Alex, George, and Kevin.

OPL FP4 Proposal

Revision 1A – Written and development by Alex Gribov, Diagram and development by George Mitwasi

**ABSTRACT:**

In this group project, we would like to use Racket to bridge together programmable digital synthesizer software called Max, with the Arduino open-source hardware platform, to create a novel music composition tool called **sdfdsf**. Max offers the capability to program a custom audio synthesizer, which can be controlled by MIDI input, and this is a separate project that one of our group members is doing in a different class. What our Racket program will do, is take in user commands and MIDI input through various sources, provide the user a variety of functions and features that manipulate their input and control the sound, then send the data to Max, where it will be processed into sound, and output through the speakers.

**FEATURES:**

Our program will have a handful of unique features that allow the user to interact with the software, and compose their own music. These include:

* **Piano input**
  + We will use Rackets Overtone? library, which can input MIDI data from a physical keyboard connected via USB, and output MIDI data to Max’s virtual MIDI port. This allows us to process the MIDI data in Racket before sending it out to Max, which will turn that data into sound. For example, a sequence of notes can be input, and if an effect which raises the pitch is applied, then Racket will output a sequence of higher pitch notes to Max.
* **Live Looping**
  + The software written in Racket will enable the user to record a sequence of notes from their MIDI keyboard with a push of a physical button, and then alter the sequence of notes using the effects and controls listed below. Once the loop is recorded, it have its own MIDI channel sent to Max, and can be altered independent of other loops or live sounds. These loops can be a sequence of any sounds, ranging from instruments, to drums, and maybe vocals.
* **Assign effect, sounds, and sequences to physical button**
  + The Max synthesizer will have many built in effects and sounds, and our software will allow the user to run functions that assign control of these effects to physical buttons. In addition to that, we will have the capability to record a series of notes played, and assign it to a button so that sequence can be replicated with one push.
    - List of Effects:
* **Potentiometer control**
  + What’s the difference between the sound of a trumpet, and that of a guitar if they’re playing the same note? There are many factors that affect the timbre, or sound of a note, and small changes in data can drastically affect that sound perceived. Synthesizers use a series of signal processing units such as low frequency oscillators, envelope generators, and oscillators of various waveforms. Our software will allow these units (within Max), to be controlled by potentiometers, so that turning a physical knob can alter the sound output by a particular module.
    - List of things controlled by pot:
* **Presets**
  + Using the features above, the sound can be tweaked to the desired setting. Our software will allow the user to save the setting that they create to a file, and access them in the future, and assign them to physical buttons for quick access.

**HOW IT WORKS:**

As shown in Figure 1 below, our Racket interface will interact with the Arduino Processing library, Overtone composition library, and our own custom Max interface, to make all the magic happen. The processes shown in Figure 1 can be broken into three categories, and divided among the teammate.

Buttons and Knobs

Arduino

Speakers

Synthesizer (Max/MSP)

(converts data to sound)

Custom Racket Interface

Midi Keyboard

Overtone (Racket’s Composition Library)

MIDI

Control Data

Notes

Notes

MIDI

*Figure 1: Graphical workflow of program name.*

**Alex’s job:**

* Design a circuit of potentiometers, lights, and buttons that will connect to an Arduino, and write Racket code that will both send and receive data from the buttons.
* Data from Arduino:
  + Buttons send digital I/O signal, which can be used to turn on and off any preset, or effect on the synthesizer.
  + Potentiometer sends analog signal, which Racket will read and convert to an output on a 0-100 scale. This can be used to control variable effects such as timbre and volume.
  + Touchscreen? Can be used to control 2 dimensions of effects with a single touch. This would be really cool to implement, especially if Fred makes us do more than otherwise planned.
* Data to Arduino:
  + I/O signals to turn on and off LED’s representing various effects in the software.
* Create state-change map for buttons and potentiometers. When a button is pushed, it puts the program into a specific state, and flashes lights to represent that state. See Workflow example below.

**George’s job:**

* Interface Max with Racket, so that Max can receive the I/O data described above, and change the necessary parameter. Will use the following libraries: \_\_\_\_\_\_This can be done because Max has: \_\_\_\_\_\_Another example of non-MIDI data being sent to control a Max program can be found here: \_\_\_\_. NOTE: If you can’t find that example I mentioned, then very VERY strongly consider how you will do this. Turning our data into sound is **on you**, so plan carefully how you plan to do this. It might turn out to be the easiest part of the project, or it might be impossible. You need to figure that out in the next 12 hours.
* George will also be responsible for the flow of MIDI data. Ensuring that the racket code MIDI output can be input to Max’s virtual inputs. (Where will the MIDI be output from? It needs to be MIDI data that has been altered by Kevin’s code. Will his code output it directly to a virtual port? Or will it be sent back to Overtone to be output from there? You and Kevin need to do some joint research as to how this will happen, pick a method, and choose who is responsible for implementation).

**Kevin’s Job:**

* Creating the code that controls the looping, and the effects. Kevin’s code will read in data from the keyboard through Overtone Library, and in some cases take control data output by Alex’s Arduino interface functions. It will use all of this data as inputs to functions which will apply the necessary effects, or initiate looping, and output the MIDI data to Max (Please refer to what I wrote in George’s section for details on this. I wasn’t sure who will be in charge of this, since it is the interface between your work and Max, so you two should both read up on the topic and hash that out without me).
* George’s comment from what he sent me:
  + - Goal: Take MIDI input and have Overtone interpret and use it as its own data.
      * This is a lot trickier than it seems. An obstacle I thought of is that MIDI holds data for the length of a note in seconds and doesn’t take into account a note’s musical length. We need the note’s musical length to change BPM, arrangement, etc. Not sure how that conversion will work.
      * My opinion: I think this would be really simple, but maybe I’m missing something? Do we even need to convert its length to sheet music-style format? Not necessary for what we’re doing in my opinion. Even if we did want to do that, there’s plenty of software that does that. I’m sure we can look it up and code it in Racket if it comes to it. Kevin, please asses this and report back to us, and let us know if there’s any questions.
* This task includes responsibility for making all the mini-functions that take in data, and apply effects. Ex) Pitchbend function can read accelerometer, and output the MIDI data with an altered pitch. I don’t know exactly what effects we want yet, there’s a space for those on the first page when we figure them out.

**EXAMPLE OF WORKFLOW:**

* Record button is pushed. Channel 1-4 buttons start flashing. (Now in choose channel state)
* Channel 1 button is pushed, it holds steady light, others shut off (record sequence state)
* A sequence is played on the keyboard.
* Buttons are pushed to end sequence. Sequence is saved as “Channel 1”. Lights resume normal function, state is set to default.
* MIDI sequence is stored.
* Channel 1 button is pushed. Channel light turns on (state change), sequence loops.
* Button to assign effect is pushed
* All unassigned buttons light up.
* One of those is pushed.
* It starts flashing, to signal that it is ready for programming
* User types the name of the function he would like to assign to the button. For example: Trumpet.
* Button stops flashing, assignment is complete.
* User pushes newly made “Trumpet” button. Channel buttons flash to signify “Choose channel” state.
* User pushes Channel 1 button, it goes back to previous state, now with Trumpet effect applied.

\*\*\*\* Can be done similarly for analog effects and knobs. Can write up what that looks like if you like what you see so far\*\*\*\*