Investors' report

David Kastrup

November 5, 2012

This time, you get two reports folded into one again. There has been quite a flurry of developments in October, making it hard to find the impetus of writing up a nice report about things that were already a bit behind. Anyway, I caught up with October, and hope you don't mind too much.

September & October

September

Fixed payments (€)
160
150
3×100
60
37
3×25
20
10
812

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

Totals after adding €140 from August reserves: €1047

So while this is about the level that is sustainable, it required pulling in from the August reserves.

October

Fixed payments (€)
160
2×100
80
60
30
3×25
20
16
11
3×10
682

Variable plans (€)
25 + 50 (target €1200)
75

Totals after adding €140 from August reserves: €897

This is barely the level that I am able to manage balancing expenses of living (including rent and health insurance) and income, and it again required pulling in from the August reserves. There are only €140 left from that source. So covering taxes at the end of the year is not likely going to be fun.

Development results

One can list the developments submitted in that time frame of two months using

```
git shortlog origin --since 2012/09/01 --until 2012/11/01 -n
```

in a checkout of the project repository.

It shows 80 commits in that time frame (though there is some duplication for commits backported from 2.17 to 2.16 and being brought into view a second time through the translation branch), and there are a few noteworthy ones.

Apart from incremental work on the big Guile migration (being able to move to Guile v2.0 rather than the currently used v1.8), there is a lot of work on the ultimate goal of not requiring the type of expressions and particularly function calls to be known to the parser in advance. One result is that you can use functions defined with `define-scheme-function` almost anywhere now where one can write a string or markup.

But probably the most important change can be illustrated by what input/regression/line-arrows.ly looked like in 2.17.5, and in 2.17.6:

2.17.5 version:

```
\header {
  texidoc = "Arrows can be applied to text-spanners and line-spanners (such as
}
```

```
\version "2.16.0"

\paper {
  ragged-right = ##t
}

\relative c'' {
  \override TextSpanner #'bound-padding = #1.0
  \override TextSpanner #'style = #'line
  \override TextSpanner #'(bound-details right arrow) = ##t
  \override TextSpanner #'(bound-details left text) = #"fof"
  \override TextSpanner #'(bound-details right text) = #"gag"
  \override TextSpanner #'(bound-details right padding) = #0.6

  \override TextSpanner #'(bound-details right stencil-align-dir-y) = #CENTER
  \override TextSpanner #'(bound-details left stencil-align-dir-y) = #CENTER

  \override Glissando #'(bound-details right arrow) = ##t
  \override Glissando #'arrow-length = #0.5
  \override Glissando #'arrow-width = #0.25

  a8\startTextSpan gis8 a4 b4\glissando
  b,4 | g' c\stopTextSpan c
}
```

2.17.6 version:

```
\header {
  texidoc = "Arrows can be applied to text-spanners and line-spanners (such as
}

\version "2.17.6"

\paper {
  ragged-right = ##t
}

\relative c'' {
  \override TextSpanner.bound-padding = #1.0
  \override TextSpanner.style = #'line
  \override TextSpanner.bound-details.right.arrow = ##t
  \override TextSpanner.bound-details.left.text = #"fof"
  \override TextSpanner.bound-details.right.text = #"gag"
```

```
    \override TextSpanner.bound-details.right.padding = #0.6

    \override TextSpanner.bound-details.right.stencil-align-dir-y = #CENTER
    \override TextSpanner.bound-details.left.stencil-align-dir-y = #CENTER

    \override Glissando.bound-details.right.arrow = ##t
    \override Glissando.arrow-length = #0.5
    \override Glissando.arrow-width = #0.25

    a8\startTextSpan gis8 a4 b4\glissando
    b,4 | g' c\stopTextSpan c
  }
```

Can you spot the difference? Actually, there are two: for one thing, symbol lists like

`#'(bound-details right arrow)`

can now alternatively be written as

`bound-details.right.arrow`

and for another, `TextSpanner` and `bound-details` *can* now be joined into a single list by putting another dot between them.

Dotted lists of names can be employed also as music function arguments, so `\override`-similar commands like `\overrideProperty` and `\tweak` can now also use dotted name lists in argument. Since dots can't be made optional there, using them everywhere makes sense even though the old form, whether spelled as

`Glissando #'(bound-details right arrow)`

or as the new alternative spelling

`Glissando bound-details.right.arrow`

will be supported at least throughout the next stable version.

A number of functions have had their interfaces refined to make use of this newly made equivalence of the Scheme "symbol list" structure with the "dotted word list" LilyPond syntax.

These include `\accidentalStyle`, `\alterBroken`, `\footnote`, `\hide`, `\omit`, `\overrideProperty`, `\shape`, and `\tweak`. Actually, `\hide` and `\omit` are also new in this list and shorthands for often-used overrides. There also have been new commands `\temporary` (for making overrides temporary, meaning that a subsequent `\revert` will restore the previous value rather than the default) and `\single` for turning an override into a tweak, meaning that it will, in contrast to `\once\override`, only affect a single element at a given timestep.

If this does not mean anything to you: those are complementing the existing tools for manipulating settings in a reasonably straightforward way, addressing a number of "I would need to do something like xxx, but the available tools use an unsuitable mechanism" problems. So they improve the chances for you receiving a concise solution (rather than oodles of home-spun Scheme code which *do* the trick but are hard to remember or generalize) to questions asked on the LilyPond user list.

Ongoing tasks

Clearly, the above mentioned change needs quite more work in the line of documentation: this concerns both the normal notation manual as well as tying the logic and system of the new syntax possibilities into the extension manuals and putting the Scheme aspect into context.

Guile 2.0 migration is moving at glacial speed currently. There needs to be more activity on that front. It would likely be over-optimistic to hope we can reach a new stable release series based on Guile 2.0 in time for the spring releases of various GNU/Linux distributions. But we likely also can't continue working with Guile 1.8 indefinitely.

And of course, the MusicXML angle makes most sense to tackle after going Guile 2.0 since this version of Guile offers significant processing tools for XML and it would not make much sense to design interfaces without taking those into account.

I have still quite a bit of work ahead for getting the syntax in general to the flexible and consistent state I want to arrive at, making for a better blend of the type systems and interactions of LilyPond and Scheme. This is mostly happening 'undercover', meaning that just more and more things work according to 'naive' expectations. Of course, as long as some things work and some things don't, the overall gain in consistency is limited.

In terms of supporting stable releases, 2.16.1 is likely to be released in the next days. This also requires coordination, testing, and planning detracting from the work on the stable releases.

Perspectives

There have been several new sponsors including corporate ones, partly recruited from national user lists. It is disconcerting that in spite of that, the contributions have declined considerately in the last two months, reaching a level where I am not even able to sustain myself, let alone acquire enough of a buffer to pay for taxes or pension funds.

Now in October, basically *all* donations have been from regular contributors. This *is* an improvement against the situation when I started, when the majority of contributions were one-time contributions of a size people could not be expected to keep up indefinitely. The downside is that I have my doubts about the ability to make this model work by just appealing to the audience I have so far been able to reach: it seems that most are doing more than they can reasonably expected to do already. Another downside is that my message to the regular contributors can only be: "Please keep it up. I can't make this work without all of you.".

So it would appear that it is necessary to find significant additional sources of funding for my work on LilyPond, or there will not be a viable long-term perspective for staying on the project. Given the suspected number of LilyPond users (looking at download numbers alone), there should be enough with an interest in keeping LilyPond moving forward with regard to being a consistent and flexible system with powerful and straightforward extension mechanisms for music typesetting.

The question is how to reach them.

I'll likely appeal again on the user and developer lists soon, but expect to be mostly preaching to the choir, people who are already aware of the deal I offer on a continuing basis to the LilyPond community and who have made their choice about how they want to make use of it.

In the mean time, I am thanking you for what you do to let me continue my work on LilyPond.

David Kastrup