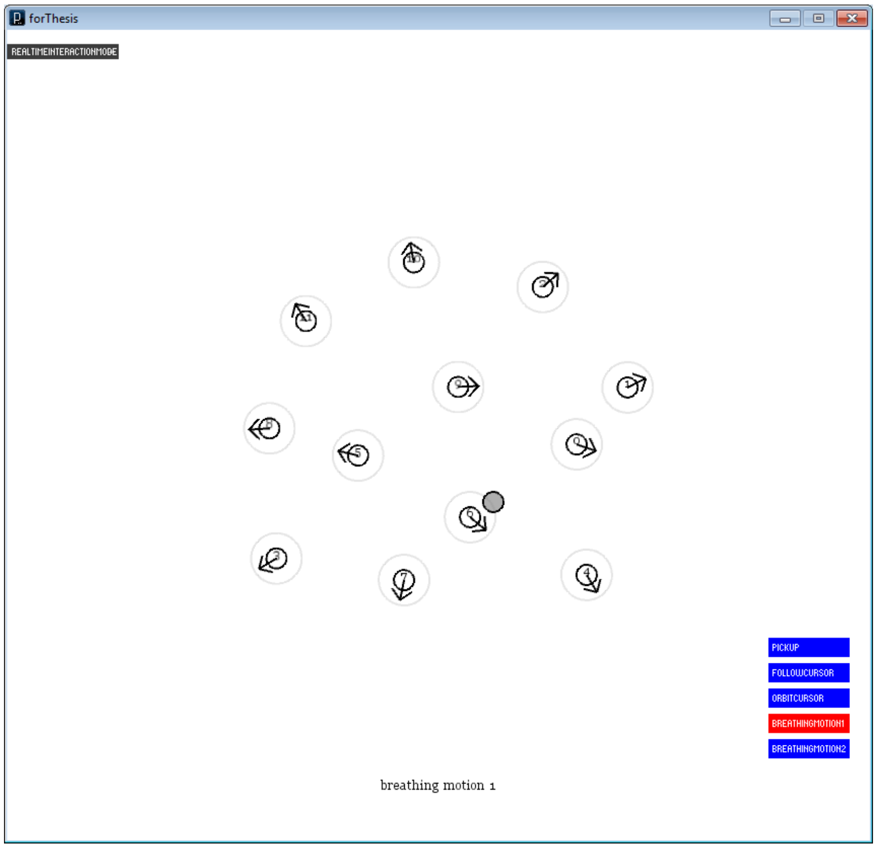


Figure 21: Breathing motion 1 phase 2 – separation

Indirect Interaction: Breathing Motion 2

The last interaction mode is a similar choreographed behavior that indirectly controls the behavior of robots except that this interaction mode incorporates rotational motion. Robots start off orbiting the center of a screen at a given distance as shown in Figure 22, again, still obeying the flocking rules just as they have been in all previous interaction modes. Over time however, the orbiting distance is shortened gradually until agents bump into each other. In the previous multi agent simulation I could do this type of behavior by modifying both the separation distance and the rotation parameter. Again, because I want to have an indirect means of controlling the robots, I decide not to use two sliders and instead use timers to indicate the different phases of this choreographed motion. Additionally, controlling two sliders might be more cumbersome and confusing. Figure 22 shows the robots orbiting outwards to their max distance. Then over time that distance is shortened, as shown in Figure 23.

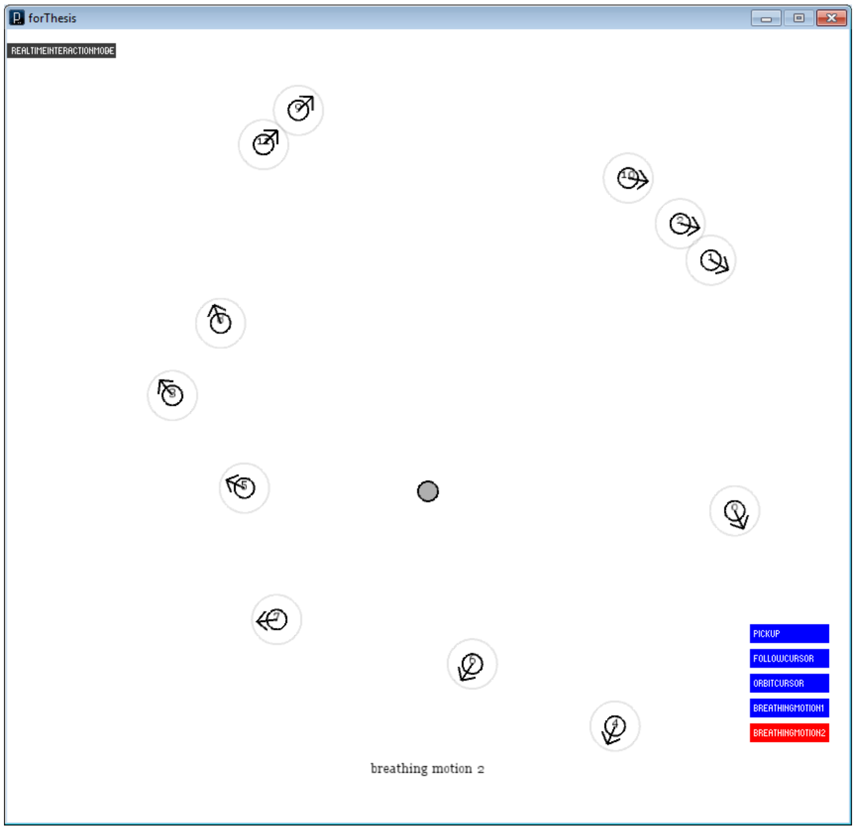


Figure 22: Breathing Motion 2 Phase 1 - Orbiting Towards Max Distance

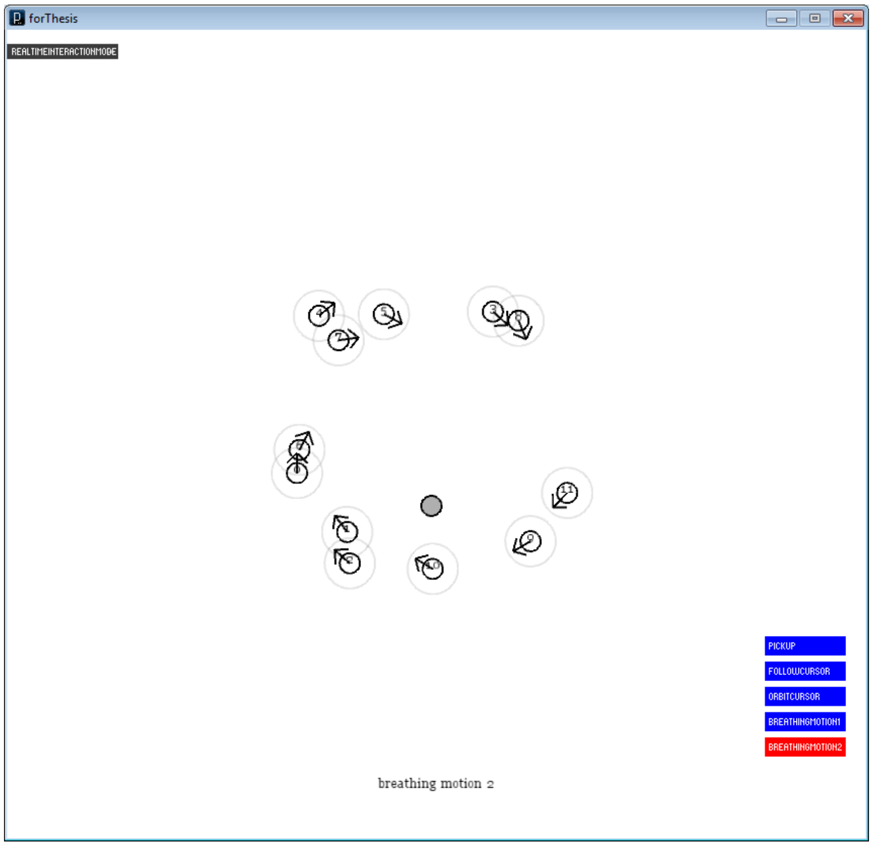


Figure 23: Breathing motion 2 phase 2 - decreasing radius

Just as in the first breathing motion, this behavior repeats; agents start out orbiting far away, spiral inward, and then spiral back out again.

With the swarm robot simulation complete, I am now ready to test the claims of my hypothesis regarding the legibility of motion to music mappings and the preferences with regards to interaction.

CHAPTER 7 USER STUDY

In an effort to evaluate the legibility of motion to musical mappings of the robot simulation, and to evaluate user preferences for modes of interaction, I design a two-scenario experiment involving a total of twelve music technology students, six students for each scenario. I select music technology students as the subjects because they typically have experience with computer music, computer programming, algorithmic composition, and interactive music; they represent a target group that is most likely to interact and compose with robotic musical swarms. None of the subjects involved in this study have any experience with swarming music. I obtain informed consent from all the subjects, which includes giving them a description their specific scenario.

Scenario 1: Time Trial for Legibility

The first scenario addresses the legibility of the three motion to music mappings of the robot boid simulation (angle, neighbors, and speed) through the use of a problem solving task. Additionally, this scenario addresses the interaction modes preferred by the subject in this context. The subject must determine the correct musical mapping as soon as possible.

Introduction Phase

After I obtain consent from the subject, I run the simulation program and demonstrate each of the five interaction modes, showing him or her how to manipulate swarm motion. After I explain all five modes, I ask the subject to sit in front of the computer and interact with the simulation program, using each of the modes to direct the robot swarm. Once the subject indicates that he or she understands how to use each of the five interaction modes, I show examples in the simulation program of the types of Euclidean rhythms that an individual member can play. I also show examples of the swarm robot agents playing combinations of those rhythms. In both the examples, the music heard by the subject is in no way mapped to the motion of the flock; I merely fill in the sequencers of the robots with the Euclidean rhythms. The purpose of this step is to ensure that the user is not surprised by the rhythmic music or the types of sounds the agents make when the trial actually begins, which could unduly influence the times taken to understand the first mapping.

Testing Phase

The subject must determine how the changes in rhythm are related to some individual parameter. Specifically I ask the subject: "what attribute of the individual member of the swarm, when changed, causes a change in the rhythmic behavior of that member." The subject is instructed to use the five interaction modes to help them answer this question as quickly as possible. I do not provide any example of possible answers; I only say that the answer will be based on some parameter common to all agents. To deal with problems of ambiguity, I accept a number of similar themed answers: direction and orientation, for the angle mapping, proximity for the neighbor mapping, and velocity for speed. If the subject guesses the answer correctly before using all the interaction modes, I give the subject the option of moving onto the next trial, or continuing to play with the simulation by using the remaining interaction modes. The computer records the video display output from the screen using the CamStudio 2.0 video capture program. The simulation records a log of the subject's interaction choices with associated timestamps to a text file. The subject will do three trials, one for each motion to music mapping. I configure the motion to music mapping by pressing a key. This indicates the start of the trial. After the three trials are completed, I give the subject three questions to answer, as shown in Table 1.

Table 1: Questions asked After Completion

q1	Which mapping best describes the motion of the flock?
q2	Which mapping did you prefer the most for any reason?
q3	Which interaction mode did you prefer the most for any reason?

Expected Outcomes

I ask the first question to determine the subject's opinion on which of the three motion to music mappings results in the best description of swarm motion. I ask the second question to determine the subject's opinion about the musical mapping. If the answers to the first and second questions are mappings that are more legible, meaning the mappings that took less amount of time to answer, then I can claim that under the condition of a problem solving task with simulated swarm robots, legibility is important to user preferences. If the answers to the first and second question are the same, then this could mean that typical users who interact with swarm robots in a problem solving context are more likely to prefer musical mappings that best describe the motion of the flock. I ask the third question to see what types of interaction modes are preferred by the users in this problem solving scenario. Since the task is to determine the motion to mapping as quickly as possible, it would seem reasonable that subjects choose the particular modes of interaction that helped them determine the mapping. It will be useful to compare the answer to the third question to the mode in which users spent the most time.

Scenario 2: Free-form Interaction

The second scenario addresses the level of interest among the motion to music mappings in a context where there is no time limit and the mappings are explained beforehand. Additionally, this scenario addresses the interaction modes preferred by the users in such a context. I use the time spent in each interaction mode and mapping as an indicator of subject preferences.

Introduction Phase

After consent is obtained, I run the simulation program and demonstrate each of the five interaction modes to the subject, showing him or her how to manipulate swarm motion, just as in scenario one. After I explain each mode, I ask the subject to interact with the simulation program, using each of the modes to direct the robot swarm. Once the subject indicates that he or she understands the five interaction modes, I show examples in the simulation program of the types of Euclidean rhythms that an individual member can play and also examples of the swarm robot agents playing combinations of those rhythms. Additionally, I also explain the three motion to music mappings. I explain to the subject how to change the mappings by selecting the 1, 2, and 3, keys corresponding to angle, neighbor, and speed respectively, but I do not let the subject set them until the start of the interaction phase. Once the user indicates that he or she understands all the options available, the interaction phase begins.

Interaction Phase

I ask the subject to use the simulation as long as he or she desires. The only requirement for the subject is that he or she uses each of the motion to music mappings at least once and each of the interaction methods at least once; thus there are 15 combinations at least. The subject is permitted to revisit different mappings and interaction modes. The trial ends after the subject interacts with at least all 15 combinations and indicates that he or she wishes to stop. The choices of mappings and interaction modes are logged by the simulation program. Video of the simulation is recorded as well. Afterward, the subject is asked the same three questions given in Table 1. I use the timestamps of all the actions taken by the users to determine the time spent in each mapping and interaction mode.

Expected outcomes

While the previous scenario addresses the legibility of the mappings and the time spent in each interaction mode, this scenario does not include the stress of a problem solving task. In this scenario, I discern the level of interest of the motion to music mappings and the interaction modes, using the time spent in each as a measure of interest.

I ask the first question to determine the subject's opinion on which of the three motion to music mappings results in the best description of swarm motion. I ask the second question to determine the subject's opinion about the musical mapping. If the answers to the first and second questions are mappings that are the more legible mappings as determined from scenario 1, then I can claim that under the condition of an open-ended interaction context with simulated swarm robots where the mappings are known, legibility is important to user preferences. If the answers to the first and second question are the same, then this could mean that typical users who interact with swarm robots in a free form interaction context are more likely to prefer musical mappings that best describe the motion of the flock. The third question is being asked to see what types of interaction modes, direct or indirect, is preferred by the subject in this context. It will be useful to compare the answer to the third question to the mode in which the subject spent the most time also to compare the answers for all questions to those of scenario one.

CHAPTER 8 RESULTS AND ANALYSIS

The following are the results and analysis of the user study as described in the previous chapter.

Scenario 1 Results

Table 2: Time in minutes for scenario 1 trials

	angle	neighbor	speed
sum	28.05845	13.0779167	41.0740667
average	4.67640833	2.17965278	6.84567778
var	10.7711038	2.84313407	14.17158
std	3.28193599	1.68615956	3.76451591

Table 1 shows statistics regarding the time taken to guess the mappings correctly. The neighbor mapping took the least amount of time to guess correctly; it can be assumed that this is the most legible mapping; speed is therefore the least understandable.

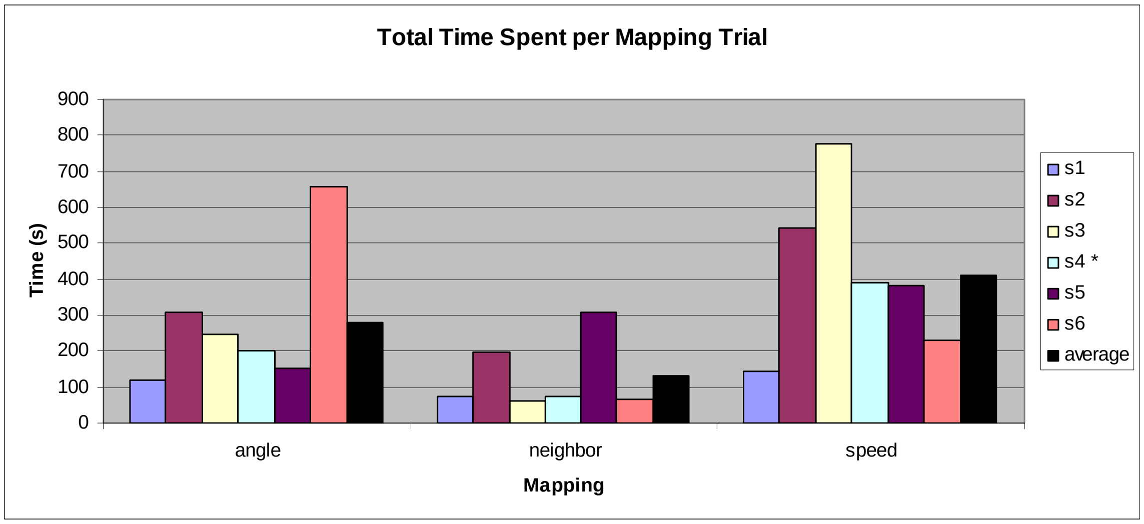


Figure 24: Total time spent per mapping trial

Figure 24 shows the total time in seconds spent in each mapping trial among all six subjects. One subject, marked with an asterisk, could not determine the speed mapping after going through all of the interaction modes, so the time was notated at which point the subject chose to end the trial. The neighbor mapping is the most understandable for five out of the six participants. Five out of six subjects took the longest time to guess the speed mapping correctly.

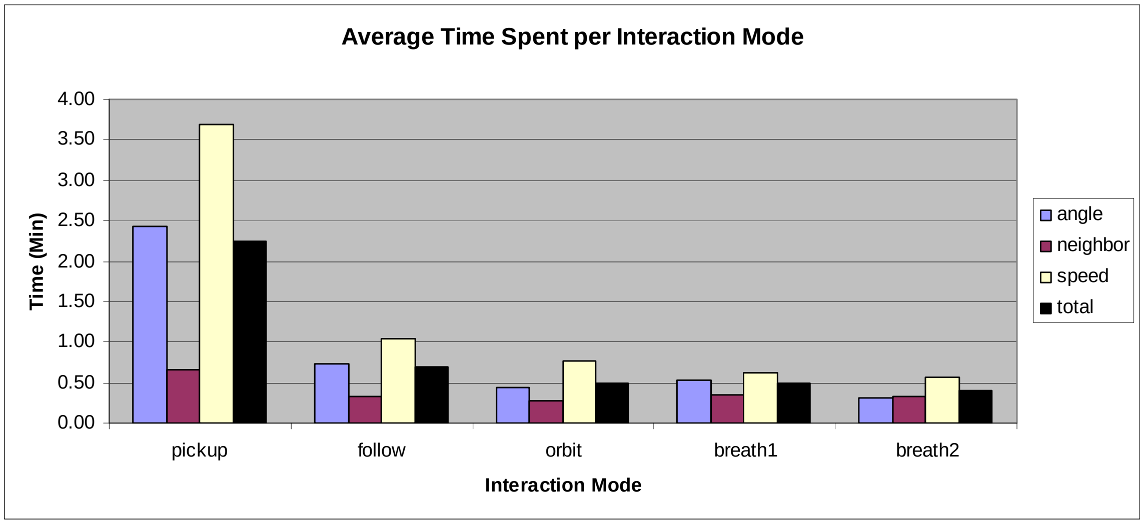


Figure 25: Average time spent per interaction mode

Figure 25 shows the average time spent per interaction mode in each of the three mappings for all six subjects. Again what we can see from this graph is that the neighbor mapping in general is the most legible, whereas the speed mapping takes much longer to determine. Interestingly, all users spent the most time in pickup mode. It is tempting to think that this data might give some indication of user preferences of interaction modes; however, time in this problem solving task scenario can only reliably used as a measure of understanding the legibility of the mappings and not as a measure of effectiveness of the interaction to determine legibility. For example, orbiting, follow, and breathing motion 2 can all be used to determine angle mapping; the modes are biased depending on what the user selects

Behavioral Findings

When the mapping is less understandable, subjects will try out the different modes, transitioning back and forth between them. Table 4 shows the total number of interaction mode transitions made for each mapping. By transitions, I mean how many times the user switched the modes of interacting with the swarm.

Table 3: Total Number of Transitions - Scenario 1

	angle	neighbor	speed
transitions	40	22	55

For mappings that are easier to understand, users do need to make as many . Conversely, users switch back and forth between modes in order to try and determine more difficult mappings. The total transitions are higher for the speed mapping, which is to be expected since this is the least legible mapping.

<u>Questions</u>

Table 3 lists the answers to the questions given by each of the six subjects at the end of Scenario 1.

Table 4: Questions Scenario 1

	q1	q2	q3
s1	angle	angle	pickup
s2	neighbors	speed	breath2
s3	neighbors	neighbors	follow
s4	angle	angle	follow
s5	angle	angle	pickup
s6	angle	neighbors	breath1

When asked which of the motion to music mappings best described the motion of the flock, four subjects select angle and two choose neighbor mappings. Speed is never chosen, perhaps because it is the most difficult to determine. For the second question, when asked which of the mappings the subjects preferred for any reason, three subjects prefer the angle mapping and two choose neighbor. Only one subject prefers the speed mapping. For four out of the six trials, the mapping that best described the motion of the flock is the same as the one the subjects prefer. This might make sense given the context; getting the correct answer could simply lead to more satisfaction. When asked which of the interaction modes they preferred in the third question, two subjects prefer the pickup mode, two prefer follow, and the remaining two choose the two breathing motions. From this we might suppose that users will prefer modes of direct and intermediate interaction for a problem solving task

Possible Sources of Error.

The angle mapping is always presented first for all six subjects. It is possible that at the start of the trial, the subjects might not know what types of answers to put forward until they guess the answer correctly. Therefore, the angle mapping in actuality could have a lesser amount of time to guess the score correctly. The order of the motion to music mappings should have been randomized to account for this. Another potential problem with the time trial is that while the subjects are instructed to answer as quickly as possible, with no penalty for incorrect guesses, some subjects seem to be more apprehensive and deliberately make fewer guesses; they do not forward any answer until they are sure they are correct. One last problem concerns the collision behavior. Oftentimes subjects put forward collision as a possible mapping. When boids collide, the speed and orientation suddenly change; since the neighborhood radius of the boid is small, it might seem that collision is technically the correct answer for all three mappings. Collision is technically a correct way of describing what the subjects observe, even if the question asked is about a specific attribute or parameter of individual agents. Thus this could have caused confusion which might unduly influence the time taken to guess the mapping. To remedy for this in future iterations, I could tell them beforehand that collision is not one of the mappings. Another option would be to increase the separation behavior, though this would result in a change in overall flocking behavior.

Scenario 2 Results

While the first scenario involved a time trial to determine the legibility of the musical mappings, the second scenario is a free-form interaction context.

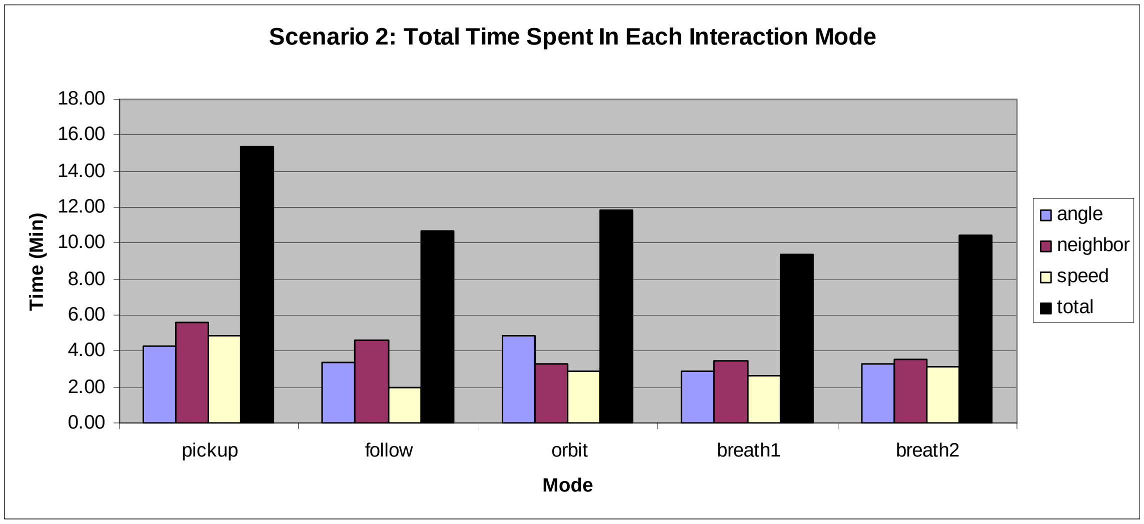


Figure 26: Total time spent in each interaction mode

As Figure 26 illustrates, when participants are allowed to interact without the stress of a problem solving task, subjects still spend the most time in the pickup mode cumulatively, then orbit, follow, breathing motion 2 and finally breathing motion 1. However, in the case of the angle mapping, users spend more time in orbiting, 4.83 minutes, compared to 4.25 minutes spent in pickup. There is not much disparity between the modes of direct and indirect interaction. Users still prefer to spend the the most time in the intermediate interaction mode

Users spend still more time in angle and neighbor mappings than in speed. Table 5 shows that users spend almost equal time in angle and neighbor mappings. The neighbor and angle mappings are the most legible according to the results of the first scenario.

Table 5: Total Time Spent in Mappings

	Angle	neighbor	speed
Time (Min)	21.32	21.70	17.10

However, Figure 27 shows how each of the subjects spends the time for all musical mapping.

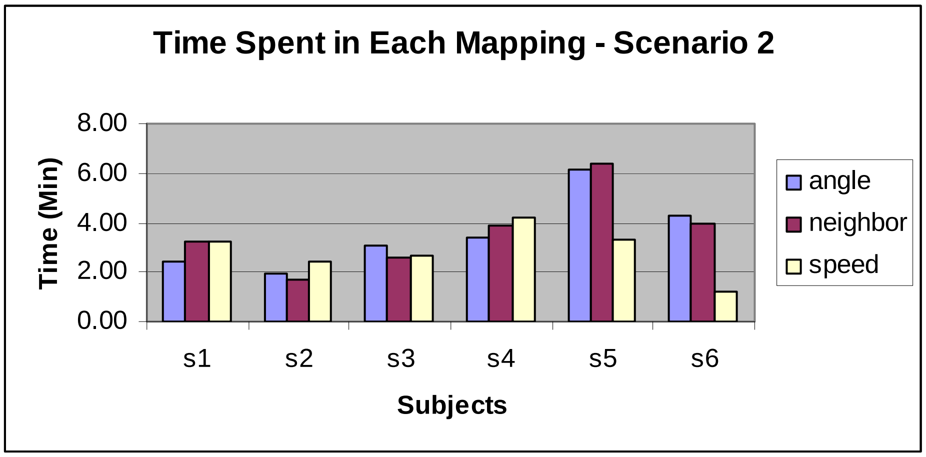


Figure 27: Time spent in each mapping - scenario 2

Although the speed mapping is the least understandable as the time trial experiment showed, in a free form context, some users choose to spend more time in speed mappings. What this could mean is that understandable mappings are not necessarily the ones that are preferred. This would seem to be in line with Hsu's personal aesthetic to avoid mappings that seem straightforward.

Behavioral Findings

One important behavioral finding in the second scenario is how the subjects utilize the different musical mappings. In the first trial, the mappings are set before hand and the users must determine the mappings. In the second scenario, the subjects have the option of selecting the mapping. Some subjects switch back and forth between angle mappings while staying in one interaction mode. The subjects do not interfere with the motion of the swarm robots, but they do affect their behavior. This is most certainly a type of human robot interaction. While they give up control over the motion of the robots, they take control over the musical mappings, and thus, affect the musical behavior of the robots immediately. Therefore while I can assume that intermediate and indirect modes of motion interaction are favored in a free form interaction context, direct forms of interaction with respect to musical mappings might be favored in such a scenario. A way to account for this discrepancy would be to design a new partially free form interaction context scenarioin which the subject is restricted only to one musical mapping per trial, but is allowed to experiment with all the modes of interaction without time constraint.

Questions

Table 6: Questions Scenario 2

	q1	**q2**	**q3**
s1	neighbors	speed	breath2
s2	Angle	angle	orbit
s3	Speed	speed	breath1
s4	Angle	neighbors	breath1
s5	neighbors	neighbors	breath1
s6	Angle	speed	pickup

Looking at the responses to questions 1 and 2 as seen in Table 4, for three out of the six answers, the mapping that best described the motion was the one that was preferred. Half of the users also chose speed as their preferred mapping, which is the least legible according to the time trial. This again supports the idea that legible mappings are not always going to be the most preferred by users who interact with a robotic swarm if they know what the mappings are beforehand and if they are not tasked with trying to understand them. Looking at the answers to the third question, users tend to prefer indirect modes of interaction. These modes give less control over the robots in this open ended context. What we can say now is that is that given a situation where users can more freely interact with a robot swarm, users are fine with giving up some measure of influence over the robots motion, and might prefer to let the agents themselves dictate the changes in rhythm.

Applying the Results

Using the information from this study, I now have some insight about how users might behave with a real swarm robot system. My first hypothesis is incorrect: parameters that are more legible in a problem solving context are not preferred more in an open ended context. My second hypothesis is partially correct: modes of intermediate and direct interaction are preferred in a problem solving context. In an open-ended context, however, users prefer indirect modes even if they spend more time in direct and intermediate modes.

If people were to interact with the robots in an exhibition setting with no explanation given about the robots' motion behaviors then it might be prudent to use neighbor and angle mappings along with a means for picking up the robots or directing them to specific locations. However, if the people who interact with the robots have some advance knowledge of the robots' operations, then the legibility of the mapping might not have any effect on preferences. These users might prefer interacting with the swarm in choreographed ways to let the motion behaviors of the swarm dictate the music without human interference with swarm motion. However, control should be given over the selection of musical mappings in this case.

In order to design a system that can be picked up and moved around by the users, there are some issues to consider if the swarm robots are being tracked with an overhead camera. If users move under the overhead camera system, the computer vision algorithms used to track motion should be robust enough to handle interference from humans and should be able to recover the positions of the swarm robots quickly. If following is going to be used, the vision system needs to be able to track the position of humans in some way as well. If the robots do not require an overhead camera system, then picking up the robots is not a problem, so long as the robots can find out their orientation with respect to the other robots.   The choreographed breathing motions and orbital gestures would benefit immensely from an overhead tracking system. The orbiting interaction around the mouse seems to be the least preferred mapping according to the results from the user study. Frustrations could lie in the fact that it took too long for the robots to move given the radius. Keeping this in mind, if the swarm robots rotated about a point without the use of a camera system, then these types of rotational motions would occur more slowly. As the multi-agent control simulation illustrates, agents move to their targets quicker if they have more information about the relative distances to their neighbors and provided that graph describing the interconnectedness is appropriate. It is a good decision in general to design first using an absolute positioning system like a top down camera to obtain relative distance information without having to deal with the problems associated with local range finding methods such as occlusions of the targets; if the choreographed motions are deemed satisfactory in this setting, then later on robots can be designed to deal with only local range-finding methods.

CHAPTER 9 CONCLUSIONS AND FUTURE WORK

The thesis aims to study the ways in which humans might interact with musical swarm robots. A musical swarm robotic system could provide new ways to explore algorithmic compositions as well as novel modes of human robotic interaction. The research is focused on rhythmic music with sequencers, human interaction, acoustic sound, and the constraint that motion of the swarm robots causes the changes in music. The contribution of this thesis towards this goal is two-fold. First, I design an interactive simulation that uses multi-agent control to sonify the motion of agents. This simulation adheres to the constraints of rhythmic music with sequencers, and human interaction, and provides a methodical way in which to map motion to sound as well as to elicit control over the motion. The second contribution is in the design of a boid-based robot swarm simulation program that adds the additional constraint of acoustic sound, while synthesizing interaction modes on an axis of direct, intermediate, and indirect control. Finally, I evaluate the swarm robot simulation using a user study with music technology students in order to assess whether the legibility of the motion parameters as determined in a problem solving task will be preferred in an open ended context. I attempt to determine which modes of interaction are preferred in these same contexts.

The results indicate that legibility may be useful in problem solving contexts with a swarm; however, when given advanced knowledge of the musical mappings, legibility is not as relevant in an open ended interaction.. With these results in hand, I can now design musical swarm robots for use in these two kinds of contexts, knowing which mappings and which modes of interaction might be preferred for each scenario.

In the future, more user studies with the simulation could prove useful, especially among students who are either not in music technology, or do not have as much experience with such simulations. New modes of interaction have already been developed in the swarm robot simulation that takes a stigmergic approach, affecting both musical mappings as well as the motion of agents. I must consider the type of objects to be placed in the environment carefully, both in a simulation context and in the physical design of the system. Admittedly, a top down approach perspective in the swarm robot simulation is not how people would interact with a real robotic swarm. An even more realistic simulation would require a virtual 3D interaction environment that also includes ambisonic sound to simulate a more accurate depiction of being among musical swarm robots. In future work, this level of design could be useful.

In parallel, I am developing robot hardware using mostly inexpensive hobbyist parts. For example, a striker is created using small motors and used to strike against a hand-bell which is mounted on the robot. Inspiration for parts and design come from a number of sources, including the Yellow Drum Machine [56]. I use a top down camera system to track the positions of robots, but only the parameters of neighbors, angle and speed will be used to modify the robot's sequencers. Currently, by using Android phones as the brains of the robots and using Arduino-based ADK boards [57]. I plan to design swarm singletons with behaviors as described here in this thesis. This platform can be used to create new and interesting ways to explore human and swarm robot interaction, musical robotics, multi-agent control, and swarming and interactive music.

REFERENCES

[1] F. Mondada, et al., "The Cooperation of Swarm-bots: Physical Interactions in Collective Robotics," Robotics & Automation Magazine, IEEE, vol.12, no.2, pp. 21-28, June 2005.

[2] M. Turduev et al., "Cooperative chemical concentration map building using Decentralized Asynchronous Particle Swarm Optimization based search by mobile robots," Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on , vol., no., pp.4175-4180, 18-22 Oct. 2010

[3] H. Lee and K. Sim, "Distributed moving algorithm of swarm robots to enclose an invader," Control, Automation and Systems, 2008. ICCAS 2008. International Conference on , vol., no., pp.729-733, 14-17 Oct. 2008.

[4] D. Dorhout, Prospero Robotic Farmer, online, available: http://www.robotliving.com/diy-robot/prospero-robotic-farmer, 2011.

[5] A Takanishi et al , "Development of an anthropomorphic flutist robot WF-3RII," Intel Robots and Sy.s '96, IROS 96, Proc. of the 1996 IEEE/RSJ International Conf. on, vol.1, no., pp.37-43, 4-8 Nov 1996.

[6] E. Van-Burskirk, “Robot Band Backs Pat Metheny on Orchestrion Tour,” online, available: http://www.wired.com/underwire/2010/01/orchestrion/, 2011.

[7] Y. Kim et al, "Enabling humanoid musical interaction and performance," Collaboration Technologies and Systems (CTS), 2011 International Conference on , vol., no., pp.212-215, 23-27 May 2011.

[8] G. Weinberg and S. Driscoll.. “Toward Robotic Musicianship”. Computer Music Journal 30, 4, pp. 28-45, 2006.

[9] G. Hoffman and G. Weinberg, “Shimon: An Interactive Improvisational Robotic Marimba Player,” In Proc. of 28th of the Int. Conf Extended Abstracts on Human Factors in Comp. Sys. (CHI EA '10), 2010.

[10] A. Albin, S. Lee and P. Chordia, "Visual Anticipation Aids in Synchronization Tasks," Abstract In Proc. of the 2007 Society for Music Perception and Cognition (SMPC), 2011.

[11] G. Hoffman, G. Weinberg, “Interactive Improvisation with a Robotic Marimba Player,” Autonomous Robots, pp 1-21, May, 2011.

[12] T. Blackwell, “Swarming and music,” Evolutionary. Computer. Music, pp. 194–217, 2007.

[13] C. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” In Proceedings of the 14th annual conference on Computer graphics and interactive techniques (SIGGRAPH '87), ACM, New York, NY, USA, 25-34, 1987.

[14] J. Alonso-Mora et al., "Multi-robot system for artistic pattern formation," Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp.4512-4517, 9-13 May 2011.

[15] M. Egerstedt and M. Mesbahi, Graph Theoretic Methods in Multiagent Networks, Princeton University Press, New Jersey, 2010.

[16]. T. Davis and O. Karamanlis, “Gestural control of sonic swarms: Composing with grouped sound objects” Proceedings of the 4th Sound and Music Computing Conference , 11-13, 2007.

[17] D. Bisig, et al, “Interactive Swarm Orchestra - A Generic Programming Environment for Swarm Based Computer Music. Proceedings of the International Computer Music Conference. Belfast, Ireland, 2008.

[18] Browning, Cameron, “Phlock,” Chelsea Art Museum, May 2008, Online, available: http://5cameron.com/dt/?q=node/84, 2011.

[19] T. Machover, “Instruments, Interactivity, and Inevitability,” Proc. of the 2002 Conference on New Instruments for Musical Expression (NIME-02), Dublin, Ireland, May 24-26, 2002.

[20] D. Kirsh, and P. Maglio, “On Distinguishing Epistemic from Pragmatic Actions,” Cognitive Science, (1995).

[21] B. Fry, and C. Reas, “Processing,” 2004, online, available at: http://www.processing.org, 2011.

[22] G. Toussaint, “The Euclidean algorithm generates traditional musical rhythms,” Proceedings of BRIDGES: Mathematical Connections in Art, Music, and Science, Banff, Alberta, Canada, July 31 to August 3, pp. 47-56, 2005.

[23] E. Sahin and W. M. Spears, Swarm Robotics Workshop:State-of-the-art Survey. Erol S¸ ahin and William Spears, editors, Lecture Notes in Computer Science 3342, Springer-Verlag, 2005.

[24] E. Bonabeau, "Editor's Introduction: Stigmergy." Special issue of Artificial Life on Stigmergy. Volume 5, Issue 2 / p.95-96, 1999.

[25] J. Gancet et al, "User interfaces for human robot interactions with a swarm of robots in support to firefighters," Robotics and Automation (ICRA), 2010 IEEE International Conference on , vol., no., pp.2846-2851, 3-7 May, 2010.

[26] Z. Kira and M. Potter, "Exerting human control over decentralized robot swarms," Autonomous Robots and Agents, 2009. ICRA 2009. 4th International Conference on, vol., no., pp.566-571, 10-12 Feb. 2009.

[27] Y. Uozumi, “A Musical Framework with Swarming Robots.” CMMR-LNCS. vol. 4969. 360-‘67, 2008.

[28] J. McLurkin, “Stupid robot tricks: A Behavior-based Distributed Algorithm Library for Programming Swarms of Robots,” M.S. thesis, Dept. Elect. Eng. Comp. Sci., MIT, Cambridge, MA, 2004.

[29] J. McLurkin, personal webpage, online, available at: http://people.csail.mit.edu/jamesm/swarm.php, 2011.

[30] A. Kapur, “A History of Robotic Musical Instruments,” Proceedings of the 2005 International ComputerMusic Conference, San Francisco, California: International Computer Music Association, pp. 21–28, 2005.

[31] J. Boyd, “This Robot Toots Its Own Flute,” IEEE Spectrum Inside Technology, July 2008, online available: http://spectrum.ieee.org/robotics/humanoids/this-robot-toots-its-own-flute/0, 2011.

[32] K. Petersen, J. Solis, A. Takanishi, "Implementation of a musical performance interaction system for the Waseda Flutist Robot: Combining visual and acoustic sensor input based on sequential Bayesian filtering," Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on , vol., no., pp.2283-2288, 18-22 Oct. 2010.

[33] T. Jurek, Review of Pat Metheny: Orchestrion, online, available: http://www.allmusic.com/album/r1700629/review, Nov, 2011.

[34] P. Metheny, "About Orchestrion," online, available: http://www.patmetheny.com/orchestrioninfo/, Nov, 2011.

[35] Rudy's Corner, Review of Pat Metheny: Orchestrion online, available: http://www.rudyscorner.com/678/review-pat-metheny-orchestrion.html, Nov, 2011.

[36] T. Doyle, "Pat Metheny's Orchestrion: Ochestrion Manoeuvres" online, available: http://www.soundonsound.com/sos/apr10/articles/orchestrion.htm, Nov, 2011.

[37] G. Weinberg, G. Hoffman, R. Nikolaidis, and R. Aimi. Shimon + ZOOZbeat: an improvising robot musician you can jam with. In ACM SIGGRAPH ASIA 2009 Art Gallery & Emerging Technologies: Adaptation (SIGGRAPH ASIA '09). ACM, New York, NY, USA, 84-84, 2009.

[38] M. Gardner, "Mathematical Games: The fantastic combinations of John Conway's new solitaire game 'life'," Scientific American, vol: 223, issue: 4, pp: 120-123, 1970.

[39] Wolfram Research Labs, "Wolfram Tones: An experiment in a new kind of music." online, available: http://tones.wolfram.com, 2011.

[40] E. Miranda. "Granular Synthesis of Sounds by Means of a Cellular Automaton." Leonardo Music Journal, vol 28, No. 4, 1995.

[41] E. Miranda, "Cellular Automata Music: From Sound Synthesis to Musical Forms," Evolutionary Computer Music, 170–193, 2007.

[42] S. Papert, "Teaching Children to be Mathematicians vs. Teaching Them About Mathematics," International Journal of Mathematical Education and Sciences, V3, pp. 249-262, 1972.

[43] The Lion King, Disney, 1994.

[44] T. Blackwell, "Swarm Music: Improvised Music with Multi-Swarms," M.Sc Thesis, University College London, 2003.

[45] T. Blackwell and P. Bentley "Improvised music with swarms," Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on, vol.2, no., pp.1462-1467, 2002.

[46] W. Hsu, "On Movement, Structure and Abstraction in Generative Audiovisual Improvisation," Proceedings of the International Conference on New Interfaces for Musical Expression, Oslo, Norway, 2011.

[47] T. Unemi, D. Bisig, “Playing Music by Conducting BOID Agents– a Style of Interaction in the Life with A-Life,” In Proceedings of the Ninth International Conference on Artificial Life IX, 546 – 550, Boston, USA, 2004.

[48] D. Bisig and M. Neukom, “Swarm Based Computer Music - Towards a Repertory of Strategies,” Proceedings of the Generative Art Conference. Milano, Italy, 2008.

[49] MacDonald, E., “Multi-robot assignment and formation control”: ECE, GaTech, Atl. GA, 2011.

[50] ORACLE, “Java Sound API: Soundbanks,” online, available: http://java.sun.com/products/java-media/sound/soundbanks.html, 2011.

[51] G. Ligeti, “Pierre Boulez: Decision and Automatism in Structure Ia.” Die Reihe 4 (“Young Composers”): 36–62, 1960.

[52] A. Albin, “Multi-Agent Simulation,” online, available: http://vimeo.com/28592794 2011.

[53] D. Shiffman, “Autonomous Steering Behaviors,” online, available: http://www.shiffman.net/teaching/nature/steering/, 2011.

[54] The Boe Bot Robot, Parallax Inc, online, available http://www.parallax.com/go/boebot, 2011.

[55] Freesound.org, http://www.freesound.org.

[56] F. Lyneborg, “Yellow Drum Machine,” Let’s Make Robots, online, available: http://letsmakerobots.com/node/112, 2011.

[57] Google, “Android Open Accessory Development Kit” online, available: http://developer.android.com/guide/topics/usb/adk.html, 2011.