Communication via Sound with Bird-Like Autonomous Agents

Mir Jeffres
Georgia Institute of Technology
Atlanta, Georgia, USA

Mason Mann
Georgia Institute of Technology
Atlanta, Georgia, USA
mmann33@gatech.edu

ACM Reference Format:
Mir Jeffres and Mason Mann. 2021. Communication via Sound with Bird-Like Autonomous Agents. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 CONCEPT

The concept of our project was to create a group of autonomous, self-contained physical objects that could sing to each other and hear each other singing and respond according to computational algorithms. The idea was that simple behavior could be programmed into each agent and that together, complex group behavior would arise. Agents would switch between a state where they are listening, and another state where they respond. During the listening state they would determine the loudest note that they heard and use that as the input to a first order Markov model. With all the agents separately playing single notes from the same Markov model, the idea was that the whole group behavior would be similar to that of a single entity playing a single Markov-model. We considered this to be a sort of "distributed Markov model." Additionally the average loudness that each agent is hearing at any given time will have an effect on the way it plays the notes. When the average loudness is higher, they will shorten their listening period, playing notes more often and moving through the markov model more quickly, and they will also play shorter notes creating a frantic feel. Additionally there would be a few ways that a user could interact and guide the behavior. The primary method of interaction is simply picking up the agents and moving them, if they are further apart from each other they will not hear each other as loudly, decreasing the average loudness and when far enough away, making some agents not interact at all. Another way that the users interact is through a special agent called the "Leader Bird." This agent plays a special command pitch when a button on it is pressed. All of the follower agents have four separate Markov models built on different pitch-class sets and they all start on the same one. When an agent hears this command pitch they will move to the next pitch-class set. Since they might not all hear the command pitch depending on where they are placed, they will likely begin to drift into groups playing different pitch-class sets moving from a state of consonance in the beginning of the piece to a polytonal cloud as time goes on.

Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 CONTEXT

The work that most directly inspired this project was Felix Hess's Electronic Sound Creatures. In his project, one hundred physical autonomous agents were created to perform in smaller groups. These agents had wheels, motors, microphones and speakers and were able to listen to each other, respond and perform crude localizations moving towards and away from each other depending on the notes heard. They were implemented in a fully analog manner using band-pass filters and op-amp integrators to detect pitches, which made them very computationally limited. Our project extends this concept in a few ways. First, it is implemented with an embedded microcontroller, allowing more pitches to be detected as well as allowing more complex rules to guide each agent's response to them. Additionally, while Hess's work was fully generative, we wanted to incorporate a way for audiences to interact with our project. Rather than giving the agents wheels and allowing them to move themselves, they are stationary and must be moved by the audience. Picking them up and moving them varying distances from each other will allow them to hear each other more or less which factors into the algorithm guiding their behavior. Another concept that informed this project was IOT implementations of Data-Over-Sound. The idea that complex data can be sent between agents through the physical medium of sound rather than more complex protocols such as bluetooth or network connection is very interesting. The final concept that inspired this was Craig Reynold's Boids algorithm, for modeling bird flocking behavior. It was a landmark work in complex behaviors arising from autonomous agents with simple rules guiding behavior so we thought it would be interesting to explore similar concepts in the sound domain.

3 IMPLEMENTATION

The project was implemented by creating nine autonomous agents housed in 3.5"x3.5"x5" wooden boxes. Each one contains a PCB containing a speaker, a microphone, an ATMEGA328 microcontroller, a DAC chip, and circuitry for scaling the microphone signal, driving the speaker, conditioning power and conditioning a few GPIO pins on the microcontroller to read switch data and drive LEDs. A schematic of the PCB is included in the .zip along with the project code. An NJM2073 dual power amplifier chip was used in a push/pull configuration to allow for maximum power efficiency in driving the speaker. The DAC chip used was a MCP4901 8-bit DAC chosen for its price, simplicity and ability to be driven over I2C. ATMEGA328 chips were chosen because of their cost and ease of programming through the Arduino IDE and their extensive documentation. In the end they proved to be a poor choice for audio uses. Large sacrifices had to be made in sampling rate, buffer size and wavetable size and still almost all of the memory was used. To code the ATMEGA328s, code was written in the Arduino IDE and flashed to the chip through an Arduino Uno, the board that allows for user interfacing with ATMEGA328 chips. The pitch detection section of the code used a Goertzel algorithm. This algorithm can be thought of as a single-bin FFT. We didn't have the computation to spare to run an entire FFT and since we were only looking for a finite number of discrete pitches it made sense to only look for those pitches. The bin size is a function of both the sample rate and the buffer size. Since we didn't have the memory to spare for a large buffer we kept a short buffer and dropped the sample rate incredibly low to 2500Hz. This allowed us a resolution of 10Hz per bin which was small enough that adjacent notes didn't overlap. We weren't playing any frequencies over 1000Hz so the low sample rate shouldn't really affect the sound of the system. During the listen period the microcontroller constantly samples the magnitude of all the notes it is listening for and keeps track of the maximum over that period. Then at the end of the period it takes the maximum value of this maximum array, essentially finding what the loudest note heard over the entire listening period was and uses that note to feed the markov model. The average of this maximum array is also used to scale the length of each listening period as well as the length of an ASR envelope on each note. Resulting in a higher density of shorter notes when the agent hears a lot of sound. Within the code, a Markov chain was implemented to determine the next note from the previous note the agent "heard" through the microphone. The Markov implementation uses the basic idea of a Markov chain that is taught in linear algebra where there is a transition matrix that has all rows and columns summing to 1 and contains the percent chance of each transition between notes happening, multiplied by the current state (in this case, a column matrix which contains one 1 value for the note it heard, and the rest zeros). In this project, there were five total notes, so the transition matrix was a 5x5 matrix and the current state matrix was 5x1. To determine the next note from the outputted 5x1 matrix of probabilities of the next note, an array was created by multiplying each output chance from the 5x1 matrix by 100 to give integer values that summed to 100 for the column, and then duplicating each possible note in another array n times, where n was the row of the matrix * 100. This gave a 100 point array with each possible note, and a random number between 0 and 99 was used to index that array to get the next note out. This algorithm seems complicated, but was relatively easy to implement and allows for easier manipulation of the Markov model than using a Markov library. The only library used in this implementation was one to perform the matrix multiplication to avoid writing two more loops.

4 EVALUATION

Musically, this piece is not particularly successful in its current state. It is an interesting concept and behavior has begun to arise but it would be hard to describe the sonic result as aesthetically beautiful. Some combination of low sample rate, small wavetables, imperfect power amplifier and imperfect speaker mounting distorted what should have been perfect sine waves into something closer to a distorted sawtooth. The fundamental frequencies are preserved so the agents are able to communicate with each other but the sonic result is not as pleasant to listen to. The main things learned from the difficulties in the project were about developing and iterating when building physical objects. We often were slowed down by certain parts of the project not working when we could have worked on them more in parallel. We didn't begin implementing the Markov behavior until the units worked, which took a while as there was a lot of soldering, laser cutting and a few iterations of circuit boards. Looking back, this behavior could have been developed in something like Max and then just ported over once the physical objects were functional. Debugging behavior on microcontrollers that have no serial port and need to be removed from the board to be programmed proved to be very tedious. While we were very excited to be outside of the Max environment and creating something real that didn't need a laptop to run, the power of Max and similar tools as a method of rapid prototyping of behavior is something that we could have benefited from in the early stages of the project. Another issue that we ran into was the quality of the PCBs used. They were milled on campus using a desktop CNC machine. This allowed us to quickly create iterations but the boards didn't have solder mask, making them difficult to solder to and causing quite a few shorts. This increased the amount of time spent debugging hardware by quite a bit and in the end we only had 4 out of the 9 units totally working given time constraints. The final issue with the system was that instead of running on battery power they currently are running on 9v DC wall wart plugs. In general the interactivity of the system is not quite where we would like it to be yet. The user is able to pick up the agents and move them around but not very far as they are still running on wired power. The thresholds and scaling based on how loud the notes each unit hears are could use some fine-tuning as well to help the user really feel that they are having an effect on the music. The pitch-class set switching from the leader bird does work however, allowing the user a piece of very direct control. Overall we view this project not as a complete piece but as a stepping stone. The Markov model works great and the units are all able to speak to each other but a lot of kinks in the system need to be worked out and the user interactivity absolutely needs to be explored a little more deeply.

5 REFERENCES

[1] Arm Ltd. Why Data-Over-Sound Should Be a Part of Any IoT Engineer's Toolbox. Arm | The Architecture for the Digital World. Retrieved December 7, 2021 from https://www.arm.com/resources/white-paper/data-over-sound

[2] Craig Reynolds. Boids (Flocks, Herds, and Schools: a Distributed Behavioral Model). Retrieved December 7, 2021 from https://www.red3d.com/cwr/boids/

[3] 2000. Electronic Sound Creatures. In Prerational Intelligence: Adaptive Behavior and Intelligent Systems Without Symbols and Logic. Springer, 452–457. DOI:https://doi.org/10.1007/978-94-010-0870-$9_6$2