Investors' report

David Kastrup

October 2013

So here we are again. I've received by first donation in Bitcoin in November (0.1 Bitcoin which actually has more than doubled in exchange value since I received it, as I have not traded it in yet). That's a possibility for paying. While the main advantage of that payment method should eventually be the lack of significant international transfer fees, at the current point of time that's not all too visible compared to the current large fluctuations in exchange value itself.

At any rate: here is the address of my Bitcoin wallet for LilyPond work in case you'd think of going via that route:

1Kw7HZMd8L52BCL9vEjSxdPG4p3phRvtQF

October

Fixed payments (€)
300
3×200
100
2×30
3×25
20
4×10
1195

Variable plans (€)
25 + 0×50 (target €1200 exceeded in September)
25

Totals €1220

So October just about surpassed the €1200 mark needed to pay the basic costs of living plus taxes. And even that mark is only reached because two monthly payments

of €200 from the same contributor happened to fall into the same time window. So November will be €200 or (temporarily) €400 less just because of this timing issue.

It's obvious that the total *number* of contributions is on the decline, so I'll probably try seeing whether people feel ok with me using the pending (I know, I know) release announcement for 2.18 to point out that at least *my* work on LilyPond depends on the support of the community.

Development results

October

October was probably one of the busiest months so far judging by commits: calling

```
git shortlog origin --since 2013/10/01 --until 2013/11/01 -n
```

in a checkout of the project repository lists 55 commits[1] by me, and that's just looking at the development branch. What's in there?

There were a number of improvements to cadenze, the main point being that they now only work by stopping the in-measure time and not messing with any other variables.

There were a number of documentation changes, a crash fix, a complex rewrite and simplification of the parser that now no longer parses any arguments in a hard-wired way (this previously made it impossible to create argument types like "`pitch-or-markup?`"). More importantly, the parser is now in a state where the complex part dealing with understanding music function argument lists is coded consistently rather than fiddling around with the code until it happens to do the right thing most of the time. I probably spent two weeks alone getting that in order, accounting for something like 10 commits.

There were several fixes to command line parsing and some other tools.

Another larger change was to have short grace note phrases beamed automatically, a default corresponding much better with established practice.

Ongoing tasks

We are getting close to LilyPond 2.18. After the branch-off announced in October, the next versions to be released are 2.17.96 on the stable branch, and 2.19.0 on the unstable branch.

Several regressions necessitated holding 2.18 off while putting in a number of fixes. So far, it appears like 2.17.96 will contain all known important fixes, so chances are good that we will be able to release 2.18.0 next.

There have been several additions to the 2.19 branch already. The most interesting one is likely the "standalone rhythms" patch from Issue 3648. For one thing, it makes music input such as

```
{ 4. 8 4 4 }
```

[1] The number tends to drop a few times after the final date has already passed; I wish I understood why.

possible. That does not look all that useful in itself, but for creating music functions that take just a rhythm as argument, this should prove a very nice building block. For example, specifying beaming patterns as a rhythm like that might make for a quite natural interface.

However, this will also be nice as a music input tool: if such "pure rhythms" are not picked up by a music function by the time the whole score is interpreted, the note pitch or drum type is taken from the previous note or chord.

This particularly makes the entry of basic drum patterns much more convenient: compare

```
\new DrumStaff \drummode { tambourine 4. 8 4 4 | r8 8~ 4 4. 8 }
```

with even the version using the shortcut:

```
\new DrumStaff \drummode { tamb4. tamb8 tamb4 tamb |
                           r8 tamb~ tamb4 tamb4. tamb8 }
```

and it's easy to see the advantage. Of course, for a single instrument one would rather use a `RhythmicStaff` and be able to write `c` instead of `tamb`, but the increase in readability is not all that impressive. Being able to omit a repeated pitch or drum type (rather than a repeated duration) is quite helpful for entering and particularly for reading predominantly rhythmic passages or whole parts.

Perspectives

My current computer is having problems and I need to get it serviced: the problem is that it does no longer see the CPU fan spinning. After a bit of panic, I finally learnt that I can bypass the "Fan defect" message when booting with the *Esc* key (if I don't, the computer just switches itself off again). And it's even possible to place the fan in an `auto` setting where it will switch itself on and off according to temperature instead of letting the computer do that. As opposed to the computer doing the temperature management however, there are no intermediate settings. As a result, the noise level is quite distracting. To be honest, I don't expect the fan exchange to be expensive, but I'll have to take the computer elsewhere for service, and the travel is likely going to come dearer than the fan (at least I hope so). Of course, that only holds if indeed an exchange of the fan will fix the problem and it's not the computer circuitry that is at fault.

The standalone rhythm change explained in the last section again makes clear that LilyPond's input language becomes more and more friendly to humans. While most syntax changes are upwards-compatible or can be managed using `convert-ly` for conversion, the continuing evolution of the language poses questions about the long-term archivability of LilyPond. In the long run, it will be important to get things like MusicXML export into LilyPond itself, since only then can one be sure to parse the syntax of a LilyPond file really accurately. Even then, the conversion will be only for one version of LilyPond, but such a MusicXML file might be a somewhat more stable target for

reimporting into a newer version. Not that LilyPond, at the current time, would directly read MusicXML or a comparably stable representation, either.

So archivability and interchange remain an issue with a big question mark, and particularly *because* LilyPond is improving its cooperation with humans.

The SCORA enterprise for synchronized music stands by Jan Rosseel has its first full-length public concert presentation on December 14th and 15th in Leuven, Belgium. The typesetting for the systems is done with LilyPond.

Thanks

as always for making it possible for me to continue my work. I'm doing all I can to let LilyPond 2.18 arrive in time for any celebrations around the winter solstice.

David Kastrup