CS457 Functional Programming
Winter 2012
Term Project - MIDI Drums with Haskore

Russell Miller

March 19, 2012

Abstract

I've recently become interested in MIDI and music technology. Part of that is the class I just took this term called *Modern Music Technology*. I've played guitar on and off for about ten years, but I've never been in a band. The idea that I can construct my band with MIDI is rather appealing. I did a term project in that class to mix MIDI drums with my guitar playing. It didn't turn out wonderfully, but it was a lot of fun to do.

My ambitious hope was to use the Haskell library Haskore to do so. Unfortunately, I didn't pursue that very diligently. I just used a MIDI keyboard. The Haskore Music System is an interesting library, though. I'm going to go through the steps to building a beat. I'll start out basic, since that's about the extent of my knowledge of drums. Perhaps then we'll be able to expand on it.

Set Up

Assuming you have `cabal` the set-up shouldn't be too painful. You can follow the instructions on the Haskore website.[1] The one thing that they don't tell you is that you'll also want to install `haskore-vintage`.

```
$ cabal install haskore-vintage
```

There are helpful tutorials available, but to learn to use the Drum portion of Haskore I just read the source files on Hackage.[2] In particular, the most helpful code comments were in the Drum module.[3]

Drums!

In the Drum module there is even an example drum beat called `funkGroove` that gives you an idea of how to construct your own beats.

```
> funkGroove :: MidiMusic.T
> funkGroove =
>     let p1 = toMusic GM.LowTom        qn na
>         p2 = toMusic GM.AcousticSnare en na
>     in changeTempo 3 (Music.take 8 (Music.repeat
>          ( (Music.line [p1, qnr, p2, qnr, p2,
>                         p1, p1, qnr, p2, enr])
>            =:= roll en (toMusic GM.ClosedHiHat 2 na) )
>        ))
```

[1] http://www.haskell.org/haskellwiki/Haskore
[2] http://hackage.haskell.org/package/haskore
[3] http://hackage.haskell.org/packages/archive/haskore/0.2.0.2/doc/html/src/Haskore-Composition-Drum.html

To hear the effect of this beat (as an actual MIDI file), you'll need to render it. That was covered in the tutorials I found. Here is the boiler plate.

```
> import Haskore.Interface.MIDI.Render as Render
> import Haskore.Composition.Drum (funkGroove)

> main = Render.fileFromGeneralMIDIMusic "drums.midi" funkGroove
```

Sure enough, this results in a MIDI file in your directory that plays a funky beat. From there, we can begin constructing our own beat. Starting from the bottom (I think), we'll just roll the hi-hat. This is being done in `funkGroove` and is the last line.

```
> import Sound.MIDI.General as GM
> import Haskore.Basic.Duration (qn)
> import Haskore.Music (changeTempo)
> import qualified Haskore.Music as Music

> main = Render.fileFromGeneralMIDIMusic "drums.midi" song

> song = changeTempo 3
>      $ Music.take 8
>      $ Music.repeat
>      $ roll qn (toMusic GM.ClosedHiHat 1 na)
```

Here we have a function that sets the tempo, we use `Music.take` to specify 4 measures, and we roll the hi-hat. Here I've changed it to use quarter notes instead of eighth notes. If that sounds Greek to you, each measure of music is broken into 4 beats (using standard time measures for simplicity), so a quarter note takes one fourth of the measure. An eighth note takes one eighth and is twice as fast. We're slowing it down to make the sound build more clearly.

Let's add a bass kick. `funkGroove` didn't bother to use the bass! All of the available instruments are listed under Sound.MIDI.General on Hackage.[4]

```
> song = changeTempo 3
>      $ Music.take 8
>      $ Music.repeat
>      $ hiHat =:= bassKick

> hiHat = roll qn (toMusic GM.ClosedHiHat 1 na)

> bassKick =
>      let kick = toMusic GM.BassDrum1 qn na
>      in Music.line [kick, qnr, qnr, qnr]
```

The `qnr` values here mean "quarter note rest" and are provided in `Haskore.Music`. Also provided is the operatore `(=:=)`, which allows you to add notes in parallel. There is a similar operator `(=+=)` which sequences parts together. What we're doing is basically adding a layer on top of the existing hi-hat. Each time `kick` shows up in that list, there is a quarter note (hence `qn`) of bass drum sound.

[4] http://hackage.haskell.org/packages/archive/midi/0.1.7/doc/html/Sound-MIDI-General.html

This needs snare.

```
> song = changeTempo 3
>      $ Music.take 4
>      $ Music.repeat
>      $ hiHat =:= bassKick =:= snare

> snare =
>      let psk = toMusic GM.AcousticSnare qn na
>      in Music.line [qnr, qnr, psk, qnr]
```

Alright, now it sounds like real drums! Let's mix it up.

```
> song = changeTempo 3
>      $ Music.take 4
>      $ Music.repeat
>      $ hiHat =:= bassKick =:= snare

> hiHat = roll qn (toMusic GM.ClosedHiHat 2 na)

> bassKick =
>      let kick = toMusic GM.BassDrum1 qn na
>      in Music.line [kick, qnr, qnr, qnr, kick, kick, qnr, qnr]

> snare =
>      let psk = toMusic GM.AcousticSnare qn na
>      in Music.line [qnr, qnr, psk, qnr, qnr, qnr, psk, qnr]
```

What I've done here is expand our one-measure beat to cover two measures. Every other measure instead of one bass kick there are two in a row.

Now we'll turn that speed dial back up to make it sound pretty rockin'! Just by changing `qn` to `en` for the hiHat, we get an awesome beat.

I was able to write a few other beats this way, and I decided I should just import most of that code from a module. The way that looks is

```
> module Beatz where

> renderMidi xs = Render.fileFromGeneralMIDIMusic "beat.midi" song
>    where song = changeTempo 3
>                 $ Music.take 6
>                 $ Music.repeat
>                 $ (foldl (=:=) (Music.line [])) xs
```

and now I can just import `Beatz` and call the `renderMidi` function with a list of instruments.

Here's a fun beat that demonstrates it.

```
> import Beatz

> main  = renderMidi [snare, hiHat, bassKick]

> snare = let psk = toMusic GM.AcousticSnare hn na
>         in Music.line [qnr, qnr, psk, qnr, qnr, psk]

> hiHat = roll qn (toMusic GM.ClosedHiHat 2 na)

> bassKick =
>         let k1 = toMusic GM.BassDrum1 hn na -- half note kick
>             k2 = toMusic GM.BassDrum1 en na -- eighth note kick
>         in Music.line [k1, enr, k2, enr, k2, k2, k2, qnr, enr, k2]
```

Where to go from here

I had big ambitions for this project, but I had trouble getting momentum. I spent too much time on small details and had difficulty understanding the big Haskore system. I wasn't sure exactly what I wanted to do with this, but one possibility is to parameterize drum beats in a way that allows someone to use the `Beatz` module and really just pass a few parameters like tempo and number of pieces in the drum set – perhaps also selecting a preset drum "style" – and the rest of the process would be automated. I'm not sure how useful that would be, in comparison to customizing the drum parts yourself.

Hearing these beats goes a long way, and I've uploaded them to Github with my code.
https://github.com/millertime-homework/cs457/tree/master/project

Appendix: Thanks to...

I want to give some credit here to Ben Carr cause he actually knows how to play drums and helped me out by telling me the basics.

Another really useful resource was this Haskore-Guide I used.
https://github.com/nfjinjing/haskore-guide