Electronic Musical Instrument Design
Spring 2018
Anna E. Bartlett

Conversations at the Kitchen Table
Anna Bartlett, Ben Papp, Spencer Perry, Jake Zaslav

1. Introduction

Conversations at the Kitchen Table is an interactive musical exploration designed for multiple players. It is designed to be an artpiece as well as a musical experience, and it is built to be playable by any person with any skill level. *Conversations* is a hexagonal table mounted on Lazy Susan bearings, so it can rotate smoothly. 16 sensors are mounted on the playing surface, which each produce their own individual set of sounds. The exciting part of the instrument, however, lies in its combinatorial power: when different sensors are played simultaneously, the instrument enters different states, or sets of sounds, all of which can be discovered over time by the players.

Fig. 1. Surface of Conversations at the Kitchen Table, *featuring collaborative artwork and multicolored LEDs.*

2. Evolution and Architecture

This instrument evolved from our previous project, *A Table is Just a Wall Lying Down,* in which we laid the groundwork for the design of *Conversations.* The *Table* project was a great success, consisting of a rectangular playing surface with the same set of sensors: ten FSRs and six softpot sliders. We implemented much of the logic for the musical states in *Table* that we used and modified for *Conversations,* and we also maintained several of the sound states throughout both versions.

The major changes to *Table* were the physical design and the improvement of the instrument's playability. Instead of the flat rectangle of *Table, Conversations* is a rotating hexagon, which adds allure to the overall appearance. We also made a number of cosmetic changes to the design: placing the FSRs in a more ergonomic arrangement; covering the playing surface in vinyl, so the players do not directly touch the sensors; and adding a top layer of transparent acrylic, for a more polished appearance. The acrylic is covered by collaborative artwork by several Tufts artists (and ourselves). Additionally, the surface of *Conversations* features visual feedback in the form of eight differently colored LEDs, which reflect the current musical state of the instrument.

As for the improvement of playability, one large change we implemented was a looping feature. One of our biggest takeaways from *Table* was that it seemed to only produce discreet sounds, and lacked coherence between musical ideas. Our solution was to add two pairs of switches to the design, which control a looping feature in the Max patch, allowing the players to create layers of sounds and rhythms, rather than disjointed, individual noises.

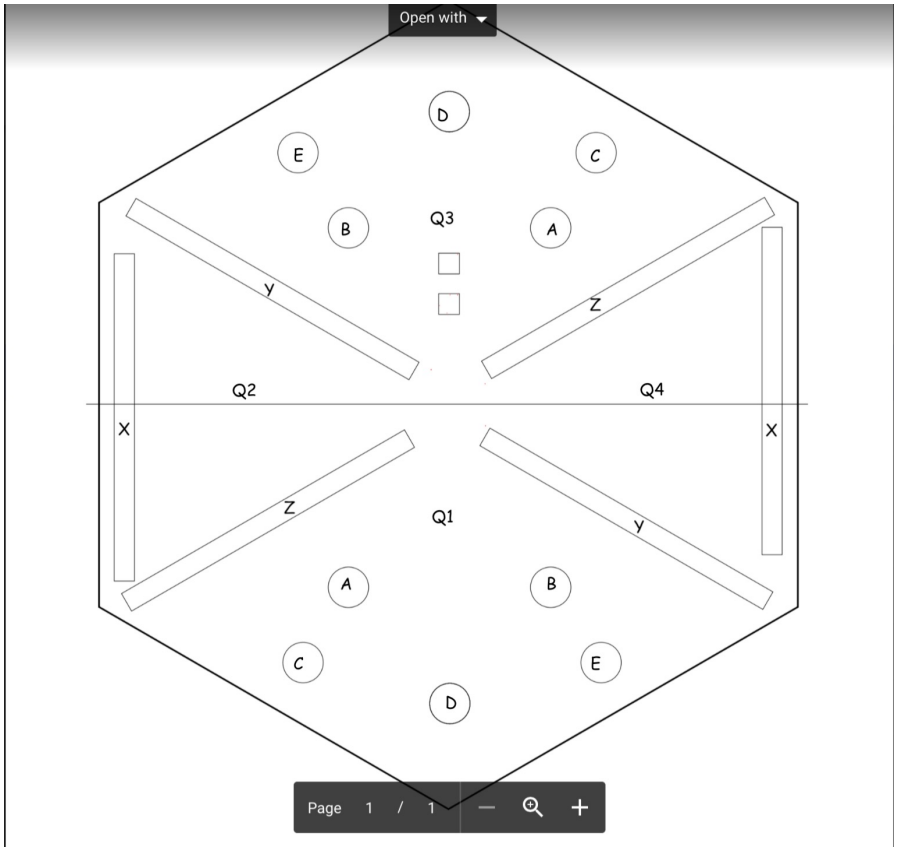

Fig. 2. Diagram of layout of sensors. A-E are FSRs; X-Z are softpots.

3. Soundscape

Our goal for the soundscape was twofold: to maintain a theme of "conversation," since the idea behind our instrument was one of collaboration and co-evolution; and to produce a performance that is clearly musical. The soundscape for *Conversations at the Kitchen Table* consists of a large set of Reason synthesizers, which range from "classic" subtractors, to field recording samples, to human chorus sounds, to animal noises.

We kept some of the states from *Table,* such as the Yodel samples, the set of four-note sequences, the Kong drum kit, and the recordings of our group saying vowel sounds together.

We modified other states from *Table,* such as the animal recordings (which were manipulated in Garage Band in a similar fashion to our Yodels), the field recordings (one of which we maintained, and some of which we replaced with new samples), and the chords (which still play the same notes as they did in *Table,* but play them in the sound of a human chorus instead. Other states we decided to design anew. These include the set of movie recordings, for which we spliced quotes from popular movies into individual words, and which the player can trigger using the softpots; and the intervals, which are now much more dynamic than in *Table,* and slide between intervals.

4. Hardware

The physical construction of *Conversations* was more involved than that of *Table,* since this iteration features moving parts. The design for mounting the playing surface on the Lazy Susan bearings took some careful planning, since the entire playing setup needed to be able to rotate on its own axis. This meant all the wiring, the breadboard, and the Arduino had to rotate together. We solved this problem by mounting the hexagonal surface on four wooden dowels, which connect to the Lazy Susan, which enables the entire system to turn on itself. The Lazy Susan bearings are themselves mounted on a hexagonal base, which has the same dimensions as the playing surface.

The playing surface contains six softpots, ten FSRs, four buttons, and eight LEDs, all of which are soldered to a single breadboard, which is mounted underneath the playing surface. The breadboard also has outgoing connections for each sensor and for the buttons to the Arduino, which is mounted beside the breadboard. The hot wires for the LEDs connect directly to the

Arduino, without passing through the breadboard. The sixteen touch sensors are connected to the analog pins of the Arduino. The buttons and the LEDs are connected to the digital input/output pins of the Arduino.

The final sensor which we use is a rotary encoder, which is mounted between the base and the top surface of the Lazy Susan. We mounted the top of the encoder to the same surface as breadboard and Arduino, so that the encoder could easily be wired to the breadboard and the digital pins of the Arduino. The shaft of the encode is sunk into a hole in the base, which is filled with hot glue to secure the shaft.

The hardware is connected to the computer via a USB cable. This is the only part of the construction that does not turn on itself; that is, the cable wraps around the dowels connecting the playing surface to the Lazy Susan, which limits the amount that the playing surface can rotate. In a future iteration *Conversations* would be a wireless instrument, so as to maximize the potential of the rotary encoder, and to make the physical turning easier.

5. Software

Most of the processing is contained within the Max patch. We re-used large portions of the original patch from *Table,* as the logic for each of the states remained the same between both projects. We tweaked some of the states to function in a better, more musical, and more exciting manner. For instance, we made sure that in this iteration, none of the FSRs acted as switches, which had been the case in *Table.* We made it a priority to take advantage of the range of data that FSRs can generate.

Major changes occurred in the implementation of the loop. This took a lot of puzzling out and splicing together different tools in Max. The result is a subpatch that reads from the binary buttons. One button triggers recording, and one mutes and unmutes the loop (there are two pairs of buttons, so two loops are building simultaneously). The "record" button starts storing MIDI data in the seq object. When it is done collecting we trigger a sequence of events that plays back the stored sequence, and sends a bang to play it back again. The "mute" button stops the seq object from producing sound, but the stored sequence is still being looped through. This means that when the looper is "unmuted," it will pick up exactly where it would have been if it had been playing out loud. A further layer of complexity is added by the rotary encoder. The encoder is able to distinguish between clockwise and counter-clockwise rotation. We use this data to control the playback speed for the seq object. This occurs through the use of a tick message, which steps through the stored sequence at a given speed. To make the looping even more exciting (and to add to the confusing nature of the instrument), the encoder has opposite effects on the two loops: as one plays back faster, the other one slows down.

A final piece of software for *Conversations* is the implementation of the logic for triggering the LEDs. Originally, we attempted to control the LED states (that is, control the Arduino output) directly from the Max patch, which apparently is possible with specific Max patches. However, none of the documentation we found actually enabled us to communicate directly between Max patch and Arduino output, so we devised a different solution. The Max patch sends codes to the Arduino controller via the serial port, bitpacking eight values which correspond to the eight LEDs. The Arduino controller unpacks these values, and, based on them, determines which LEDs (which digital pins) should be outputting.

6. Conclusion

Conversations at the Kitchen Table turned out to be an exciting and challenging project. We ran into a few issues at surprising points in the process, such as struggles with the X-Carve, difficulties with various types of rotary encoders, confusing LED logic, and occasional issues with playback of the loops. Moving forward, the instrument would probably have wireless capability, and we would polish the appearance a bit more. The looping capabilities could also be more involved: during our research, we were considering implementing the loops using the mtr object in Max, which could enable multiple sequences to be looped and controlled by the same switch. Overall, however, we are very excited about the result of *Conversations at the Kitchen Table.* We achieved our goal of a playable, fun, exciting musical instrument, which also involved multiple people and has the ability to produce cohesive musical sounds.