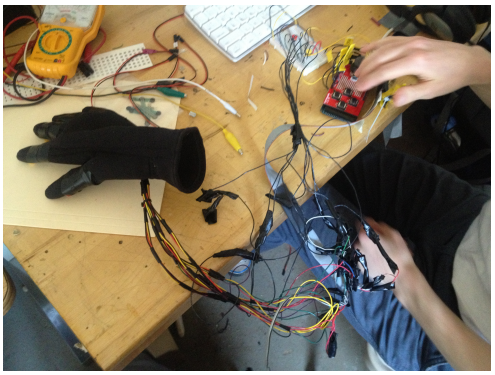
Liam Cassidy, Saul Sparber, Drew Zeiba

EMID MUS 66/Lehrman

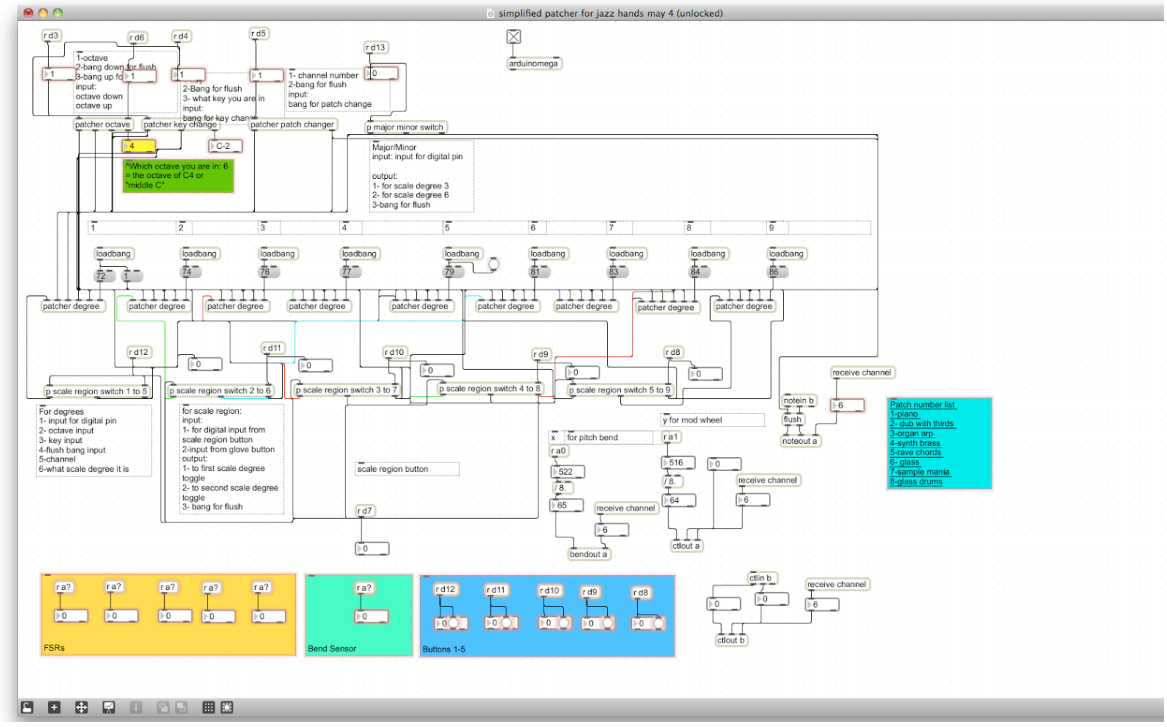
Jazz Hand(s) Mark II

The Concept:

We wanted an instrument that was immediately playable and intuitive, but still offered a degree of control, something that anyone —musician or otherwise— could make sound good.



We also wanted an instrument with a lot of performability, insomuch that it would show to an audience the effort of the performer in a one-to-one relation with the sounds being generated. We thus set out to design a musical glove. We intended for the



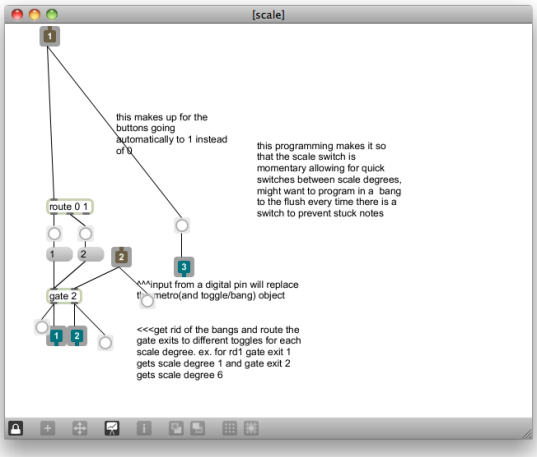
fingertips to play scalar notes with velocity, and we wanted the wrists to control properties like modulation or pitch bend. We also had hopes of incorporating LEDs to add to the excitement, as well as an additional hand controller for even more options. This instrument would be easy to pick up and play naturally, but offer a wide range of control and not confine itself to any sound set or type. Playing can be best compared to a piano that can only play a fixed scale (making the rhythm and patch choice all the more important), however this is largely because of programming design choices as this instrument is highly adaptable.

After our first version we realized the impracticability of two hands. We have since added a game-controller style board with buttons and a joystick to add additional control that is mounted on the Arduino directly.

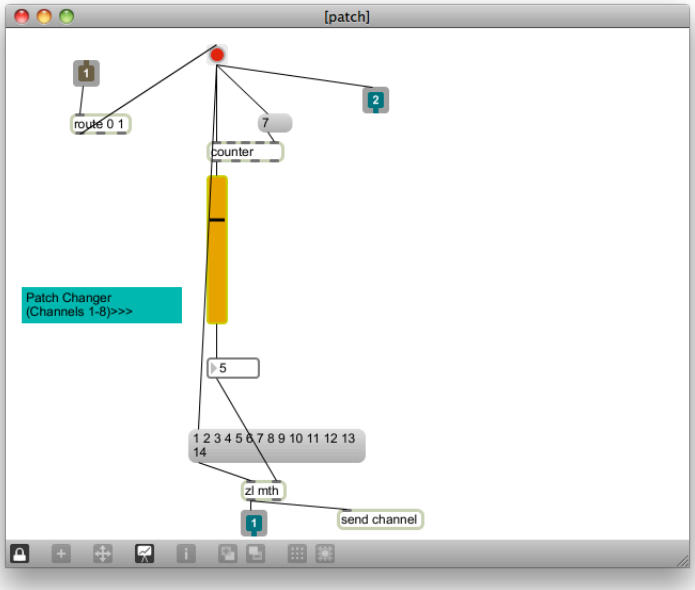
Our final major addition was the inclusion of a DJ-style control option, which has its own patch in MaxMSP and Reason that allows the hands to trigger sample loops, opening up an entire new dynamic of playability to our instrument.

Software:

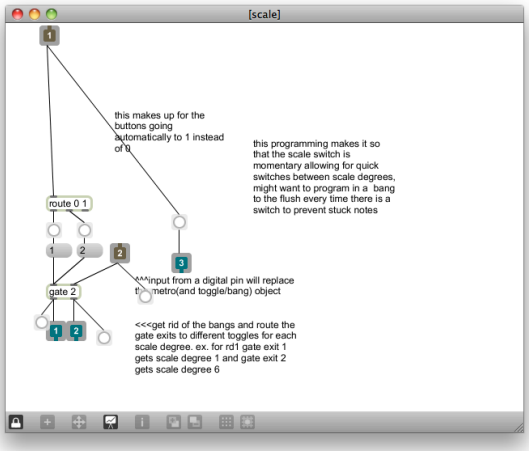
The software has been simplified and made significantly more modular for this second iteration. Given the original design, the software built in MaxMSP takes in the button values on the gamepad and controls a slider that modifies octave shift (left shifts down, right shifts up). There is a series of conditionals attached these triggers, however they are unimportant to an abstracted understanding of the functionality.

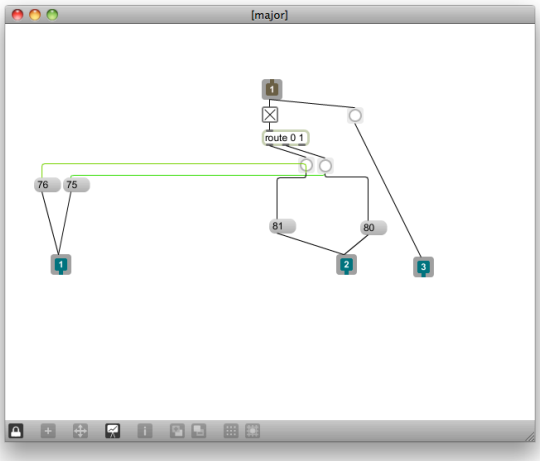


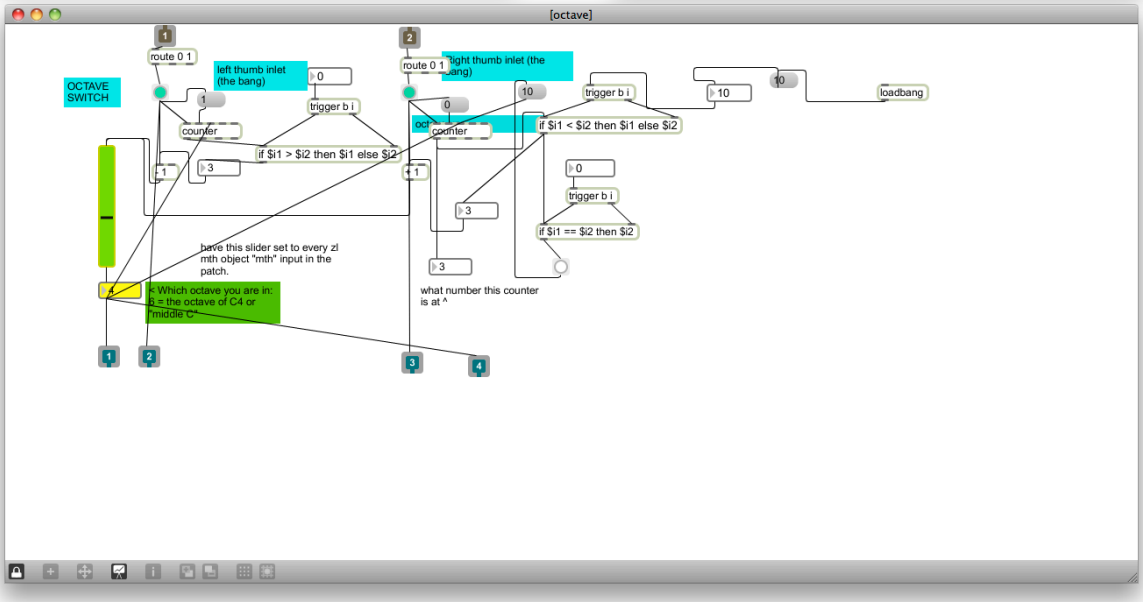
This then relates to a list in 8 essentially identical components of the patch by effecting which element of a list of number values is activated on a button press (using a **toggle**). The proper number is found given the octave passed in and the **zl mth** object. These numbers in turn correspond to notes that are sent as **noteout**s from MaxMSP to Reason.



The velocity is determined by a value derived from the analog-in corresponding to a force sensing resistor on the finger. Math is done on this to get a useful range of numbers and its







output is restricted to 10-127. We chose 10 as the lowest value so that in the case of a non-function FSR a button press would always play a note. We didn't end up implementing this on most of the fingers because we couldn't get the programming down for a really effective use of them (given the issue of the time between button presses and FSR presses), so we set velocity defaults of 90. We attempted to normalize the FSR value using the logarithmic **scale** object but found it difficult and generally ineffective, so instead we placed in values and operations that worked best to normalize each FSR (because not all were quite the same) given normal play. This also helps dilute some of the noise generated by the FSRs.

There are other patch components such as a key shifter operated by a **counter**, **slider** and fixed value of 11 attached to a **bang**. Since Liam wrote a number of really impressive Reason patches including found samples and the like there is also a patch changer that functions similar to the before-mentioned key shifter but with a list of values corresponding to the patch number and with a **zl mth** function.

There is also a bend sensor system that works similar to the FSR system but instead passes information to a midi controller value (like pitch bend or mod) and uses some functions to normalize the values. We didn't end up implementing the bend sensor in the second iteration (though it is functional) because the analog joystick offered far better control.

We also had functions that also utilized lists and a **route** object to switch between major and minor keys, as well as patches (this part also using a **slider**).

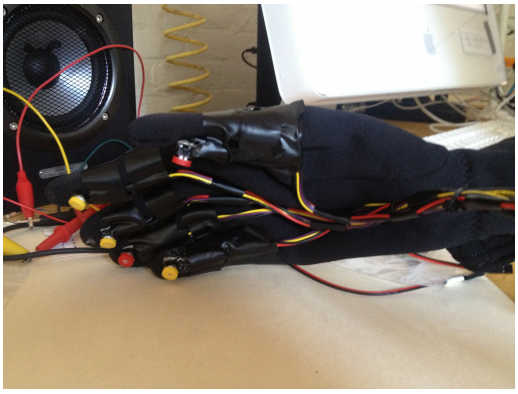
We used Tom's Arduino patch to activate and pass in the data from the Arduino board, and this, like many important components of the program functionality, are loaded through a **load bang**.

The Arduino Mega software/firmware was Tom's.

Sound design:

Liam took charge of sound design and made some fantastic patches, largely composed of

found samples or from unusual instruments. These include: piano, "dub with thirds," organ arpeggios, synth brass, rave chords, glass, "sample mania" (self explanatory), and glass drums. In "dub with thirds" Liam used several subtractor modules coupled with reverb to create an ethereal synth patch. In Synth brass Liam ran a brass instrument device through distortion and reverb and then coupled it with a subtractor synth based on saw waves. The result is a powerful brass sound that could be used in electronic music or even film scoring. The most interesting patch turned out to be sample mania. Liam sampled various objects around the Tufts campus. These sounds are combined to make a drum kit out of natural sounds. Each sample was processed in the Peak Pro software and Protools. The samples were put into multiple NN19

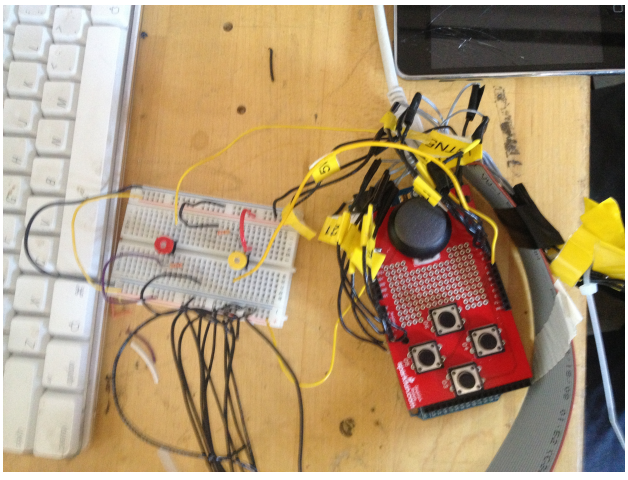


samplers and spread through every octave to ensure playability no matter the octave. In addition reverb and delay were routed through the sends and returns of the internal mixer to give more space and character to the sounds. These sounds prove how this type of playing mechanics is so versatile and works with typical piano-

type sounds, unusual synth sounds, or percussive sounds. It also means that the Jazz Hands can be accessible to a variety of types of players and used in many musical settings.

Hardware:

The Jazz Hands is a glove as well as a gamepad and two additional buttons. The FSRs are mounted below the buttons, paired together (a mounting based on lessons learned with the instrument's first iteration). We also wished to add LEDs but it was not realistic given the time. Each of these components (buttons/FSRs) were put into two wires based off their terminals, one of which was then split into two additional wires. We color coded these for simplicity and used a light gauge of wire to increase flexibility. On one of these ends a resistor was added (1kΩ for the buttons, 22kΩ for the FSRs, 22kΩ for the bend sensors) and these lines eventually went to ground. The other half of this split was sent to the appropriate analog or digital port. The remaining terminals were sent to a five volt source. The wiring was combined together in various ways. All 5V analogs were wired together, as were the digitals, meaning 2 wires instead of 11. The grounds are all compressed onto a breadboard since the resistors had been built into the wiring. This mess of cables was all attached to a ribbon cable (carefully labeled of course) to help neaten the situation. The wires were taped and tied and mounted on the glove for neatness and to increase flexibility.



We also added a game controller and two additional buttons on a breadboard. The game-controller has a PCB so there weren't really any additions needed, though Tom needed to make some firmware modifications to account for the use of the Arduino's built in resistors. The analog stick controls pitch-bend and a second parameter (such as LFO or mod). This is especially interesting on the drum patches, giving us pitched percussion. The four buttons control octave switches, key switching, and patch switching. The two additional buttons (those on the breadboard) act as momentary buttons to add the rest of the scale beyond the available fingering and a major/minor switch. This makes the glove hugely playable and complex, yet still simple to pick up and play for the less musically inclined.

The components were mounted to the outside of the glove (except for the bend sensor) because putting them inside would have made the gloves unwearable.

We chose to use buttons and FSRs to have redundancy and a system in which we knew the note would be consistently triggered (momentary buttons tend to be more reliable sensors than FSRs).

The Arduino Mega was our microcontroller, which is an incredibly robust system however we did have some seemingly dead ports. The Mega was chosen for its large number of analog ports.

Issues:

The most obvious issue you can notice when you look at the Jazz Hands is that there's only one hand. After what we learned from the first try at the instrument, we decided to stick with one hand but design an instrument understanding it would be one glove and one hand

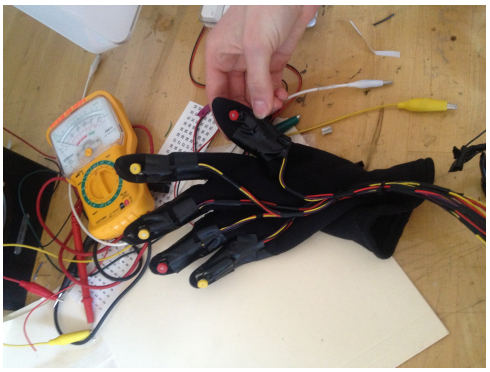
modifying the playability. This worked really well and actually opened up a lot of control and tons of options. We also moved much more quickly the second time around which meant a far less buggy instrument. Also the FSRs were terribly flimsy, ripping and tearing, and they gave inconsistent data (compared to one another, that is), though a high resistance improved this. This complex wiring and lack of time is why we decided against the LEDs (functionality was more important than looks). After completing this second glove we definitely have a better understanding of circuit design and the challenges involved, as well as troubleshooting physical systems that will be infinitely helpful in any future projects. Luckily software was not a major issue for us and neither was sound design. Our teams hardware skills definitely improved and I think we made a really good instrument that offered a lot of options given our limitations.

Roles:

Liam: Programming, sound design

Saul: Hardware, some programming

Drew: Hardware, some programming



Doing it again:

Building another iteration of the gloves was a much smoother, more elegant looking project that went *much* more quickly. Using thinner gauge wires also made a huge usability difference. Definitely solving the problems of effectively mounting the controls to the garment, and doing so in a visually appealing way, would be nice. Finding a way to simplify everything would be good, however FSRs still seem to inconsistent as a way of triggering notes exclusively

although it would mean nearly half the wiring. It is definitely a tradeoff between complexity and efficacy. The FSR programming is also very complex given the timing and it would be nice to figure out an effective way to handle that. However, complexity in and of itself is dangerous. Also having a secondary *hand* controller would be nice for more dynamic, active playability. Working out the ergonomics and perfecting the playability would be a good way to advance the product, too.