LilyPond
Automated music formatting
and The Art of Shipping

Han-Wen Nienhuys
LilyPond Software Design
Jan Nieuwenhuizen

7th Fórum Internacional Software Livre
April 20, 2006. Porto Alegre, Brazil
“But that’s been done before, no?”

(Finale 2003)

Gold standard
Hand engraved scores (early 20th century)
Beautiful music typography

- A thing of beauty is a joy forever
- Ease of reading
- Better performance
Automated music typography

- Problem statement
- Design overview
- Examples of *engraving*
- Implementation
  - Typography algorithms
  - Formatting architecture
- Zen and the Art of Shipping Software
- Conclusions
Modeling notation

pitch

measures

chords

\{ staff \\

\text{time}
Modeling notation

Simple hierarchy does not work for complex notation
Divide and conquer

typography \iff notation \iff hierarchical representation

1 Typography: \textbf{where} to put symbols
2 Notation: \textbf{what} symbols for which music
3 Music representation: how to \textbf{encode} music
4 Program architecture: glue together everything
Typography

Music engraving: create pleasing look

- Visual: distance and blackness
- A craft: learned in practice
- No literature
Font

- Heavy look, matching line thickness
- Rounded shapes
- No prescribed rules, so imitate.

Henle (2001)  Bärenreiter (1950)  Feta (LilyPond v2.0)
Spacing

Create illusion of evenness:

Regular spacing:

Optical correction:
Algorithms for aesthetics

- Literature: rule of thumb
- Rule of thumb: cover all cases?
- Case analysis: unwieldy
- Must derive rules from examples
Scoring

- Define ugliness of a configuration
- Try every configuration
- Select least ugly one

\[
\text{variance}=15.39 \quad \text{TOTAL}=15.39 \quad \text{idx}=0
\]

\[
\text{slope}=2.00, \ L \ edge=1.71, \ R \ edge=9.37 \quad \text{TOTAL}=13.08 \quad \text{idx}=13
\]

\[
\text{slope}=2.00, \ R \ edge=10.04 \quad \text{TOTAL}=12.04 \quad \text{idx}=4
\]
Score based formatting

- Beam: stem lengths, slope
- Ties: collisions, notehead/tie distance
- Line breaking
- Page breaking

Pro/Con

+ declarative programming
− expensive

? how to define ugliness?
Program architecture

- Music typography is visual
- Impossible to automate for all cases
- Allow manual override for users
- Need flexible program architecture:

  “Any sufficiently complicated C or Fortran program contains an ad hoc informally-specified bug-ridden slow implementation of half of Common Lisp.”
  (Phil Greenspun’s 10th rule of programming.)
Software duct-tape

Put real LISP interpreter (GUILE Scheme) in C++. Symbols represented by “Layout objects”, containing variables

- **Style:** default values
  `(RepeatSlash
   . ((stencil .
     ,Percent_repeat_item_interface::beat_slash)
    (thickness . 0.48)
    (slope . 1.7)))`

- **Function value:** callback

- **Tweak:** override defaults
Benchmarking output

LilyPond 1.4

Bärenreiter

LilyPond 2.7.29

Han-Wen Nienhuys  Automatic music formatting (FISL 7.0)
LilyPond today

- 9.5 years old; 9.5 man-years
- 10,000 downloads/month. 20,000 to 100,000 users?
- Most frequent comments: “Thank you,” “Beautiful output.”
- Focus on engraving is unique.
- Support through LilyPond Software Design
LilyPond today

- 9.5 years old; 9.5 man-years
- 10,000 downloads/month. 20,000 to 100,000 users?
- Most frequent comments: “Thank you,” “Beautiful output.”
- Focus on engraving is unique.
- Support through LilyPond Software Design

<table>
<thead>
<tr>
<th>(March ’06)</th>
<th>Total</th>
<th>Linux</th>
<th>Windows</th>
<th>MacOS X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>%</td>
<td>%</td>
<td>%</td>
</tr>
<tr>
<td>Website</td>
<td>26,389</td>
<td>21</td>
<td>67</td>
<td>11</td>
</tr>
<tr>
<td>Downloads</td>
<td>12,966</td>
<td>11</td>
<td>67</td>
<td>22</td>
</tr>
</tbody>
</table>

- Non-technical program: non-technical users
Ship early, ship often

Why binaries?

- Get rid of install/compile questions
- Limit version support burden
- Quality control
- Expand user base
- Condition for paid support
First attempts

- Build LilyPond + dependencies (±20)
- Existing solutions: fink, mknetrel, autopackage, etc.

Problems

- Duplication of effort
- Unreliable & unpredictable
- Need native machine
Enter GUB, *Grand Unified Builder*

- Mini package manager/distribution builder
- Cross-compiling: no native machine required
- Assemble into single installer
- Python based: No More Shell Scripts!
- Python class = package build spec
- Share code for platforms via inheritance
- Bugfix rollout: 25 min (6 platforms, Celeron 2GHz).
Build your own binaries

Lessons

- Long feedback cycle
- Cross-building: libtool Shiatsu and autoconf Voodoo
- Unix relocation: not there yet.
- Windows32 sucks.

Future

- automated release testing?
- continuous building/testing?
- use for other packages too?
Conclusions

- Music typography: subtle and difficult, but fun
- Computer engraving
  - score based aesthetics
  - flexible program architecture
  - benchmarking: compare with real engraving
- End-user software: ship early, ship often
sie-gend durch_ die Nacht. Schö-ner Tag, du_bist er - wacht._ Mit ge-
das be-weg - te_ Herz, sanft, wie ein ge-lieb - ter Schmerz._ Dürft ich
Print music database

Automatically convert MIDI, MusicXML, ABC → SVG, PDF or pixmap:

(RISM 2002 database)
Collect scores on-line
http://www.mutopiaproject.org/

Approximately 5000 pages of music.