

# TUGBOAT

Volume 34, Number 3 / 2013  
TUG 2013 Conference Proceedings

<b>TUG 2013</b>	246	Conference sponsors, participants, and program
	252	Norbert Preining / <i>TUG 2013 in Tokyo</i>
<b>Publishing</b>	259	Didier Verna / <i>The incredible tale of the author who didn't want to do the publisher's job, ...</i>
<b>L<sup>A</sup>T<sub>E</sub>X</b>	263	Jason Lewis / <i>How to make a product catalogue that doesn't look like a dissertation</i>
<b>Multilingual Document Processing</b>	268	Clerk Ma and Jie Su / <i>Project Fandol: GPL fonts for Chinese typesetting</i>
	269	Matthew Skala / <i>Tsukurimashou: A Japanese-language font meta-family</i>
	279	Clerk Ma / <i>Braille fonts in Project Fandol</i>
	281	Ken Nakano and Hajime Kobayashi / <i>Case study: Typesetting old documents of Japan</i>
	285	Takuji Tanaka / <i>upT<sub>E</sub>X—Unicode version of pT<sub>E</sub>X with CJK extensions</i>
	289	John Plaice / <i>Typesetting and layout in multiple directions—Proposed solution</i>
<b>Software &amp; Tools</b>	293	Norbert Preining / <i>T<sub>E</sub>X Live Manager's rare gems: User mode and multiple repository support</i>
	297	Norbert Preining / <i>Redistributing T<sub>E</sub>X and friends</i>
	302	Andrew Mertz and William Slough / <i>A gentle introduction to PythonT<sub>E</sub>X</i>
	313	Lu Wang and Wanmin Liu / <i>Online publishing via pdf2htmlEX</i>
	325	Shinsaku Fujita / <i>The X<sup>T</sup>M<sub>E</sub>X system for publishing interdisciplinary chemistry/mathematics books</i>
	329	Pavneet Arora / <i>TANSU—A workflow for cabinet layout</i>
<b>Bibliographies</b>	332	Nathan Hagen / <i>Bibulous—A drop-in BIBT<sub>E</sub>X replacement based on style templates</i>
	340	Michael Cohen, Yannis Haralambous and Boris Veytsman / <i>The multibibliography package</i>
<b>Graphics</b>	344	Aleksandra Hankus and Zofia Walczak / <i>L<sup>A</sup>T<sub>E</sub>X and graphics: Basics and packages</i>
	349	Boris Veytsman and Leyla Akhmadeeva / <i>Plots in L<sup>A</sup>T<sub>E</sub>X: Gnuplot, Octave, make</i>
	357	Mari Voipio / <i>Entry-level MetaPost 3: Color</i>
<b>Abstracts</b>	360	TUG 2013 abstracts (Cho, Hagen, Hakuta, Maeda & Kaneko, Minoda, Mittelbach, Moore, Shikano, Takata, Terada, Verna, Wetmore, Yabe)
	363	ConT <sub>E</sub> Xt Group: Proceedings, 6th meeting (2012)
	364	MAPS: Contents of issue 44 (2013)
	365	Die T <sub>E</sub> Xnische Komödie: Contents of issues 3–4/2013
<b>Hints &amp; Tricks</b>	366	Karl Berry / <i>The treasure chest</i>
<b>General Delivery</b>	367	Denis Bitouzé / <i>In memoriam: Jean-Pierre Drucbert (1947–2009)</i>
<b>Book Reviews</b>	368	Dave Walden / <i>Book review: Essential Knuth</i>
	369	Clerk Ma / <i>Book review: Introduction to L<sup>A</sup>T<sub>E</sub>X</i>
<b>TUG Business</b>	370	TUG institutional members
<b>Advertisements</b>	370	T <sub>E</sub> X consulting and production services
<b>News</b>	372	Calendar

## T<sub>E</sub>X Users Group

*TUGboat* (ISSN 0896-3207) is published by the T<sub>E</sub>X Users Group.

### Memberships and Subscriptions

2014 dues for individual members are as follows:

- Regular members: \$105.
- Special rate: \$75.

The special rate is available to students, seniors, and citizens of countries with modest economies, as detailed on our web site. Also, anyone joining or renewing **before March 31** receives a \$20 discount:

- Regular members (early bird): \$85.
- Special rate (early bird): \$55.

Membership in the T<sub>E</sub>X Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed, as well as software distributions and other benefits. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in TUG elections. For membership information, visit the TUG web site.

Also, (non-voting) *TUGboat* subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. The subscription rate is \$105 per year, including air mail delivery.

### Institutional Membership

Institutional membership is a means of showing continuing interest in and support for both T<sub>E</sub>X and the T<sub>E</sub>X Users Group, as well as providing a discounted group rate and other benefits. For further information, see <http://tug.org/instmem.html> or contact the TUG office.

### Trademarks

Many trademarked names appear in the pages of *TUGboat*. If there is any question about whether a name is or is not a trademark, prudence dictates that it should be treated as if it is. The following list of trademarks which commonly appear in *TUGboat* should not be considered complete.

T<sub>E</sub>X is a trademark of American Mathematical Society.  
METAFONT is a trademark of Addison-Wesley Inc.  
PostScript is a trademark of Adobe Systems, Inc.

[printing date: December 2013]

Printed in U.S.A.

### Board of Directors

Donald Knuth, *Grand Wizard of T<sub>E</sub>X-arcana*<sup>†</sup>

Steve Peter, *President*\*

Jim Hefferon\*, *Vice President*

Karl Berry\*, *Treasurer*

Susan DeMeritt\*, *Secretary*

Barbara Beeton

Kaja Christiansen

Michael Doob

Steve Grathwohl

Taco Hoekwater

Klaus Höppner

Ross Moore

Cheryl Ponchin

Arthur Reutenauer

Philip Taylor

Boris Veytsman

David Walden

Raymond Goucher, *Founding Executive Director*<sup>†</sup>

Hermann Zapf, *Wizard of Fonts*<sup>†</sup>

\*member of executive committee

<sup>†</sup>honorary

See <http://tug.org/board.html> for a roster of all past and present board members, and other official positions.

#### Addresses

T<sub>E</sub>X Users Group

P. O. Box 2311

Portland, OR 97208-2311

U.S.A.

#### Electronic Mail

(Internet)

General correspondence,  
membership, subscriptions:  
[office@tug.org](mailto:office@tug.org)

#### Telephone

+1 503 223-9994

Submissions to *TUGboat*,

letters to the Editor:

[TUGboat@tug.org](mailto:TUGboat@tug.org)

#### Fax

+1 815 301-3568

Technical support for

T<sub>E</sub>X users:

[support@tug.org](mailto:support@tug.org)

#### Web

<http://tug.org/>

<http://tug.org/TUGboat/>

Contact the Board

of Directors:

[board@tug.org](mailto:board@tug.org)

Copyright © 2013 T<sub>E</sub>X Users Group.

Copyright to individual articles within this publication remains with their authors, so the articles may not be reproduced, distributed or translated without the authors' permission.

For the editorial and other material not ascribed to a particular author, permission is granted to make and distribute verbatim copies without royalty, in any medium, provided the copyright notice and this permission notice are preserved.

Permission is also granted to make, copy and distribute translations of such editorial material into another language, except that the T<sub>E</sub>X Users Group must approve translations of this permission notice itself. Lacking such approval, the original English permission notice must be included.

**2013 Conference Proceedings**

T<sub>E</sub>X Users Group  
Thirty-fourth Annual Meeting  
Tokyo, Japan  
October 23–26, 2013

# **TUGBOAT**

COMMUNICATIONS OF THE T<sub>E</sub>X USERS GROUP

TUGBOAT EDITOR      BARBARA BEETON

PROCEEDINGS EDITOR    KARL BERRY

VOLUME 34, NUMBER 3

PORTLAND

•

OREGON

•

•

2013

U.S.A.

**TUG 2013**

**Tokyo, Japan**

**October 23–26, 2013**

## Sponsors

- Graduate School of Mathematical Sciences, The University of Tokyo (東京大学大学院数理科学研究科)
- SANBI Printing Co., Ltd. (三美印刷株式会社) • Gijutsu-Hyohron Co., Ltd. (株式会社技術評論社)
- Tokyo Educational Institute Co., Ltd. (Tetsuryokukai) (株式会社東京教育研(鉄緑会))
- Green Cherry Ltd. (株式会社 Green Cherry) • PLAIN corporation (株式会社プレイン)
- Livretech Co., Ltd. (株式会社リーブルテック) • Top Studio Co., Ltd. (株式会社トップスタジオ)
- ULS & Company (株式会社ウルス) • Tatsu-zine Publishing Inc. (株式会社達人出版会)
- Ohmsha, Ltd. (株式会社オーム社) • Kato Bunmeisha Co., Ltd. (株式会社加藤文明社印刷所)
- Fujiwara Printing Co., Ltd. (藤原印刷株式会社) • Saiensu-sha Co., Ltd. (株式会社サイエンス社)
- Suurikougaku-sha Co., Ltd. (株式会社数理工学社) • Enishi Tech Inc. (株式会社えにしテック)
- Maruzen Publishing Co., Ltd. (丸善出版株式会社)
- Dainippon Hourei Printing inc. (大日本法令印刷株式会社)
- Gravel Road Inc. (株式会社グラベルロード)
- TeX Users Group • Korean TeX Society • DANTE e.V.
- Thomas BIETENHADER • KOSAKAI Eiichiro (小酒井 英一郎) • KIEDA Yuwsuke (木枝 祐介)
- SHIBATA Mitsuya (柴田 充也) • FUJIMURA Yukitoshi (藤村 行俊) • NARIAI Kyoji (成相 恭二)
- KANOU Hiroki (狩野 宏樹) • AOKI Yoshihiro (青木 義弘) • SHIKANO Keiichiro (鹿野 桂一郎)
- and many anonymous sponsors.

*Thanks to all!*

ご協力頂いた全ての方へ感謝いたします。

## Conference committee

- ABE Noriyuki (Hokkaido Univ.) • Karl BERRY (TUG) • HONDA Tomoaki (SANBI Printing)
- ICHII Shingo (The Univ. of Tokyo) • KUROKI Yusuke • OKUMURA Haruhiko (Mie Univ.)
- Steve PETER (TUG) • Norbert PREINING (JAIST)
- TAKAHASHI Masayoshi (Tatsu-zine Publishing) • YAMAMOTO Munehiro (Green Cherry)

## Bursary committee

- Steve PETER, chair • Jana CHLEBIKOVA • Bogusław JACKOWSKI • Alan WETMORE

## Participants

(anonymous)

ABE Noriyuki  
阿部 紀行  
*Creative Research Institution,  
Hokkaido University*

AKIMOTO Yoshitomo  
秋元 良友  
*Student, Faculty of Science and  
Engineering, Chuo University  
Yokosuka, Japan*

ABE Hirosuke

阿部 央輔  
*Graduate School of Arts and  
Sciences, The University of Tokyo  
Japan*

Leila AKHMADEEVA  
レイラ アフマディエーヴァ  
*Bashkir State Medical University  
Bashkortostan, Russia*

Maissa ALAMEDDINE  
メイッサ アラメッдин  
*Sydney, NSW, Australia*

- AOKI Yasuhiro  
青木 康博  
*Department of Forensic Medicine,  
Nagoya City University  
Nagoya Japan*
- AOKI Yoshihiro  
青木 義弘  
*freelance, TDON KK  
Tokyo, Japan*
- Pavneet ARORA  
パヴアニート アーロラ  
*Bolton, ON, Canada*
- ASE Harumi  
阿瀬 はる美  
*ARS System Corporation*
- Nelson H F BEEBE  
ネルソン ビービー<sup>1</sup>  
*University of Utah  
Salt Lake City, UT, USA*
- Barbara BEETON  
バーバラ ビートン  
*American Mathematical Society  
Providence, RI, USA*
- Thomas BIETENHADER  
トーマス ビーテンハーダ  
*Bussersdorf, Switzerland*
- Jin-Hwan CHO  
趙 珍煥 / 조 진환 / チョウ ジンワン  
*Korean TeX Society  
Seoul, Republic of Korea*
- Jennifer CLAUDIO  
ジェニファー クラディーオ  
*Synopsis Outreach Foundation  
San Jose, CA, USA*
- Michael COHEN  
公園 マイケル  
*Spatial Media Group,  
University of Aizu  
Aizu-Wakamatsu, Fukushima,  
Japan*
- FUJINO Seiji  
藤野 清次  
*RIIT, Kyushu University  
Hakozaki, Fukuoka, Japan*
- FUJITA Shinsaku  
藤田 真作  
*Shonan Institute of  
Chemoinformatics and  
Mathematical Chemistry  
Kanagawa, Japan*
- FUJIWARA Makoto  
藤原 誠  
*KINU Corporation  
Chiba, Japan*
- FUKAYAMA Osamu  
深山 理  
*IST, The University of Tokyo  
Tokyo, Japan*
- FUNAKI Naoto  
船木 直人  
*Tokyo, Japan*
- Hans HAGEN  
ハンス ハーゲン  
*Pragma ADE  
Hasselt, Netherlands*
- HAKUTA Shizuya  
白田 静哉  
*Japan*
- HAMADA Tatsuyoshi  
濱田 龍義  
*Faculty of Science,  
Fukuoka University  
Fukuoka, Japan*
- HAMAMO Hisato  
濱野 尚人
- Aleksandra HANKUS  
オラ ハンクス  
*University of Silesia  
Katowice, Poland*
- HASHIMOTO Ryuta  
橋本 竜太  
*Kagawa National College  
of Technology  
Mitoyo, Kagawa, Japan*
- HASHIMOTO Takafumi  
橋本 孝文  
*College of Arts and Sciences,  
Tokyo University  
Tokyo, Japan*
- HAYASAKA Miwako  
早坂 美和子
- HAYASHI Tsunetoshi  
林 恒俊  
*Takatsuki, Osaka, Japan*
- HAYASHIDA Hiroki  
林田 裕樹
- Malte HELMHOLD  
マルテ ヘルムホルド  
*System Development,  
TU Dresden,  
Dresden, Saxony, Germany*
- HIMEI Akira  
姫井 晃  
*Morisawa Inc.  
Tokyo, Japan*
- HONDA Tomoaki  
本田 知亮  
*SANBI Printing Co., Ltd.  
Tokyo, Japan*
- ICHII Shingo  
一井 信吾  
*Graduate School of Mathematical  
Sciences, The University of Tokyo  
Tokyo, Japan*
- ITOH Hiroyuki  
伊藤 裕之  
*Gravel Road Inc.  
Nagoya, Japan*
- IWASAKI Shinichi  
岩崎 慎一  
*SANBI Printing Co., Ltd.  
Tokyo, Japan*
- KAIO Naoto  
海生 直人  
*Faculty of Economic Sciences,  
Hiroshima Shudo University  
Hiroshima, Japan*
- KANADA Naoki  
金田 直樹  
*Tokyo, Japan*
- KANAZAWA Katsuhiko  
金沢 克彦
- KANEKO Masataka  
金子 真隆  
*Faculty of Pharmaceutical  
Sciences, Toho University  
Kisarazu, Japan*
- KANOU Hiroki  
狩野 宏樹  
*IWATA Corporation  
Tokyo, Japan*
- KASHIMA Junko  
鹿島 順子  
*Tokyo, Japan*
- KATO Maiko  
加藤 麻衣子
- KAWABATA Taichi  
川幡 太一  
*NTT Network Innovation  
Laboratory  
Tokyo, Japan*
- KAWAKUBO Toru  
川久保 亮
- KIEDA Yuwsuke  
木枝 祐介  
*Tokyo, Japan*
- Young Rock KIM  
金 永錄 / 김 영록 / キム ヨンロック  
*Mathematics Education, Hankuk  
University of Foreign Studies  
Seoul, Korea*
- KITAGAWA Hironori  
北川 弘典  
*Tokyo, Japan*

- KITAMURA Naru  
北村 成  
*LSI Japan Co., Ltd.*  
*Tokyo, Japan*
- KITTA Yoriyuki  
橘田 順之  
*University of*  
*Electro-Communications*  
*Tokyo, Japan*
- KIZAKI Saki  
木崎 早樹  
*Kawasaki, Japan*
- KOBAYASHI Hajime  
小林 肇  
*Livrettech Co., Ltd.*  
*Tokyo, Japan*
- KOBAYASHI Ken  
小林 健
- Daniel KOBAYASHI-BETTER  
ダニエル 小林ベター  
*Faculty of Letters,*  
*Kobe University*  
*Hyogo, Japan*
- KOJIMA Tadashi  
小嶋 忠詞  
*Japan*
- KOSAKAI Eiichiro  
小酒井 英一郎  
*Kenkyusha Printing Co., Ltd.*
- KUME Ayaka  
久米 純佳
- KUROKI Yusuke  
黒木 裕介  
*Yokohama, Japan*
- Jason LEWIS  
ジェイソン ルイス  
*Organic Trader Pty Ltd*  
*Sydney, NSW, Australia*
- Wanmin LIU  
柳 万民 / リイウ ワンミン  
*The Hong Kong University of*  
*Science & Technology*  
*Hong Kong*
- MAEDA Kazuki  
前田 一貴  
*Japan*
- MAEDA Yoshifumi  
前田 善文
- MASUKO Moe  
増子 萌
- MATSUURA Tomoyuki  
松浦 智之  
*Rich Laboratoy*  
*Machida, Tokyo, Japan*
- MATSUZAKI Shuji  
松崎 修二  
*SANBI Printing Co. Ltd.*  
*Nishinippori, Tokyo, Japan*
- Andrew MERTZ  
アンドルー メルツ  
*Eastern Illinois University*  
*Charleston, IL, USA*
- Jessica MERTZ  
ジェシカ メルツ  
*Charleston, IL, USA*
- Lothar MEYER-LERBS  
ローター マイヤーレルブス  
*Bremen, Germany*
- MINODA Yasuhide  
蓑田 恭秀  
*Tokyo Educational Institute*  
*(Tetsuryokukai)*  
*Tokyo, Japan*
- Frank MITTELBACH  
フランク ミッテルバッハ  
*LATEX3 Project*  
*Mainz, Germany*
- MIYAKAWA Noriyoshi  
宮川 憲欣  
*KEIBUNDO Co., Ltd.*  
*Shiodocho, Shinjuku, Tokyo, Japan*
- MIYOSHI Akihiro  
美吉 明浩  
*PLAIN corporation*  
*Tokyo, Japan*
- Robyn MOORE  
ロビン ムーア  
*Mount Colah, NSW, Australia*
- Ross MOORE  
ロス ムーア  
*Mathematics Department,*  
*Macquarie University*  
*Sydney, NSW, Australia*
- MUTO Kenshi  
武藤 健志  
*Top Studio Corporation &*  
*Debian Project*  
*Tokyo, Japan*
- NAGATA Yoshihisa  
永田 善久  
*Faculty of Humanities,*  
*Fukuoka University*  
*Fukuoka, Japan*
- NAITO Takashi  
内藤 郁之
- NAKAI Wataru  
中井 渉  
*Shiga, Japan*
- NAKAJIMA Junji  
中島 淳二
- NAKAMURA Masaya  
中村 真也  
*School of Engineering,*  
*The University of Tokyo*  
*Tokyo, Japan*
- NAKANO Ken  
中野 賢  
*Livrettech Co., Ltd.*  
*Tokyo, Japan*
- NARIAI Kyoji  
成相 恭二  
*National Astronomical*  
*Observatory of Japan*  
*Suginami, Tokyo, Japan*
- OKADA Ryo  
岡田 亮  
*Kato Bunmeisha Printing Co.,*  
*Ltd.*  
*Chiyoda-ku Tokyo, Japan*
- OKUDONO Takamasa  
奥殿 貴仁  
*Faculty of Liberal Arts,*  
*The University of Tokyo*  
*Saitama, Japan*
- OKUMURA Haruhiko  
奥村 晴彦  
*Faculty of Education,*  
*Mie University*  
*Tsu, Japan*
- OSBORNE Masako  
オズボーン 昌子  
*ULS & Company*  
*Tokyo, Japan*
- OSHIMA Toshio  
大島 利雄  
*Faculty of Science,*  
*Josai University*  
*Tokyo, Japan*
- OTOBE Yoshiki  
乙部 嶽己  
*Department of Mathematical*  
*Sciences, Shinshu University*  
*Matsumoto, Japan*
- John PLAICE  
ジョン プレイス  
*Montreal, Canada*  
*UNSW in Sydney, Australia*
- Norbert PREINING  
ノルベルト プライニング  
*Japan Advanced Institute*  
*of Science and Technology*  
*Kanazawa, Ishikawa, Japan*
- Chris ROWLEY  
クリス ローリー  
*LATEX3 and FutureLearn, UK*  
*London, UK*

- SAIKAWA Takafumi  
才川 隆文  
*Graduate School of Mathematics,  
Nagoya University  
Nagoya, Japan*
- SAITO Shingo  
斎藤 新悟  
*Faculty of Arts and Science,  
Kyushu University  
Fukuoka, Japan*
- SAITO Taiichi  
斎藤 泰一  
*SANBI Printing Co., Ltd.  
Tokyo, Japan*
- SAKAMOTO Noriaki  
酒本 典明  
*Tokyo, Japan*
- SATO Chiyoko  
佐藤 智代子  
*ULS & Company  
Tokyo, Japan*
- SATO Kiyoshi  
佐藤 基昭  
*ULS & Company*
- Volker RW SCHAA  
フォルカー シャー<sup>1</sup>  
*DANTE e.V.  
Darmstadt, Germany*
- SENOO Ken  
妹尾 賢  
*Department of Environmental  
Engineering, Graduate School  
of Engineering, Kyoto University  
Osaka-fu, Japan*
- SHIINA Takehito  
椎名 建仁  
*PLAIN Corporation  
Tokyo, Japan*
- SHIKANO Keiichiro  
鹿野 桂一郎  
*Nishi-Nippori, Tokyo, Japan*
- SHIMIZU Mikiko  
清水 美貴子
- SHIRO M.
- SHISHIKURA Mitsuhiro  
宍倉 光広
- Maria SHMILEVICH  
マリア シュミレヴィッチ  
*Sterling, VA, USA*
- Matthew SKALA  
マッシュ スカラ  
*University of Manitoba  
Winnipeg, MB, Canada*
- Marlene SLOUGH  
マリーン スラウ  
*Charleston, IL, USA*
- William SLOUGH  
ビル スラウ  
*Eastern Illinois University  
Charleston, IL, USA*
- SUDO Masaki  
須藤 真己  
*Gijutsu-Hyohron Co., Ltd.  
Tokyo, Japan*
- SUGIMURA Mika  
杉村 美佳
- SUWA Takashi  
諏訪 敬之  
*College of Arts and Sciences,  
The University of Tokyo  
Nishinomiya, Japan*
- SUZUKI Hayao  
鈴木 駿  
*The University of  
Electro-Communications, Tokyo*
- SUZUKI Kentaro  
鈴木 健太郎  
*Tokyo, Japan*
- SUZUKI Shigeya  
鈴木 茂哉  
*Graduate School of Media and  
Governance, Keio University  
Tokyo, Japan*
- TAKAGI Kazuhito  
高木 和人
- TAKAHASHI Masayoshi  
高橋 征義  
*Tatsu-Zine Publishing Inc.  
Tokyo, Japan*
- TAKATA Yumi  
高田 裕美  
*Typebank Co., Ltd.  
Tokyo, Japan*
- TAKATO Setsuo  
高遠 節夫  
*Faculty of Science,  
Toho University  
Kisarazu, Japan*
- TAKENAKA Yoshiro  
竹中 義朗  
*SANBI Printing Co., Ltd  
Nishinippori, Tokyo, Japan*
- TAMORI Hideaki  
田森 秀明  
*The Asahi Shimbun Company  
Tokyo, Japan*
- TANAKA Takuji  
田中 琢爾  
*the upTeX project  
Tokyo, Japan*
- TATSUZAWA Masahiro  
立澤 正博  
*Maruzen Publishing Co., Ltd.  
Tokyo, Japan*
- TERADA Yusuke  
寺田 侑祐  
*Tokyo Educational Institute  
(Tetsuryokukai)  
Tokyo, Japan*
- TOGASHI Hideaki  
富樫 秀昭  
*Japan*
- TSUCHIMURA Nobuyuki  
土村 展之  
*Kwansei Gakuin University  
Sanda, Japan*
- UJIMORI Akira  
氏森 瑛  
*SANBI Printing Co., Ltd.  
Tokyo, Japan*
- Didier VERNA  
ジジエ ベルナ / 侍侍榮 舞瑠那  
*LRDE, EPITA  
Le Kremlin-Bicêtre, France*
- Boris VEYTSMAN  
バリス ヴェイツマン  
*George Mason University  
Fairfax, VA, USA*
- Zofia WALCZAK  
ゾフィア ワルチャク  
*Polish TeX Users Group (GUST)  
& University of Lodz  
Lódź, Poland*
- Alan WETMORE  
アラン ウエットモア  
*Army Research Laboratory  
Adelphi, MD, USA*
- YABE Masafumi  
家辯 勝文  
*Tokyo, Japan*
- YADA Tsutomu  
矢田 勉  
*Graduate School of Letters,  
Osaka University  
Osaka, Japan*
- YAMAKAWA Kazuki  
山川 和樹  
*SANBI Printing Co., Ltd.  
Tokyo Japan*
- YAMAMOTO Munehiro  
山本 宗宏  
*Green Cherry Ltd.  
Chiba, Japan*
- YATO Takayuki  
八登 崇之
- YATSUI Tomoaki
- YAZAWA Yuko  
矢澤 祐子
- YOSHINAGA Tetsumi  
吉永 徹美  
*Tokyo, Japan*

# TUG 2013 — program and information

**Tuesday,  
22 October**

18:00 *registration and reception*

**Wednesday,  
23 October**

08:30	<i>registration</i>	
09:00	Steve PETER & Haruhiko OKUMURA	<i>Opening message</i>
09:15	Didier VERNA	<i>TiCL: The prototype</i>
09:50	Shizuya HAKUTA	<i>LISP on T<sub>E</sub>X: A LISP interpreter written using T<sub>E</sub>X macros</i>
10:05	Andrew MERTZ & William SLOUGH	<i>A gentle introduction to PythonT<sub>E</sub>X</i>
10:40	<i>break</i>	
11:00	Yusuke KUROKI	<i>Tutorial: Introduction to tutorials</i>
11:10	Tsutomu YADA	<i>Tutorial: An introduction to the structure of the Japanese writing system</i>
12:40	<i>lunch</i>	
13:40	Didier VERNA	<i>The incredible tale of the author who didn't want to do the publisher's job</i>
14:15	Hans HAGEN	<i>How we try to make working with T<sub>E</sub>X comfortable</i>
14:50	Keiichiro SHIKANO	<i>Tutorial: Indexing makes your book perfect</i>
15:35	<i>break</i>	
15:55	Jason LEWIS	<i>How I use L<sub>A</sub>T<sub>E</sub>X to make a product catalogue that doesn't look like a dissertation</i>
16:30	Yasuhide MINODA	<i>T<sub>E</sub>X in educational institutes</i>
17:05	Lu WANG & Wanmin LIU	<i>Online publishing via pdf2htmlEX</i>
17:40	Frank MITTELBACH	<i>The stony route to complex page layout</i>
	<i>After a short break, an extra session was held for those interested:</i>	
18:35	Hans HAGEN	<i>What is ConT<sub>E</sub>Xt? A short introduction.</i>

**Thursday,  
24 October**

08:30	<i>registration</i>	
09:00	Yoshifumi MAEDA & Masataka KANEKO	<i>Making math textbooks and materials with T<sub>E</sub>X+KETpic+hyperlinks</i>
09:30	Alan WETMORE	<i>Wind roses for T<sub>E</sub>X documents</i>
10:05	Boris VEYTSMAN & Leila AKHMADEYEVA	<i>Plots in L<sub>A</sub>T<sub>E</sub>X: Gnuplot, Octave, make</i>
10:40	<i>break</i>	
11:00	Yumi TAKATA	<i>Tutorial: Japanese typeface design — similarities and differences from Western typeface design</i>
12:30	<i>lunch</i>	
13:30	Aleksandra HANKUS & Zofia WALCZAK	<i>L<sub>A</sub>T<sub>E</sub>X and graphics</i>
14:05	Frank MITTELBACH	<i>L<sub>A</sub>T<sub>E</sub>X3: Using the layers</i>
14:50	<i>break</i>	
15:10	Masafumi YABE	<i>Tutorial: Japanese text layout — basic issues</i>
16:40	Yusuke KUROKI	<i>Tutorial: Notes on Japanese T<sub>E</sub>Xt processing</i>
16:55	<i>break</i>	
17:15	Yusuke TERADA	<i>Development of TeXShop — the past and the future</i>
18:25	Norbert PREINING	<i>T<sub>E</sub>X Live Manager's rare gems: User mode and multiple repository support</i>
18:50	(Organizing Committee)	<i>Guidance for the excursion</i>

**Friday,  
25 October**

Full day excursion: letterpress printing and calligraphy workshops.

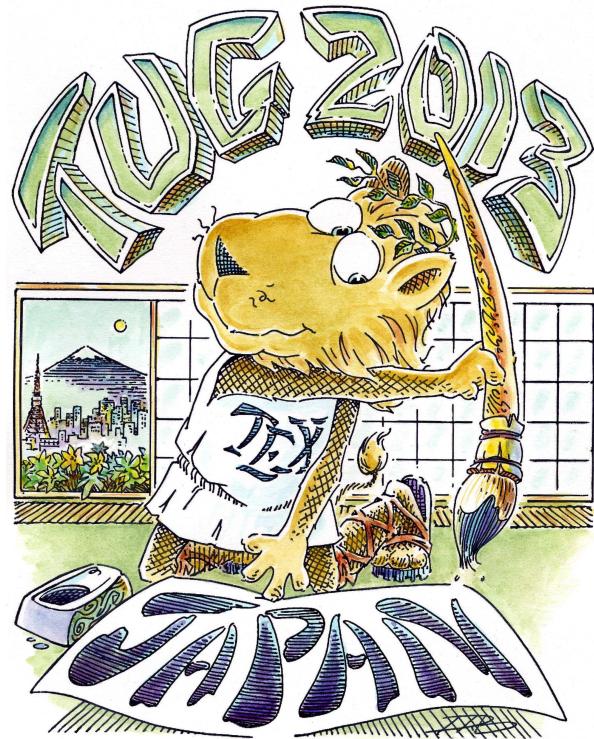
**Saturday,  
26 October**

08:30	<i>registration</i>	
09:00	Matthew SKALA	<i>Tsukurimashou: A Japanese-language font meta-family upT<sub>E</sub>X—Unicode version of pT<sub>E</sub>X with CJK extensions</i>
09:35	Takuji TANAKA	
10:10	Hiroki KANOU	<i>Tutorial: On the possibility of automatic balancing of ideographic character design</i>
10:25	Haruhiko OKUMURA	<i>Tutorial: Japanese typesetting for the mathematically oriented</i>
10:45	<i>group photo</i>	
10:50	<i>break</i>	
11:10	Ken NAKANO & Hajime KOBAYASHI	<i>A case study: Typesetting old documents of Japan</i>
11:45	Jin-Hwan CHO	<i>A case study on T<sub>E</sub>X's superior power: Giving different colors to building blocks of Korean syllables</i>
12:20	<i>lunch</i>	
13:20	Michael COHEN, Yannis HARALAMBOUS, Boris VEYTSMAN	<i>The multibibliography package</i>
13:55	Pavneet ARORA	<i>TANSU: A workflow for cabinet layout</i>
14:30	John PLAICE	<i>Typesetting and layout in multiple directions</i>
15:05	<i>break</i>	
15:25	Ross MOORE	<i>Making mathematical content accessible using Tagged PDF and L<sup>A</sup>T<sub>E</sub>X</i>
16:00	Hans HAGEN	<i>How we move(d) on with math</i>
16:35	Shinsaku FUJITA	<i>The X<sup>Y</sup>M<sub>T</sub>E<sub>X</sub> system for publishing interdisciplinary chemistry/mathematics books</i>
17:10	Norbert PREINING	<i>Distributing T<sub>E</sub>X and friends: Methods, pitfalls, advice</i>
17:45	Norbert PREINING	<i>Closing message</i>
18:00	<i>banquet</i>	



## TUG 2013 in Tokyo\*

Norbert Preining



In 2013, the TUG conference was held for the first time in Japan, at the University of Tokyo. The following was originally published on my blog [1] and edited for publication. So you want to know what you missed if you weren't able to be there? Here are my very personal recollections!

### 1 Pre-conference reception

The day before the actual conference started we had a nice reception in the university building of the conference. All kinds of snacks and drinks and warm-up chat made the hours fly by. Although I came straight from my home in Kanazawa and arrived a bit late, I had a wonderful time. Especially for me, meeting all the old friends I haven't seen for long time was a great pleasure.

### 2 First day

#### 2.1 Morning

The first day started with an opening address of Haruhiko OKUMURA<sup>1</sup> and, via Skype, from the president of TUG, Steve PETER. Steve brought his Ja-

\* Thanks to Haruhiko Okumura and Pavneet Arora for providing photos

<sup>1</sup> For consistency we will use *First LAST* for all names here, with apologies to Japanese tradition to write the surname first.

panese to a quasi-near-native level and was honored with big applause for that (and probably also for what he said, but I can't remember that as well).

The first session brought presentations on extensions and reimplementations of  $\text{\TeX}$ :

- Didier VERNA — TiCL: The prototype
- Shizuya HAKUTA — LISP on  $\text{\TeX}$ : A LISP interpreter written using  $\text{\TeX}$  macros
- Andrew MERTZ — A gentle introduction to Python $\text{\TeX}$

While I am myself a great lover of Lisp, I somehow couldn't crank my brain to think about implementing a typesetting engine in Lisp. Still, fun to hear and see the development over the years. Shizuya HAKUTA's talk was another proof that we can do everything in  $\text{\TeX}$  — which leaves the question of whether we *should* do everything. Programming Lisp in  $\text{\TeX}$  doesn't sound like something I will ever want to do. But the technical accomplishment was impressive! Finally, Andrew MERTZ's introduction to Python $\text{\TeX}$  helped a lot and gave a nice and accessible starting point for using Python as an extension in  $\text{\TeX}$ .

The second session was dedicated to tutorials:

- Yusuke KUROKI — Introduction to tutorials
- Tsutomu YADA and Daniel KOBAYASHI-BETTER — An introduction to the structure of the Japanese writing system

A short introduction to the tutorials by Yusuke KUROKI was followed by an excellent tutorial by Tsutomu YADA and Daniel KOBAYASHI-BETTER on the history, structure, and peculiarities of the Japanese writing system. Filled with examples, old and new, a very enjoyable time. Paired with the insight into Japanese culture given by the combined presentation of a professor and his assistant, it was a memorable experience in all senses. I only hope they can get permission to publish all their presentation slides, since I would very much like to read over them once more.

#### 2.2 Afternoon

The first session in the afternoon brought two talks and a tutorial:

- Didier VERNA — The incredible tale of the author who didn't want to do the publisher's job
- Hans HAGEN — How we try to make working with  $\text{\TeX}$  comfortable
- Keiichiro SHIKANO — Indexing makes your book perfect

Didier's second talk gave us funny stories about his life as author-editor-fighter for human rights in the publishing business. Filled with anecdotes on how bad it can get when you are working with an in-

competent publisher, he reminded many of us of our own hard times. Hans HAGEN's talk tried to make us more comfortable with  $\text{\TeX}$ —well, most of us may already be comfortable with it, but I guess his work is very much appreciated since levels of comfort vary. Mine, for example, is very low. The moment I see an  $\backslash\expandafter$  I run away screaming. The indexing tutorial of Keiichiro SHIKANO gave a good overview of problems with indexes in various languages, filled with nice examples of Manga usage for teaching math/statistics.

After another break followed one of the highlights in my opinion (but then, there were so many highlights!), a mixture of talks, each a pearl of its own:

- Jason LEWIS—How I use  $\text{\LaTeX}$  to make a product catalogue that doesn't look like a dissertation
- Yasuhide MINODA— $\text{\TeX}$  in educational institutions
- Wanmin LIU—Online publishing via pdf2htmlEX
- Frank MITTELBACH—The stony route to complex page layout

Jason LEWIS started off with how he managed to generate a catalog for his wholesale business in down-under, which used a lot of different techniques merged together. I liked how he didn't get religious and presented  $\text{\TeX}$ , Perl, MS Access and more, mixed together to get his company working. The following talk then blew me away: Yasuhide MINODA presented the installation of  $\text{\TeX}$  as the main document processor in a preparation school for the University of Tokyo entrance exam. About 200 teachers there were trained in  $\text{\LaTeX}$ , and now all the products, from homework to internal notes, are done in  $\text{\LaTeX}$ . That was completely beyond my imagination—a history teacher, or classical Chinese teacher—using  $\text{\LaTeX}$ . Same for the next talk by Wanmin LIU, who presented a program to convert PDF to HTML. The conversion is not done from the source code, but from the PDF, and the output looks very much—often nearly indistinguishable—from the original PDF. Great work. Finally, one of Frank MITTELBACH's great talks on how complicated things can be, especially when it comes to multi-column typesetting and the wishlist of users. I didn't know till now that this is so complicated, but now I do.

### 2.3 Evening

After the successful first day a few people spent the rest of the evening in a nice izakaya (casual pub) with food from the southern parts of Japan (Kyushu and Okinawa), accompanied with lots of Orion beer,

fun talk, and much laughter.

## 3 Second day

The second day brought a big section on graphics, two excellent tutorials on Japanese typefaces and text layout, a hands-on tutorial on typing Japanese on computers, and plenty of other talks.

### 3.1 Morning

The morning session started with a series of talks on how to use various graphics packages:

- Masataka KANEKO—Making math textbooks and materials with  $\text{\TeX}+\text{KETpic+hyperlinks}$
- Alan WETMORE—Wind roses for  $\text{\TeX}$  documents
- Boris VEYTSMAN & Leila AKHMADEYEVA—Plots in  $\text{\LaTeX}$ : Gnuplot, Octave, `make`

Myself being a hard-core  $\text{TikZ}$ -user, I still enjoy seeing other graphics system. In the case of  $\text{\TeX}+\text{KETpic+hyperlinks}$ , I was surprised what can be done. It would have been even more interesting to me to see more actual code, as I want to know whether it is easy to write such code. Alan WETMORE's talk presented us some beautifully designed wind roses. I really appreciate these kinds of talks, since we always have lots of technical talks; some artistic design reminds us that we should go forth and create *beautiful* works of printing. Last in the first session was Boris VEYTSMAN (who gets the prize for the most questions—as far as I can remember he had questions or comments after each talk) and Leila AKHMADEYEVA on how to automate plot generation using `make` and Gnuplot and Octave.

Before lunch we had one tutorial, one I was eagerly awaiting:

- Yumi TAKATA—Japanese typeface design: Similarities and differences from Western typeface design

This tutorial started with an excellent introduction to Japanese writing styles and its history, only slightly overlapping with the tutorial of the first day. After that Yumi TAKATA got into the specifics of type design and how to create the huge amount of glyphs necessary. In the last part she gave a glance at the difficulties of encodings in use. Although I personally would have liked to hear more about the actual design process and technical procedure, this tutorial was one of the highlights for me. All the pieces were very well presented and explained. Thanks!

### 3.2 Afternoon

The first session in the afternoon brought two talks:

- Aleksandra HANKUS & Zofia WALCZAK— $\text{\LaTeX}$  and graphics

- Frank MITTELBACH— $\text{\LaTeX}3$ : Using the layers
- Our guests from Poland were very enthusiastic in presenting the history and different options of using graphics in  $\text{\LaTeX}$ . While there were a few omissions, they did a very good job in reminding us of what else there is besides  $\text{TikZ}$ . But maybe it was only me who was reminded. Frank MITTELBACH spoke about  $\text{\LaTeX}3$  and how it is structured—or how it will be structured. While the future of  $\text{\LaTeX}3$  is not clear to me, even after that talk, I see the ‘use-now’ packages in ever-growing use in  $\text{\LaTeX}2\epsilon$ , so I am confident that we will see further developments.

After a short break (Let me mention here that the coffee breaks were excellent, too. The variety of snacks, cakes, crackers, strange tube-shaped sweets, fruit gelees, and much else I’ve forgotten, really drew the attention of at least all the foreign attendees! Thanks to the team!) another set of two tutorials:

- Masafumi YABE—Japanese text layout:  
Basic issues
- Yusuke KUROKI—Some notes on Japanese  
 $\text{TeX}$  processing

Masafumi YABE is a long-term contributor to several standards of Japanese text layout, and thus the perfect source of detailed information. Layout of the page in a typical Japanese book, details about spacing between Japanese and non-Japanese glyphs, vertical versus horizontal typesetting—you name it. All the important information without losing oneself into the details. After that, Yusuke KUROKI gave a hands-on tutorial on how to actually input Japanese text. Supported by fast-fingered Moe MASUKO, they explained how to input text on smart-phones as well as computers in a variety of ways. He also gave some warnings concerning implementations and spacing, as well as the current state of  $\text{TeX}$  engines.

The last session brought two talks, on  $\text{TeXShop}$  and  $\text{TeX}$  Live Manager:

- Yusuke TERADA—Development of  $\text{TeXShop}$ —the past and the future
- Norbert PREINING— $\text{TeX}$  Live Manager’s rare gems: User mode and multiple repository support

First, Yusuke TERADA gave a good overview of the current state of  $\text{TeXShop}$ , one of the very user-friendly  $\text{TeX}$  editors on Mac, and how over the years, thanks to him and other Japanese developers, the capabilities with respect to Japanese typesetting have been improved. His experiments with his own name, containing a special kanji, were very amusing, since it often gets garbled up during operations. I guess for many in the audience seeing these examples finally made them understand how nasty beasts are lying down there in the implementations, and often not

even companies like Apple manage them properly. Unfortunately one talk had to be cancelled, namely “ $\text{TeX}$  Live for Android”, since the presenter could not attend the conference. A pity, as many had been looking forward to that talk. The last one for the day was my own talk on  $\text{TeX}$  Live Manager’s rare gems, user mode and multiple repository support. While I have talked already last year about multiple repositories, a few more features have been added over the year to  $\text{tlmgr}$ . And user mode, although not often used, needed some explanation, too.

### 3.3 Evening

After the successful second day, not surprisingly, a few people spent the rest of the evening in a nice okonomiyaki place—a bit upmarket and posh, but with excellent food. And Didier finally got his most-beloved Japanese food, a nice Kyoto-style okonomiyaki (a savory Japanese pancake) . . . not to forget the beer.

## 4 Third day—Excursion

The third day of the conference was dedicated to an excursion to the Tokyo Printing Museum [2] housed in the Toppan Printing Company’s building. Divided into three groups, we took turns in three activities: a guided tour through the museum, a letterpress printing workshop, and a calligraphy workshop.

### 4.1 Printing museum

After arriving in the museum with the bus and first introduction the three groups started off into their courses. My group started with the guided tour, and I was responsible for translating the guide’s explanations from Japanese to English for our foreign guests. I must apologize here for the poor and incomplete translations.

Before entering the main exhibition space we were guided along a wall filled with replicas of famous objects related to writing. From a copy of the stele of the Code of Hammurabi (the original is in the Louvre), over ancient Chinese and Japanese prints, to a Gutenberg Bible, from French cave drawing over Japanese Hanga art to modern books and ads, all in replicas, all to be touched, all to be experienced. It was the second time that I visited the museum, and I believe this is an interesting and funny idea. Of course it is impossible to have all the originals, but the collection along the timeline creates an interesting effect.

The guided tour was followed by some free time to explore the exhibition space and a lunch at the restaurant in the same building.

## 4.2 Calligraphy workshop

Satisfied with the morning and with stomachs filled we were off to the calligraphy workshop. An initial introduction and welcome message followed by two teachers showing us the variety of calligraphy by writing the Japanese letter for wind 風, and let us compare their two writings. One I could recognize without any problems, the other on the contrary looked so stylistic that I had no chance of recognizing it—even Japanese colleagues nearby slightly twisted their heads while trying to decipher it.

After that we were shown how to make ink, how to use the brush, some techniques on drawing lines, etc. And then we dived into practice. All of us had a bunch of exercise papers which we filled more or less eagerly with our own inspiration. I for myself was a complete newcomer to calligraphy. I remember the only time I had to write with a brush was during my wedding in a Japanese shrine, and I was so nervous that I did not even manage to write the most simple things, not remotely thinking of calligraphy and style. So I considered this my first trial, and consequently filled page after page with simple kanjis—or kanjis I thought to be simple: 山 (mountain), 岳 (peak), 水 (water), 道 (path), and words like 山岳 (mountains) etc. (I realized that I have an inclination to mountains, not so strangely for those who know of my non-TEX activities!)

After having practiced for some time—and me actually running out of exercise paper due to my frantic writing—we were told that now is the time to create our *masterpieces*. Meaning that we got a nice (and bigger) piece of paper, and a bigger brush, and should decide on something to write, meditate on the meaning of this particular word or sign, and then draw it full of our own feelings. After everyone has finished this and some signing and stamping a seal onto the *masterpieces*, everyone stood up and explained what he wrote and why.

I choose the path 道, with a quite wild look and the character somehow running out of the frame. For me it was a bit like my future, unclear on how and where. Others wrote words related to printing, to feelings, to their families. All very interesting and nice pieces of personality, if not to say of art.

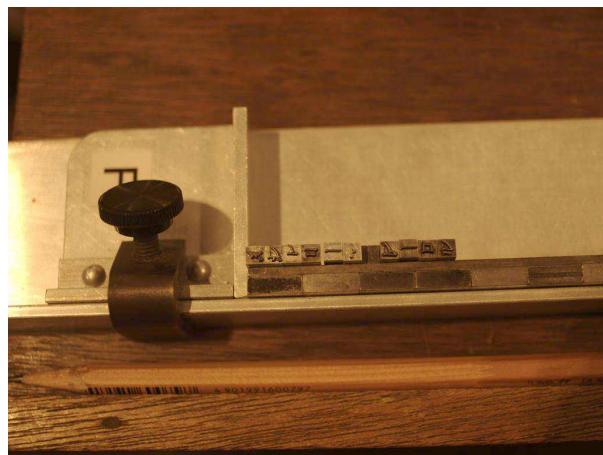


After packing up all our exercise sheets and masterpieces we were sent off with more presents—a

calligraphy written by one of the masters, candies, and origami—and returned to the printing museum.

## 4.3 Letterpress printing workshop

Returning to the printing museum, we had more free time. Some explored the exhibition space as well as the temporary exhibition on the first floor, some retreated to the coffee house for relaxation. But soon we all gathered together for the last activity of the day, the letterpress printing workshop. We were about to set our names in katakana and print bookmarks with it. We started with putting the metal types into the composing stick. This wasn't so difficult except for the small metal types and my clumsy fingers.



Having managed to compose our names and centering it (and getting my name's spelling corrected—after four years I still have problems), the names were transferred into a bigger frame by the instructor, put into the letterpress printing machine, and after some trial runs we were ready to print our bookmarks. Splitting them carefully and packaging them up, we had to wait for a day or so, but now all of us have three beautifully printed bookmarks with our names—self-made!



Having finished the printing workshop, we still had time to explore the surroundings of the workshop room, where many strange and disturbing things could be found: Metal type of Japanese kana at 3.5 pt—that is so small that I couldn't even see the letters on it, never mind trying to move them. Also real type, I mean not electronic fonts that have been downloaded, but real metal types of hundreds of fonts. How beautiful. I could have spent hours digging through old fonts, and trying to print all kind of things by myself.



But soon we had to leave for the last part, a movie of typesetting math—in the old style. The printing museum personally set a page of mathematics in old style for us, made a movie and— incredible—a 3D animation how the setting was done—in which order, which pieces come in when. We were all deeply impressed, both by the difficulty of setting math by that method, and the love of detail with which they have produced the movies. Here you see the final metal frame used to print the mathematics.



Filled with lots of new experiences we left the museum at around 6 pm for the conference and hotel

area. From the feedback we got already on the bus, everyone really enjoyed the time.

#### 4.4 Evening

After this once again fully packed day, I ventured out with other organizers and spent the evening over excellent fish, delicious sake, long discussions about typography, the conference, and much more.

### 5 Last day

The last day brought a wild mixture of math, fonts, touching various aspects and exhibiting the power of *TeX & friends* far from its original target.

#### 5.1 Morning

The morning session started with two talks and two short tutorials related to Japanese typesetting and typography:

- Matthew SKALA—Tsukurimashou: A Japanese-language font meta-family
- Takuji TANAKA—upTeX: Unicode version of pTeX with CJK extensions
- Hiroki KANOU—On the possibility of automatic balancing of ideographic character design
- Haruhiko OKUMURA—Japanese typesetting for the mathematically oriented

In the first talk Matthew SKALA introduced the audience in a very humorous style to the components of kanji, and how he is using them in building up a Metafont family for Japanese (and more). Enriched with lots of Manga-like cartoons and episodes he not only presented the essentials of his Tsukurimashou project, but also additional tools for searching in large kanji-corpora for constituents. For a practical one-man-project, a very impressive achievement. The next talk also featured a one-man project: Takuji TANAKA's upTeX, a Unicode-enabled version of pTeX, the main typesetting engine in Japan. I myself am deeply in gratitude to TANAKA-san, as I use upTeX almost exclusively. Many of my files are UTF-8, and in addition contain not only ASCII, but also German umlauts and other Latin-1 characters. A breeze with upTeX—big thanks!

After the two regular talks a scheduled presentation on the history of *TeX* in China unfortunately had to be canceled due to visa problems. Two Japanese colleagues graciously stepped forward to give short tutorials. The first was by Hiroki KANOU on automatic balancing in character design. I think the few participants included in the audience very much enjoyed the presentation, as it gave interesting points on how to balance stroke width in ideographic characters to achieve a balanced output. What type

designers for Latin characters normally do on a one-by-one basis requires some approach of automation to achieve in the context of thousands of ideographs. The second tutorial by Haruhiko OKUMURA recapitulated the spacing aspect of the tutorial on Japanese text layout from the second day, but targeting mathematicians, by providing a representation of the spacing rules compressed into a simple table.

After the obligatory group photo we had another session before lunch with two excellent presentations on the power of  $\text{\TeX}$ :

- Ken NAKANO & Hajime KOBAYASHI—A case study: Typesetting old documents of Japan
- Jin-Hwan CHO—A case study on  $\text{\TeX}$ 's superior power: Giving different colors to building blocks of Korean syllables

Ken NAKANO told us about the great pains their company has to go through to typeset old documents, with all the scientific necessity of corrections, corrections of corrections, corrections of corrections of corrections ... and so on. Packaging all of these peculiarities into macros and producing an actual well-printed book was very impressive. During my studies of Latin and Greek I was often confronted with ‘critical apparatus’ as it is called, pages of references and citations and quotations. But what was shown there surpassed the worst critical apparatus I have ever seen.

The last talk before lunch was by our honored guest from Korea, Jin-Hwan CHO, well known for his contributions to various Korean  $\text{\TeX}$  packages as well as the main author of `dvipdfmx`, widely used not only in Korea but also in Japan. His talk gave a short introduction to the Hangul characters and their formation, followed by an excursion into auto-composition of all the Hangul characters from relatively few components. And as a consequence the ability to display the parts of Hangul ideographs in different colors, something completely unthinkable with any other software. It was particularly interesting for me to see the relation between the first talk and this one, both touching the problem of how to compose glyphs from simpler components.

## 5.2 Afternoon

The first session in the afternoon brought three talks:

- Michael COHEN & Boris VEYTSMAN—The multibibliography package
- Pavneet ARORA—TANSU: A workflow for cabinet layout
- John PLAICE—Typesetting and layout in multiple directions

Michel COHEN and Boris VEYTSMAN presented a new approach to bibliographies, based on the idea

that having references sorted in only one way might not be sufficient, and differently sorted views onto the references should be provided. I think it an excellent idea, particularly for online publications where page limits are not so strict, especially for all those bibliographies with hundreds of entries. I see a great potential for this idea, but would like to see it integrated into the `biblatex` package.

Pavneet ARORA gave us a view into a different world, the world of interior design, especially quick sketch-ups of cabinets using  $\text{\TeX}$  and friends. Integrating many different tools (YAML,  $\text{\TeX}$ , Asymptote, etc.) into a professional workflow.

Before the break, John PLAICE, renowned as a co-creator of  $\Omega$ , guided us through the intricacies of typesetting directions and mixing them. With his long years of experience in dealing with these problems, John described a concise and clear blueprint for complete support of all necessary text directions, as well as guidance in the problems of mixing directions. His examples were very elaborate—but most impressive was how fast and without any failure he could pronounce the word ‘pneumonoultramicroscopicsilicovolcanoconiosis’.

After another short break, we had the last session of this conference:

- Ross MOORE—Making mathematical content accessible using Tagged PDF and  $\text{\LaTeX}$
- Hans HAGEN—How we move(d) on with math
- Shinsaku FUJITA—The  $\text{\XeLaTeX}$  system for publishing interdisciplinary chemistry/mathematics books
- Norbert PREINING—Distributing  $\text{\TeX}$  and friends: Methods, pitfalls, advice

Accessibility is more and more often a requirement for many publications. Ross MOORE gave us a view of what is possible with tagging PDFs for proper audio reproduction. The demonstrations were quite funny, because the Adobe Acrobat program seems to randomly decide which document to read out loud, and then stick to it for a long time. Still, it was impressive to see what difference can be achieved in the audio output of the PDF content by adding some features.

After this multimedia experience we returned to the original virtues of  $\text{\TeX}$  in Hans HAGEN's talk on math typesetting. Recapitulating the history and presenting the current status, Hans came to the sad conclusion that  $\text{\TeX}$  is no longer paving the way, but running behind other players. But he didn't leave us completely without hope. As the cards are remixed several times,  $\text{\TeX}$  might jump forward again with new techniques mixing OpenType features with the layout excellence of  $\text{\TeX}$ .

The following talk by Shinsaku FUJITA on the X<sup>M</sup>T<sub>E</sub>X system gave insights into the development and usage of his drawing package for chemical structural formulas. Enriched with many examples from his books, it was a great pleasure to see X<sup>M</sup>T<sub>E</sub>X in action.

I myself had the honor to give the last talk of the conference on distributing T<sub>E</sub>X Live. I tried to give a quick overview of what distributors (such as Debian, Red Hat, SuSE, ...) have to take care for when re-packaging T<sub>E</sub>X Live for the respective distributions. Since I am involved in both the upstream development of T<sub>E</sub>X Live as well as the Debian re-packaging of T<sub>E</sub>X Live, I thought it would be good to sum up common mistakes and errors which we encounter.

Another very full day was finished, and I also had the honor to give the closing remarks.

## 6 Closing

As it was my honor to close the conference I want to convey the same thoughts I tried to express during the closing address. Let us start with the hard facts: 141 active participants (at least), 35 interesting and funny talks, an excursion full of experiences, and not to forget the long chats during breaks, dinner, at any free time. It might be one of the best-attended TUG conferences ever. I have only been at TUG 2012 in Boston, and only checked a few of the former conferences, but thanks to the huge interest in the Japanese T<sub>E</sub>X community the number of participants exceeded all our expectations.

And not only the number of participants, but also the number of presentations—35, some of them 1.5 hr tutorials—made for long and dense days. And in spite of this challenging schedule, most participants attended virtually all the talks, even when we finished around 8pm. The variety of talks was not less than at any other T<sub>E</sub>X conference, something I really appreciate—one never gets bored.

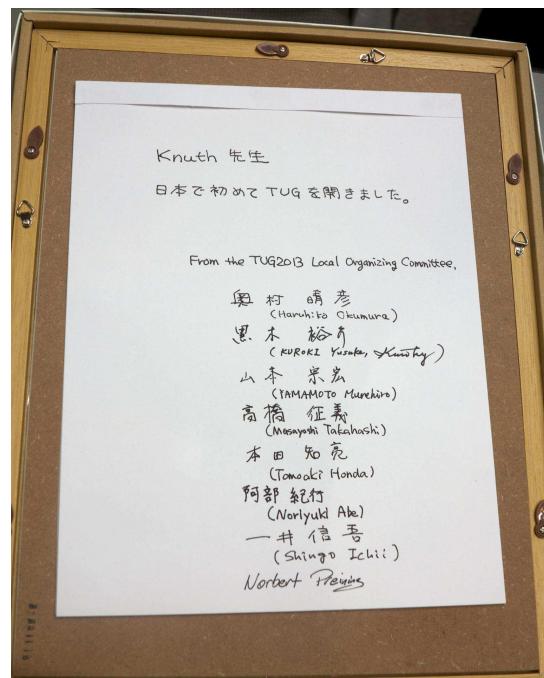
### 6.1 Conference dinner

After all the formal talks and greetings we changed over to another event, the conference dinner. Our excellent guide during the evening, Harumi ASE, led us through a program of speeches by Nelson BEEBE, various toasts by Shinsaku FUJITA, greetings from Haruhiko OKUMURA as the chair of the organizing committee, as well as a closing message by Barbara BEETON. All accompanied by excellent food and lots of drinks. Even after the Sambon-jime led by Yusuke KUROKI the drinking and partying continued until we had to leave the dinner location. A memorable conference dinner for a memorable conference!

## 6.2 At the end

It was the first time that the TUG conference came to Japan, and I remember well the first reaction of my Japanese colleagues to this proposal: ‘The Japanese side is not ready for this.’ I think the conference showed all of us, the guests as well as the hosts, that the Japanese T<sub>E</sub>X users were in fact very well prepared. And for this my gratitude goes to all the Japanese T<sub>E</sub>X users, the organizing committee, the excellent lecturers and tutorial speakers, and all the participants.

My hope—and my feeling tells me I am not completely wrong—is that every participant could take home some great idea, some new knowledge, something that will improve, extend, diversify our T<sub>E</sub>X experience in the long run. For me personally, this was definitely the case.



The organizing committee sent the original poster for the conference to DEK, with this inscription.

## References

- [1] Blog—there and back again.  
<http://www.preining.info/blog/>.
- [2] Tokyo printing museum.  
<http://www.printing-museum.org/en/>.

◊ Norbert Preining  
Japan Advanced Institute of  
Science and Technology  
Nomi, Ishikawa, Japan  
norbert (at) preining dot info  
<http://www.preining.info>

**The incredible tale of the author who didn't want to do the publisher's job, but eventually had to because the publisher didn't have a clue about typesetting, although to be honest, the author did some stupid things as well, but fortunately, everything is all right now, however this is an experience the author wouldn't wish upon anyone, but is still going to narrate it for your greatest enjoyment, and will do so both crying and laughing (yes, that was the title)**

Didier Verna

## Abstract

In this article, I relate a recent experience of mine: writing a book chapter for a publisher who doesn't have a clue about typesetting. I confess my futile attempt at using  $\text{\TeX}$  for writing the chapter in question. I describe the hell that descended upon me for daring to do that. I however admit that the hell in question would have been even greater, had I not done so. This article is both a nervous breakdown and a peal of laughter, and I am seeking for the reader's comfort.

## 1 Prologue

The story began on May 07, 2011, when I received an invitation to author a chapter for a particular Computer Science book. Here is an excerpt from the email I received:

*The objective of the book is to provide original academic work about current research ... a comprehensive overview ... comprehensive material ... provide new results and answers to some open problems ... indispensable for researchers, professionals and practitioners as well as for educators who would like to have a comprehensive ..., useful resource for graduate and undergraduate level courses.*

Well, a book containing practically everything, and targeted at practically everyone. That sounded like quite a challenge, and I decided to accept it. I sent a proposal on June 14.

On July 20, I received an email informing me of the acceptance of my proposal. The message contained a URL pointing to a chapter template, and two attachments with some "chapter organization and formatting guidelines" and some "details to keep in mind". All of this sounded fine, but my ears were already starting to tickle me: why a chapter template, why guidelines, why in two different attachments? Don't they provide a style or a class file?

I looked more closely, and that is when I discovered that the provided chapter template was in fact a Word document, and that the two attachments were in docx format.

Then, I started to worry.



On top of that, I had several questions that were not answered in the documents I received, so I asked them by email to the editor (my only contact then) on July 21. What about copyright assignment? Do I get a contract of some sort? How is the book going to be published: online, on paper? Will it be available for sale or for free? With an open source license? Do I get royalties on the sales ... etc. The next day, I got the following response.

*My responsibility is out of scope of publishing.  
At this point I don't have the answers to your  
questions. I need to contact the publisher.*

Well, thank you very much, because you see, these are important issues nevertheless ...

## 2 Submission

After thinking about it for some (very short) time, I decided to be clever. I sent my initial submission on October 4, in PDF format. I explained to the editor that I had written the chapter in  $\text{\LaTeX}$  because there was no way I could work with Word. I also promised that I would convert it to Word for the final version, in order to comply with the publisher's requirements.

The editor had no problem with that, and then followed a reviewing period (contents only!) until the final acceptance of the chapter (contents only!) on March 10, 2012.

Let's be clever? Not so much ... In terms of PDF to Word conversion, I had spotted at least a dozen websites offering this service for free before, so I thought it was going to be easy. That's why, in all my self-sufficiency, I waited until the last minute before taking care of that. First, I had some minor adjustments to do.

**Unnumbered sections** No, the publisher doesn't want section numbers. Don't ask. Easy enough to do in  $\text{\LaTeX}$ , though. One just needs to use starred sectioning commands. Oh, but on second thought, what happens to all those nicely `varioref`-formatted section references? Gone is the answer, I think. So I had to get rid of those, and find a textual workaround to point the reader to other parts of the chapter, without numbers. Very convenient indeed. Dammit.

**No figures (go figure)** Yes, the publisher wants the figures in a separate zip file, not in the chapter itself. Don't ask. Surely, they have implemented

Calcagno, C., Taha, W., Huang, L., **and** Leroy, X. (2003).  
 Calcagno, C., Taha, W., Huang, L., & Leroy, X. (2003).

Manipulation (PEPM), pages 95-99. ACM SIGPLAN.  
 Manipulation (PEPM) (pp. 95-99). ACM SIGPLAN.

Symbolic Computation, 13(1-2):51-55.  
 Symbolic Computation, 13(1-2), 51-55.

**Figure 1:** Bibliographic format divergences

`unzip` and `TEX`'s float placement algorithm with Word macros. Anyway, easy to do in my `LATEX` source file, but then again, I had to rewrite all the figure references manually.

So I eventually recompiled my PDF with those adjustments, and started using a PDF to Word conversion service. At the last minute (did I say so, already?).



At that moment, all hell descended upon me.

**Big mistake #1** The conversion didn't work very well, and that is a euphemism. Basically, the font size was ok, and everything else was lost: font families, shapes, verbatim and code formatting. Everything. And because I did that at the last minute (did I say so, already?), I decided that it would be safer for me to spend the last night restoring all that was lost *by hand* directly in Word, rather than trying to look for yet another half-baked service in the naïve hope that it would work better. So I took my little mouse with me and went through all 30 pages, clicking like crazy. Welcome to the world of WYSIWYG.

Here, I must confess that in the heat of the moment, another simpler and totally obvious alternative didn't even occur to me. Months later (in fact, when I was preparing the slides for TUG 2013), I realized that I could have opened the PDF file resulting from the `LATEX` compilation, selected the whole contents, and just cut and pasted it into Word. I tried that and it actually went *better* than any of the online conversion services I tried. Shame on me ...

**Big mistake #2** The second big mistake I made was to *not* use the exact bibliographic style the publisher wanted. More precisely, the author instructions mentioned "strict APA" conformance, and I simply used `apalike`. In the end, there were subtle differences in the formatted bibliographic output, some of them illustrated in figure 1. I also had some errors in the `.bib` files which led to incorrect sorting and other oddities.

Here, the obvious choice would have been to go fix the `.bst` file in order to produce the desired out-

put, and regenerate the PDF. But do you remember that I had already spent hours fixing *Big mistake #1* in Word? I simply couldn't bear the very idea of having done so for nothing, so I decided to go fix all those details *by hand*, in the Word document.

### 3 Interlude

I am a Boduka. I practice martial techniques in order to reach ultimate self-control, peace and harmony with the Universe. Let's breathe. Deeply. Okay.

In the meantime, I had my first contact with people from the publishing company (remember that until now, my only contact was with the book editor). On August 28, I received an email from the publisher, part of which is transcribed below.

*From: marketeer #1*

*Greetings! . . . personally thank you for your excellent contribution! lots of marketing crap/ Your development editor, marketeer #2, very much enjoyed working with you, and now as your marketing representative, I look forward to assisting you with your promotional efforts.*

First of all, I am delighted to learn that *marketeer #2* "very much enjoyed working with me", although I feel the urge to mention that I had never heard of this person before. Next, let me see if I understand this correctly: I am writing a book chapter for which I am most likely *not* going to be paid, and in their infinite generosity, the publisher is kindly offering to assist *me* in promoting the book that *they* are going to sell?

But wait. There's more.

*I have also created an Exclusive Discount Offer form. This form allows you to order one or more copies with our exclusive author discount.*

So, I am writing a book chapter *for free*. I have to do the promotion of the book *myself*, and in their infinite generosity, the publisher is kindly offering a discount for me to actually *buy* the book I'm contributing to write??

But wait. There's more.

*You will hear from us again regarding how to access a complimentary PDF copy of your individual chapter in the book.*

Wow wow wow. Hold it right there. I am writing a book chapter *for free*. I have to do the promotion of the book *myself*. I will have to *buy* the book I'm contributing to write, and in their infinite generosity,

the publisher is kindly offering me a “complimentary” PDF of *my own chapter*, which I wrote *myself*, and for which, obviously, I already have a PDF, and what’s more, of a much better quality since it has been generated with `pdflatex`???

In fact, the situation was not so bad as it seemed in the first place. 3 clauses in the copyright assignment form I eventually received contained implicit answers to some of my original questions.

*2. Author(s) understand that no royalties or remuneration will be paid by the Publisher to the author for the above named submitted manuscript. Further, Author(s) acknowledge the manuscript is being provided on a volunteer basis for the professional recognition obtained by the publication.*

Read: I work for the glory.

*5. The Publisher will have the right to edit the work for the original edition and for any revision, provided that the meaning of the text is not materially altered.*

We will see later on how a publisher who doesn’t have a clue about typesetting can actually alter the meaning of the text without even realizing it ...

*6. The Publisher will furnish 1 copy of the book to the lead Author of each chapter without charge. The coauthor(s) of the manuscript will receive a copy of the manuscript along with a copy of the title page of the book. Copies of the book for the author’s/co-author’s use may be purchased at a 40% discount from the list price.*

So after all, I *will* get a free copy of the book. And thank God I wrote my chapter alone, because I would have hated to have to explain to my co-authors that they had earned the right to *buy* the book ...

#### 4 Proofreading

After all this agitation, I honestly thought that the worst was finally behind me. Little did I know ...

On September 11 (notice the date?) I got a new message from yet another person at the publisher’s, inviting me to proofread the book (or at least, my own chapter):

*From: yet@another.guy  
I am very pleased to send you the proof of the book. Please copy and paste this link into your browser: http://.../EditorProof.pdf*

I haven’t mentioned yet the apparent publisher’s concern for security, requesting digitally signed copyright assignments as well as paper copies and so on. All of this for putting the editor-proof version of the book

Before:

```
(defclass face ()
  ((name :initarg :name)
   (bold :initarg :bold)))
```

After:

```
(defclass face ()
  ((name: initarg: name)
   (bold: initarg: bold)))
```

**Figure 2:** Removing spaces between identifiers

online and sending the url in the clear by email ... Anyway. The deadline for proofreading was 6 days later only, there was a list of items to specifically check for, and there was also the following comment in the message.

*Please do not be concerned with house style layout application, such as font type / size; title and subtitle styles; spacing and formatting*

For some reason, this smelled very bad to me, and I decided to pay a very special attention to those points. I was quite right (but that, you guessed)!

On September 17, I finished proofreading my chapter, and sent the following message.

*To: yet@another.guy  
There are many things that have gone wrong in my chapter. Some of them may belong to your “do not be concerned with house style” category, but they are so worrying that I need to mention them anyway.*

Then, I started enumerating, by decreasing DEF-CON level, all the things that had gone wrong in my chapter, between my Word version and their PDF.

**Figures swapped** My chapter contained exactly two figures. They managed to swap them. Figure 1 was referencing figure 2 and vice-versa. That’s what you get when you don’t let your typesetting software automate the referencing (let alone the placement).

**Code excerpts** Originally, I had a nice layout for my code examples, automated with `lstlisting`. They completely messed up all of them. Worse: probably by editing the code *manually*.

First, they “conveniently” removed all spaces appearing before colons, as shown in figure 2, hereby concatenating all consecutive tokens of code. Remember clause #5 in the copyright assignment?

*The Publisher will have the right to edit the work [...] provided that the meaning of the text is not materially altered.*

Well, there you go!

## INTRODUCTION

Domain-specific language (DSL) design and implementation is inherently a transverse activity (Ghosh, 2010; Fowler, 2010). It usually requires from the product team knowledge and expertise in

**Figure 3:** Fancy spacing

Next, they “prettified” the double quote string delimiter character: `(show-keys :key2 "test")` became `(show-keys:key2 “test”)`. Cute, but not a string anymore.

Pretty much all code excerpts also had their indentation completely messed up, and I’m not even mentioning hyphenation (hint: what happens when you hyphenate a variable name, in a language which allows dashes in identifiers?).

Again, all of this is what happens when someone clueless about what a programming language actually is, starts editing code excerpts *by hand*.

**Inline quotes** The formatting of inline quotes, originally achieved with the `quote` or `quotation` environments was gone. Basically, all quotations were turned into mere paragraphs, and therefore indistinguishable from the surrounding text.

**Float positions and references** Along with the two figures I mentioned earlier, my code excerpts were all floats, placed automatically and referenced with `variorref`. All placements and references were destroyed, again, probably by manual editing. Here are just two examples of what I got:

*Blah blah blah . . . is given below in Box 10.*

Hint: the box below is Box 9.

*Which we can use like this as shown in Box 4:*

That sentence, which is not even a correct one (note the trailing colon), was standing alone as a whole paragraph in the middle of a page. The box was not even there.

**Spacing** Finally, there were spacing problems in almost every page of the chapter. Figure 3 illustrates this. I particularly enjoy the “readability” of the 4th line, and I guess this is what you get from a typesetting system which is clueless about aesthetics.

## 5 Epilogue

Facing all this mess, and without the ability to fix things myself, I ended up wasting countless hours carefully reviewing every single page, locating all the problems and noting them down for email reporting, which I eventually did. On September 21, I received the following message (note the sender).

*From: no-reply*

*I would like to take this opportunity to express our many thanks for your excellent contribution . . .*

*At the following link you will be able to access a printable copy of your final typeset chapter in PDF format . . .*

Unsurprisingly, many of the problems I had reported before were still unfixed, and new problems had appeared. I performed yet another careful review of the chapter, and boldly worked around the `no-reply` individual, sending my new report to *all* of my previous contacts at the publisher’s. I didn’t get any response, but by the looks of the actual book I received, most of the remaining problems were indeed fixed.

Several months later, when the book was printed, I received this final message (note the sender again).

*From: no-reply*

*Again, thank you for your outstanding contribution, and we look forward to working with you on another project.*

Well, maybe not! ☺

## 6 Acknowledgment

As far as I could see in the editor-proof version of the book, most problems I encountered with my chapter (in code formatting notably) also affected the other ones, which, I guess, is not really surprising.

In all this marvelous adventure, I wish to express my gratitude to the *editor* of the book, who invited me to write a chapter for it in the first place, and who contributed greatly to the holly hunt for typos and formatting mistakes, not in one chapter, but in the whole book.

- ◊ Didier Verna  
EPITA / LRDE  
14-16 rue Voltaire  
94276 Le Kremlin-Bicêtre Cedex  
France  
didier (at) lrde dot epita dot fr  
<http://www.lrde.epita.fr/~didier>

## How to make a product catalogue that doesn't look like a dissertation

Jason Lewis

### Abstract

Needing a robust way to produce a catalogue from a product database, Adobe InDesign, DocBook and L<sup>A</sup>T<sub>E</sub>X were evaluated. L<sup>A</sup>T<sub>E</sub>X was favoured due to its layout flexibility and non-proprietary nature. The challenge was to query the database and produce suitable L<sup>A</sup>T<sub>E</sub>X for the catalogue.

Native L<sup>A</sup>T<sub>E</sub>X was unable to query the database, so templating tools were investigated. Template Toolkit (TT) was chosen over PHP, favouring its Perl roots and broader applicability. Using TT and DBI for querying the database a dynamically generated L<sup>A</sup>T<sub>E</sub>X document can be quickly constructed. TT filters are developed to check and correct user supplied content for constructs that would cause the L<sup>A</sup>T<sub>E</sub>X compiler to fail. Filters are also used to sanitise Windows file paths for use in L<sup>A</sup>T<sub>E</sub>X.

Styling the document to look like a product catalogue was achieved using sans serif fonts, `flowfram` thumb tabs, colourful chapter and section headings developed using `tikz`, long tables that allow page breaks, alternating row colours, wrapping of paragraph text around images and the highlighting of new products. Full page PDFs are included in the document for the covers, front matter and adverts.

The result is system to generate a product catalogue, quickly and easily directly from our database using free and open source tools. This has reduced the workload in producing a catalogue and increased staff productivity and efficiency.

### 1 Introduction

I produce an 80 page full colour product catalogue in L<sup>A</sup>T<sub>E</sub>X. This paper outlines the tools I used to implement the system, link it to our database and style the catalogue for printing.

We print a new catalogue every six months and it has approximately 1000 products, 16 full page colour adverts and takes less than two minutes to build from the command line.

### 2 Why did I write this?

I am part owner of a small wholesale distribution business in Australia. When we established the business in 2001, we had approximately 40 products in our catalogue, for which a one page price list and order form in Microsoft Excel sufficed.

As we grew the business and added more products to our portfolio, the catalogue became unwieldy

to produce in this way. So in 2004 I set out to create a more robust system to produce it.

My goals were to create a system that would produce a product catalogue automatically from our database, wouldn't require long and detailed proof reading to ensure pricing was correct, be something I could delegate to staff for catalogue production, and that the staff need not have any special technical knowledge to use the system.

### 2.1 Tools I looked at

First, InDesign:

- InDesign at the time had limited scripting ability. This has improved now and you can write scripts in Microsoft Visual Basic or JavaScript.
- no proper database connectivity at the time. There are now numerous commercial tools that allow you to link InDesign to a database.
- proprietary software (I prefer to use free tools if possible).

Second, DocBook [1]:

- limited formatting and layout capabilities
- recommends using L<sup>A</sup>T<sub>E</sub>X for advanced layouts
- no database connectivity
- plain text, easy to script using a template tool

Third, L<sup>A</sup>T<sub>E</sub>X:

- flexible layout capabilities
- free and open source
- no database connectivity
- plain text, easy to script using a template tool

### 3 The challenge

#### 3.1 How to script L<sup>A</sup>T<sub>E</sub>X?

I could have written the system in native L<sup>A</sup>T<sub>E</sub>X but L<sup>A</sup>T<sub>E</sub>X has no way to retrieve data from a database. The other option was to choose a tool like PHP or Template Toolkit (TT).

#### 3.2 PHP

**Pros:** PHP is widely used.

**Cons:** geared towards HTML/web; it's PHP; it's not Perl.

#### 3.3 Template Toolkit

**Pros:**

- Generalised to templating anything
- not PHP
- written in Perl
- `Template::Plugin::Latex` can output PDF directly; can build manually for demonstration.

Astute readers might notice a slight bias towards Perl. This was mainly due to having prior knowl-

edge of Perl and meant I didn't have to learn a new programming language.

### 3.4 Database driven documents

Template Toolkit uses Perl's DBI for database access, which means it can retrieve data from just about anything. Our data is stored in Microsoft SQL and Microsoft Access. I use DBI Proxy [2] to connect to the MS Access database.

### 3.5 Compiling reliably

Building the catalogue from source is a two step process: running Template Toolkit, and then passing its output through `pdflatex`. I created a Makefile to do this, but often the document would require multiple runs to ensure all the cross-references were correct. A tool such as `lATEXmk` [3] or `rubber` [4] helps with this by providing a single command that will repeatedly run `pdflatex` until all cross-references are stable.

### 3.6 User interface for entering data to the catalogue

I needed a user interface for data entry. I chose MS Access as being easy to script, it was a familiar user interface for the staff, and I already owned it. The downside is that of course it is a proprietary tool.

## 4 Template Toolkit Primer

Here's how I scripted L<sup>A</sup>T<sub>E</sub>X using Template Toolkit.

### 4.1 Install Template Toolkit

Cpanminus [5] is a great tool to retrieve, unpack, build and install Perl modules from CPAN:

```
cpanm Template
cpanm DBD::CSV
cpanm DBI
cpanm Template::Plugin::DBI
```

Now we have Template Toolkit, DBI, and some ancillary modules installed, and can try writing a simple Hello World.

### 4.2 Hello World

`tpage` is a simple script supplied by TT that parses a template file supplied on the command line and outputs it to standard output. Given this two-line template file:

```
[% str = 'Hello TUG2013' -%]
[% str %]
```

We can parse and output it like this:

```
$ tpage helloworld.tt
Hello TUG2013
```

Anything within [% %] will be treated as TT code and executed. The minus sign before the closing

delimiter above strips the final newline from that line upon output; similarly, a minus sign after an opening delimiter (as below) removes a leading (preceding) newline.

### 4.3 Read a CSV file

Let's suppose we have this table of data:

Table 1: `simple-example.csv`

FirstName	LastName	FavouriteNumber
Jason	Lewis	34
Joe	Blogs	2
Frank	Sinatra	88

Here is TT code to parse it, using DBI to access the file as a database in the current directory:

```
[%- USE db = DBI(
    database => "DBI:CSV:f_dir=." ) -%]
[%- FOREACH item = db.query(
    "SELECT * from simple-example.csv") -%]

FirstName: [% item.firstname %] \
LastName: [% item.lastname %] \
Favourite: [% item.favouritenumber %]
[% END %]
```

Note that the column headings of table 1 were sanitised to lower case by DBI::CSV.

### 4.4 Write your own parser

Well not quite, but Template Toolkit makes it easy to implement a TT parser:

```
#!/usr/bin/env perl
use Template;
die "no TT filename given" if (@ARGV != 1);
my $tt = Template->new({
    INCLUDE_PATH => '.',
    INTERPOLATE  => 1,
}) || die "$Template::ERROR\n";

my $input = $ARGV[0];

# process input template, substituting variables:
$tt->process($input, $vars)
|| die $template->error();
```

This creates a TT object and parses the file whose name is supplied on the command line. So far, this just provides the same functionality as `tpage`.

### 4.5 Build your L<sup>A</sup>T<sub>E</sub>X document

Here we write our L<sup>A</sup>T<sub>E</sub>X document, including the TT lines to read from the database.

```
[%- USE db = DBI( database =>
    "DBI:CSV:f_dir=." ) -%]
\documentclass{article}
```

```
\usepackage[utf8]{inputenc}

\title{build-document example}
\author{Jason Lewis}
\date{October 2013}

\begin{document}
[%- FOREACH item = db.query(
    "SELECT * from simple-example.csv") -%]
First Name: [% item.firstname %]
\\ Last Name: [% item.lastname %]
\\ Favourite Number: [% item.favouritenumber %]
\\ \newline
[% END #FOREACH -%]
\end{document}
```

**Output** (where mypage is the script we just saw):

```
$ ./mypage build-document.tt
\documentclass{article}
\usepackage[utf8]{inputenc}

\title{build-document example}
\author{Jason Lewis}
\date{October 2013}

\begin{document}
First Name: Jason \\ Last Name: Lewis
\\ Favourite Number: 34 \\ \newline
First Name: Joe \\ Last Name: Blogs
\\ Favourite Number: 2 \\ \newline
First Name: Frank \\ Last Name: Sinatra
\\ Favourite Number: 88 \\ \newline
\end{document}
```

## 5 Technical problems

### 5.1 Escaping special L<sup>A</sup>T<sub>E</sub>X characters

The text in our database is generated by users, and often cut and pasted from Microsoft Word. It typically contains characters that won't be natively typeset by L<sup>A</sup>T<sub>E</sub>X and worse still, cause the L<sup>A</sup>T<sub>E</sub>X build to hang or fail.

Therefore all text from the database needs to be sanitised before being passed to L<sup>A</sup>T<sub>E</sub>X. My solution was to write a TT filter. Below is a sample function that takes a string as its input, and ensures any dollar signs within the string are escaped by prepending a backslash.

### 5.2 Sanitise for L<sup>A</sup>T<sub>E</sub>X

```
sub latex_filter {
    my $return = $_[0];

    # escape $ with a backslash for LaTeX
    $return =~ s/(\$)/\\\$1/g;

    return $return;
}
```

```
my $tt = Template->new({
    FILTERS  => {latex_filter => \&latex_filter},
    INCLUDE_PATH => '..',
    INTERPOLATE  => 1,
}) || die "$Template::ERROR\n";
```

### 5.2.1 More things that need to be filtered

Then adding more filters is simply a matter of adding a regular expression to search for it and replace it with whatever is appropriate. Some examples.

#### Degree symbol:

```
$return =~ s/°/\degree /g;
```

#### Accented characters:

Transform é into '\{e}:

```
$return =~ s/é/\'{e}/g;
```

#### Incorrect quotes:

Convert beginning-of-line right or double quotes to left quotes, and use ASCII apostrophes.

```
$return =~ s/(^|\s)'(.*)/$1‘$2/g;
$return =~ s/(^|\s)"(.*)/$1“$2/g;
# replace Windows quote (octal 0222) with ASCII ‘
$return =~ s/\\222’/‘/g;
```

#### En dash between numbers:

Convert minus sign to an en-dash by searching for single minus sign between numbers and replacing it with two minus signs: 1-10 vs. 1--10.

```
$return =~ s/(\d+)-(\d+)/$1--$2/g;
```

## 5.3 Pass through L<sup>A</sup>T<sub>E</sub>X commands

From time to time I needed a way to pass L<sup>A</sup>T<sub>E</sub>X commands through the filter. I chose an escape of <latext> which in hindsight may not have been the best choice, as it looks too much like XML. We just replace it with a backslash:

```
$return =~ s/<latext>/\\>/g;
```

For example, <latext>latext becomes \latext.

## 5.4 Calling a filter from within a template

Once a filter has been defined, you can call it with the pipe ‘|’ character.

```
[% str = "$100 is 50% of $200" -%]
Raw string is [% str %]
Filtered string is:
[% "$100 is 50% of $200" | latex_filter %]
```

Which results in:

```
Raw string is "$100 is 50% of $200"
Filtered string is:
"\$100 is 50%\ of \$200"
```

### 5.4.1 L<sup>A</sup>T<sub>E</sub>X does not like Windows paths

On our network, product images are stored on a server drive accessed via a Windows share w:\. In MS Access, users select an image to go with a product

range and the path to the image is stored in the database as a Windows path:

```
w:\some\path to\an image.jpg
```

Users often use spaces, commas, apostrophes and other abominable characters in the image paths, and `\includegraphics` does not handle paths with spaces in them.

The solution was to create a L<sup>A</sup>T<sub>E</sub>X-friendly symbolic link to the file and include that instead. This was easy to achieve by making another TT filter.

#### 5.4.2 Convert Windows path to Unix

The goal is to convert a Windows path such as  
`w:\Alive & Radiant\2013-July-product.jpg`  
 to:

```
_mnt_Alive_&_Radiant_2013-July-product.jpg
```

#### 5.4.3 Build a filter

Here is my Perl code:

```
# convert all \ to / e.g. c:\ to c:/
$return =~ s/(\\)/\\/g;
# strip drive letter c:/path/file to path/file
$return =~ s/^w:\\///g;
# strip extension: /filename.jpg to /filename
#$return =~ s/\.\w\w\$//g;
# clean up the path
$return = File::Spec::Unix->canonpath($return);
```

#### 5.4.4 Make the symbolic link

Find spaces, / or % in filenames and replace with underscores. Then use the resulting string as the name for a symbolic link to the original file. Use the symbolic link name in the L<sup>A</sup>T<sub>E</sub>X document.

```
# replace spaces, slashes, percents with _
$safe_filename =~ s/[\s\\%]/_/g;
# make a symbolic link to the file
symlink($return, $safe_filename) || die;
```

#### 5.4.5 Use new filter on image paths

Use the new filter on image paths as they are retrieved from the database:

```
[% ImagePath
= item.CategoryImagePath | path_filter %]
[% IF ImagePath != "" %]
  \includegraphics[width=12cm,
                  height=\imageheight,
                  keepaspectratio=true]
  {[%ImagePath%]}
[% END # if image exists %]
```

#### 5.5 User interface

I needed to develop a user interface for staff to manage data in the catalogues. I chose MS Access as it was easy to create and modify. I wanted to create

something that would shield the users from having to know L<sup>A</sup>T<sub>E</sub>X in order to create and edit content for the catalogue.

### 6 Make L<sup>A</sup>T<sub>E</sub>X output look like a product catalogue

Clearly I had to style the document so it would look more like a product catalogue and less like a dissertation.

#### 6.1 Sans serif fonts

The first thing I did was use a sans serif font. I chose Helvetica.

```
\usepackage{helvet}
% set the font to helvetica for body text
\renewcommand{\familydefault}{\sfdefault}
```

This provides URW Nimbus Sans which is a free clone [6] of Helvetica.

#### 6.2 Thumb tabs

Thumb-tabs are a nice feature for any catalogue. I created them using Nicola Talbot's `flowfram` [7] package, like this:

```
\setthumtab{1}{backcolor=[rgb]{0.15,0.15,1}}
\setthumtab{2}{backcolor=[rgb]{0.2,0.2,1}}
\makethumtabs[50mm]{30mm}
```

```
\begin{document}
  \tableofcontents
  \thumtabindex
  \enablethumtabs
  \chapter{1ABC}
  \Blindtext
  \chapter{2ABC}
  \Blindtext
\end{document}
```

#### 6.3 Colourful chapter and section headings

Colourful chapter and section headings were made using the `tikz` package.

```
\newcommand\colorchapter[1]
  {\def\chapterbg{#1}\chapter}
% begin CHAPTER format
\newcommand\boxedchapter[1]{%
  \begin{tikzpicture}[inner sep=0pt,
                     inner ysep=1.3ex]
    % left position of text
    % right hand edge chapter title text
    \node[anchor=base west]
      at (3,0) (counter) {};
    \path let \p1 = (counter.base east)
      in node[anchor=base west,
              text width={\textwidth-(x1+26.33em)} (content)
              at ($(counter.base east)+(0.33em,0)$)
              {\textcolor{white}{%
                \Huge\sffamily\textsc{\thechapter \ \ #1}}};
```

```
\begin{pgfonlayer}{background}
\shade[left color=\chapterbg,
       right color=\chapterbg]
let \p1=(counter.north),\p2=(content.north)
in (0,100 + \maxof{\y1}{\y2})
rectangle (content.south east);
\end{pgfonlayer}
\end{tikzpicture}%
}

\titleformat{@@html:\@@chapter}%
{}%
{}%
{0pt}%
{\boxedchapter}%
\titlespacing*\{chapter}{-100pt}{*-20}{*-1}
% end CHAPTER format
```

## 6.4 Long tables

I needed a way to create tables that could break across page boundaries, but also be able to span more than one page, and possibly have a PDF (for an advert) embedded at a split. The `xtab` [8] package produced the best results for me; however, there are very many packages for tables. I found a good summary of table package features at <http://tex.stackexchange.com/a/12673>.

## 6.5 Alternating row colours in xtab

This was easy to do in TT: while looping through query results, simply change the row colour every two lines.

```
\begin{xtabular}{l}
[% i = 1 -%]
[% FOREACH item = db.query(
    "select * from $CatInfo.query") %]
[%- IF (i mod 4 == 3)
   || (i mod 4 == 0) -%]
  \rowcolor{[%-
    item.SectionRow2BGColour -%]}
[% ELSE -%]
  \rowcolor{[%-
    item.SectionRow1BGColour -%]}
[%- END #if -%]
[%- i = i+1 -%]
[% item.col1 %] &
[% item.col2 %] &
[% item.col3 %] \\ %row data
[% END; #FOREACH %]
\end{xtabular}
```

## 6.6 Wrap description around an image

I used `wrapfig` [9] to put text around an image.

```
[% IF ImagePath != "" %]
\begin{wrapfigure}{r}{0pt}
```

```
\includegraphics[width=12cm, \
height=\imageheight, \
keepaspectratio=true]\
{[%ImagePath%]}

\end{wrapfigure}
[% END # if image exists %]
```

## 6.7 Highlight new products

New products are highlighted by yellow text with a red background.

```
[%- IF item.NewProduct -%]
\scriptsize\colorbox{red}{%
\textcolor{yellow}{NEW}}
\sffamily\footnotesize{~ \
[%-item.description | latex_filter -%]} &
\footnotesize{[%-item.description | \
latex_filter -%]} &
[%- END -%]
```

## 6.8 Including full page PDFs

We include full page PDFs in the catalogue for the front matter, rear matter and adverts. All are supplied as PDFs and we just have to include them. The trick is to turn off scaling so the bleed and trim marks appear in the correct place.

```
[% IF String.length > 0 %]
\includepdf[noautoscale=true]{[%CoverPage%]}
[% ELSE # warn that file cannot be found %]
  Cover file [% CatInfo.CatalogueFrontPage %]
  appears to be missing : [% error.info %]
[% END %]
```

## 7 Conclusion

I set out to create a tool for generating a catalogue from our database. I was able to use free tools such as L<sup>A</sup>T<sub>E</sub>X and Template Toolkit to achieve this. The overall goal was achieved, saving time and money and allowing staff to be more productive.

## References

- [1] <http://www.docbook.org/docbook>
- [2] <http://search.cpan.org/dist/DBI/dbiproxy.PL>
- [3] <http://ctan.org/pkg/latextmk>
- [4] <http://launchpad.net/rubber>
- [5] <http://search.cpan.org/dist/App-cpanminus/bin/cpanm>
- [6] <http://www.tug.dk/FontCatalogue/helvetica>
- [7] <http://ctan.org/pkg/flowfram>
- [8] <http://ctan.org/pkg/xtab>
- [9] <http://ctan.org/pkg/wrapfig>

◇ Jason Lewis  
<http://organictrader.com.au>

---

## Project Fandol: GPL fonts for Chinese typesetting

Clerk Ma and Jie Su

### 1 Fonts in Chinese typography

The most important issue of typography is about fonts. Fonts in alphabetic languages only need hundreds of glyphs. But in CJK languages, we need many thousands of glyphs.

In written Chinese, 6,762 Hanzi (汉字) are required. Furthermore, traditional typography requires four basic styles: Song (宋), Kai (楷), Fang (仿), and Hei (黑). The Song style is used in text, and acts like the roman style in western language. The Kai style's role in Chinese typography is more like italics in western typography. The Fang style is used in section titles or authors' names in contemporary Chinese magazines. The Hei style is used for emphasis.

### 2 Project Fandol and Chinese fonts

There are quite a few free Chinese fonts in the open source community. In most GNU/Linux systems, people use WenQuanYi (文泉驿) [1] fonts to display CJK glyphs. The Android platform uses Droid Sans Fallback [2] for this. These fonts are designed for screen display, and not for high quality typesetting. Arphic Technology has released two TrueType fonts (in Kai and Song styles) to the open source community in 2010 [3]. But these two fonts are not sufficient for the tradition of Chinese typography.

In the Chinese TeX community, many people need a set of free and open source Chinese font to typeset theses, reports, etc. To fill this need, Project Fandol [4] was established in 2012, with the goal of designing a set of high quality Chinese fonts which is released under the GNU General Public License with the font exception [6]. The current members of the project are Clerk Ma and Jie Su.

So far, we have developed six CID-keyed OpenType fonts in four styles. A CID-keyed font is only suitable to a particular language; we use Adobe-GB1-5 [5] to organise our fonts. We released version 0.2 to CTAN in August 2013. Here is a list of the fonts:

- FandolSong-Regular: 我能吞下玻璃而不伤身体
- FandolSong-Bold: 我能吞下玻璃而不伤身体
- FandolHei-Regular: 我能吞下玻璃而不伤身体
- FandolHei-Bold: 我能吞下玻璃而不伤身体
- FandolFang-Regular: 我能吞下玻璃而不伤身体
- FandolKai-Regular: 我能吞下玻璃而不伤身体

The glyphs in the current version of Fandol fonts cover four main categories:

- Kana (169): あいうえ
- Bopomofo (38): ウタロヒ
- Hanzi (6,762): 天地玄黄
- Yi (1,132): オモロカ

The alphabetic part of Fandol fonts are merged from Computer Modern Unicode [7]. In general, matching a western font with a Chinese font is a difficult task. Our current matching is shown below:

FandolSong-Regular: CMU Serif Roman

FandolSong-Bold: CMU Serif Roman Bold

Nonextended

FandolHei-Regular: CMU Sans Serif

FandolHei-Bold: CMU Sans Serif Demi Condensed

FandolFang-Regular: CMU Concrete Roman

FandolKai-Regular: CMU Serif Upright Italic

### 3 Acknowledgements

We are grateful to these people for their help and suggestions: Mengdi Cao, Fuzhou Chen, Jinze Huang, Hu Gao, Cheng Wang, Yi Lu, Xincheng Luo, Xiaohao Xia, Cheng Yang, Yousong Zhou, Weiwen Zhang, Xiabing Wang, Liye Ding, Nan Ding, Zhaoli Wang, Haiyang Liu, Qing Li, Wei Sun, Rui Xie, Xuan Leng, Xiang Yu, Liming Qin.

### References

- [1] WenQuanYi Project, 文泉驿 – 开源中文计划  
<http://wenq.org/wqy2/index.cgi>
- [2] Ascender Corporation, The Droid font family, 2010  
<http://www.droidfonts.com/droidfonts/>
- [3] Arphic Technology, 文鼎科技发布新的公众授权字型  
<http://www.arphic.com/cn/news/2010/20100420.html>
- [4] Clerk Ma and Jie Su, Fandol—Four basic fonts for Chinese typesetting, 2013  
<http://ctan.org/pkg/fandol>
- [5] Ken Lunde, The Adobe-GB1-5 Character Collection, 2012  
<http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/font/pdfs/5079.Adobe-GB1-5.pdf>
- [6] GNU, How does the GPL apply to fonts?  
<http://www.gnu.org/licenses/gpl-faq.html#FontException>
- [7] Andrey V. Panov, Computer Modern Unicode fonts, 2013  
<http://canopus.iacp.dvo.ru/~panov/cm-unicode/>

◇ Clerk Ma and Jie Su  
clerkma (at) gmail dot com

## Tsukurimashou: A Japanese-language font meta-family

Matthew Skala

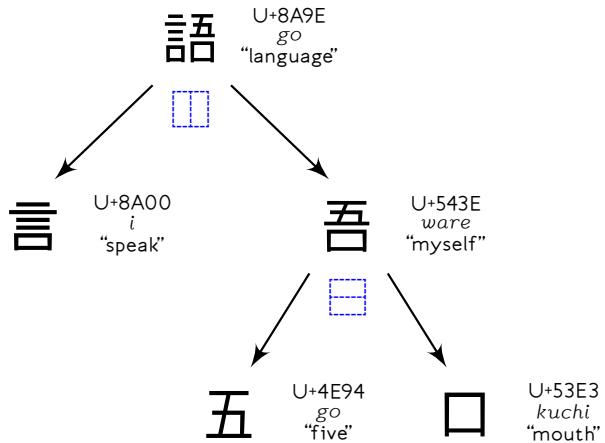
### Abstract

METAFONT-based font projects for the Chinese, Japanese, and Korean (CJK) languages have been announced every few years since the early 1980s, even predating the current form of the METAFONT language. Except for a few non-parameterized conversions of fonts that originated in other formats, in 30 years every METAFONT CJK font has been abandoned at or before the 8-bit barrier of 256 *kanji*, nowhere near the thousands required for practical typesetting. In this presentation I describe the first project to break that barrier: Tsukurimashou (<http://tsukurimashou.sourceforge.jp/>), currently at over 1500 *kanji* (as well as kana, Latin, and Korean hangul) and steadily growing. I discuss technical and human challenges facing this kind of project, how to solve them, and spin-off technologies such as the IDSGrep *kanji* structural query system.

### 1 Introduction

The Han script, used by the Chinese, Japanese, and Korean (CJK) languages among others, includes very many characters. Just counting them is tricky, but a human being might typically need to know a few thousand for basic literacy in a Han-script language. The list of 2136 characters taught in the Japanese school system (the *jouyou kanji*) is one benchmark, near the low end. Chinese requires more, and a typesetting system may require more still, because of rare characters found in names, historical contexts, and so on. A human being can get away with failing to read the occasional character; typesetting systems need to be able to print nearly all of them. Computer fonts considered usable for Japanese typically cover between six and twelve thousand characters. Databases of rare characters used in linguistic research cover tens or hundreds of thousands.

The sheer number of characters that go into a CJK font, and the amount of work implied by that number, is daunting. Considering the difficulty of building even a simple Latin font with METAFONT, it may be no surprise that there are no complete METAFONT-native CJK typefaces. But on the other hand, examination of Han-script text (even, or especially, by someone who cannot read it) quickly reveals that characters can be decomposed into smaller parts, as shown in Figure 1. Computer scientists who examine Figure 1 are likely to believe they understand it. “Of course,” one supposes, “the tens of thousands of Han



**Figure 1:** Breaking a character into its parts.

characters are just a small vocabulary of primitive shapes, perhaps only a few dozen, which combine in straightforward ways according to a spatial grammar to form tree structures!”

Computer scientists know how to deal with such things. It should be only the work of a week or two for a good programmer to lash together a prototype CJK font generator. Each primitive shape can be a subroutine; there can be other subroutines expressing the combining operations such as “place this one above that one”; a few parameters applied to the low-level shapes can allow for creating a wide range of styles; and the only real challenge is looking in the dictionary that lists the tree decompositions of all the characters. That book must exist in China, so we’ll get it by interlibrary loan. This project might even be easier than building a Latin font meta-family.

The earliest METAFONT CJK project I know of was LCCD, the Language for Chinese Character Design, described in a 1980 Stanford technical report by Tung Yun Mei [11]. Mei collaborated with Knuth and based LCCD on the METAFONT79 language developed to that point. Even in 1980, many of the ideas were already in place that a present-day computer scientist would naturally think of on viewing Figure 1. Mei’s report includes images of 346 “basic strokes and radicals”, and 112 completed characters.

Subsequent work on METAFONT-native CJK fonts includes that of Hobby and Guoan in 1984, who created 128 characters [5]; Hosek in 1989, character count unknown but two are displayed in the *TUGboat* article [6]; Yiu and Wong in 2003, in a project that targeted on-demand creation of rare characters rather than a font as such [16]; and Laguna circa 2005, with 130 characters in the last available version [10]. All these used a relatively small number of basic components, combining according to a spatial grammar to form more complicated characters.

I listed published METAFONT-related projects. Similar ideas have also been used behind closed doors in commercial font foundries (CDL from Wenlin Institute seems to be an example [15]), and non-METAFONT research projects like the LISP-based Wadalab toolkit [13]. The Wadalab font project ran during the 1990s; much of the work was lost or withdrawn, but some of its fonts survived to become widely used in the free software world. These kinds of projects use grammars of character parts, but lack the full parameterization that METAFONT users expect. There has also been work on using CJK fonts from other sources in TeX documents, sometimes including METAFONT incidentally in the workflow, but again without parameterization. For instance, the Poor Man’s Chinese and Japanese package [12] converts bitmap fonts into METAFONT code that renders scaled versions (without smoothing!) at arbitrary resolution.

It may be difficult to create fonts in METAFONT in general, regardless of the script; but human beings have done it. Several, though not many, METAFONT-native Latin fonts exist, and we can typeset a wide range of documents in Latin-script languages with parameterized METAFONT-native fonts. So after more than three decades of work, why are there no usable, parameterized, METAFONT-native CJK fonts at all?

## 2 Scaling issues

It is no coincidence that the past attempts to build CJK fonts in METAFONT have been abandoned at the same stage in development, around 120 characters. *That is the roughly the size of a Latin font.* METAFONT was designed to build fonts with sizes on that order, and thus METAFONT users have built expertise and developed tools for building fonts the size of Latin fonts. When fonts get larger, unforeseen difficulties show up like *nurikabe*—the plaster wall monsters of Japanese folklore blamed for delaying travellers by night.

### 2.1 Technical limitations

Many font file formats are limited to 256 glyphs by their use of 8-bit character codes. People who attempt to typeset CJK documents in classical TeX use elaborate workarounds involving slicing their fonts into 256-glyph sub-fonts. Handling the input encoding for documents written in large character sets with these slicing schemes is a tough problem too, but fortunately not one we as font designers must solve. There are extended versions of the TeX interpreter designed to use longer character codes directly (XeTeX is one), and those may also be able to work with font formats that store tens of thousands

of glyphs per file and don’t need to be sliced; but there is no similarly extended METAFONT to produce fonts in such formats.

Thousands of glyphs in a font does not just mean a bigger file. It also means more time spent compiling, and more memory consumption. One run of METAFONT may run out of memory or other resources trying to process an entire multi-thousand-glyph CJK font, and the user may run out of patience recompiling the whole thing after changing one glyph. To succeed at the thousand-glyph level, a project must have build tools allowing separate compilation of parts of the project. There should be tracking of dependencies among the different parts. Just being able to *find* pieces of code in a project this size—answering questions like “what was the name of the subroutine for such and such a shape?”—is an issue. These are elementary problems in software engineering, but there is little or no previous work on them in the METAFONT context because nobody has built systems this size in METAFONT before.

Classical METAFONT is designed to produce bitmap fonts, but bitmap fonts are no longer such a desired commodity. A present-day CJK font project will presumably target a vector format, but making METAFONT or some variation of it produce vector fonts requires additional layers of software, all of which are to some extent experimental. Bugs in the beyond-METAFONT software, previously undetected because previous fonts were smaller, will show up and need to be fixed. Keeping a handle on the bugs requires a test suite. The need for multiple steps in font compilation underscores the need for a capable build system. Human designers cannot be expected to issue five or six different commands in the right order to recompile every font, every time.

Earlier work on METAFONT CJK fonts has concentrated on writing code in METAFONT to draw the shapes of Han characters, as if that were the only problem to solve. Infrastructure that can scale to the size of the finished product is at least as significant.

### 2.2 Human factors

It is easy to underestimate how much work is involved in building a CJK font. We know how much work it is to design a Latin font. We know a CJK font has about 30 times as many glyphs. But it is easy to think, looking at Figure 1, that the CJK font should only be something like two or three times as much work as the Latin font (perhaps less), because so much code can be reused. In fact, less work is saved by code reuse than one might hope: every glyph requires some human attention. In computer science terms, font design is not much less than  $\Omega(n)$ .

Once it becomes clear that a human must spend time on every single glyph—it gets easier as more code exists to reuse, but there is no break point after which hundreds of characters will suddenly come for free—it is natural to hope that that human not be oneself. If we can just build a sufficiently good, easy to use set of tools, we can put them on the Web, maybe use a Wiki, and have many people in the community build a few glyphs each. Many hands make light work, once the infrastructure exists.

But to hope for someone else to build the actual glyphs after the tools are designed is to ignore a principal reason why people participate in free software projects in the first place. Designing tools for glyph construction is *fun*. Going through a list of 6000 glyphs one by one, doing simple repetitive tasks on each of them, is *work*. It is not easy to get volunteers for that sort of thing at the best of times, let alone when the volunteers must also have proficiency in an obscure programming language. The most successful large-scale collaboration is probably GlyphWiki [9], which sacrifices parameterization for a more purely graphical approach that demands less from the participants.

Finally, many of the potential rewards of a METAFONT CJK project, such as academic publications, can be had at the start, before the boring part; and then there are no more rewards until the end, and few then. You can publish one paper about your innovative techniques for building fonts; and you can publish one paper saying you have finished, years later. There is little in between. Knowing that this is the reward structure makes it tempting to write only the first paper and then start work on something else.

### 2.3 The script itself

The Han script itself may be the most ferocious *nurikabe*. Figure 1 with its clean decomposition of “language” into “speak”, “five”, and “mouth”, is deceptive. Many characters can be described as simply as that, but many others cannot. Consider Figures 2, 3, 4, and I could draw many more.

In Figure 2, “forest” is two copies of “tree” placed side by side. But the “tree” on the left is different from the “tree” on the right. If you make the two sides of “forest” look identical, readers will still know that you meant to write “forest”, but it will not look right. For a high-quality font, it has got to look right. This entails either creating two different primitives for the two trees, or having a smarter tree that knows how to change itself when it is on the left. Many character components change when they appear on the left. The modifications made

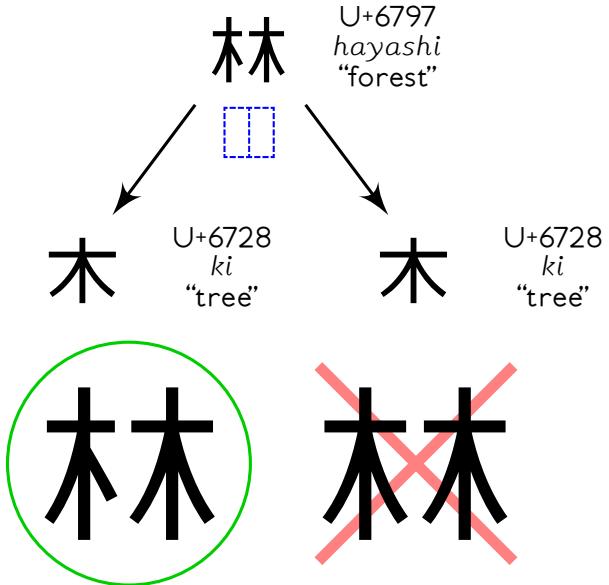


Figure 2: A forest is not two identical trees.

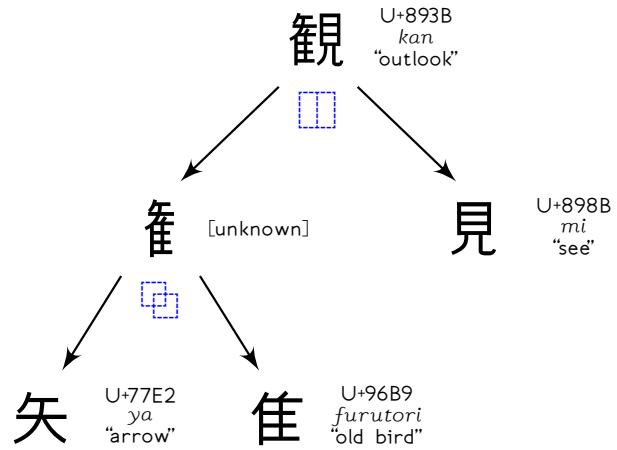


Figure 3: Combining operations can be complex.

when a component appears on the left are partially systematic, so we might hope to write code that can derive the left side shape automatically from the other shape, but it will not be simple, it will require manual supervision, and some projects have not gotten as far as noticing that it was an issue in the first place.

In Figure 3, the left side of “outlook”, in addition to not being a character in its own right, is some kind of hard to describe combination of “arrow” and “old bird”. It is not good enough to just print a scaled copy of “arrow” on top of “old bird” and hope for the best; getting it right requires modifying and deleting strokes in both parts. A generic overlap operation is unlikely to be flexible enough to do the right thing here. Every character that contains this sort of thing requires specific human attention to adjust it beyond

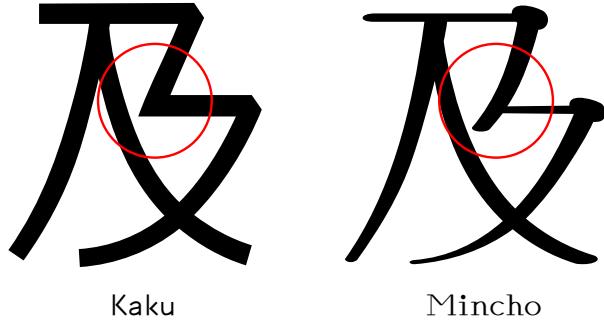


Figure 4: Two styles of U+53CA (*oyo*, “reach”).

just saying “overlap”. If the components change parametrically, then making sure they look right for all parameter values becomes even more complicated.

In Figure 4, two different styles of the same character are topologically different: the one on the left contains a single zigzag stroke that in the right-hand version is made up of two separate pieces. It is not easy to parameterize that in a way that will look good at every step in between, and if we make it a binary choice, giving up on the idea of interpolation, this difference will require some sort of “if” statement in the character description. A straightforward implementation of the grammar of shapes and combining operations suggested by Figure 1 would not provide for “if” statements.

These issues in the Han writing system point to an important conclusion: a simple grammar of parts and combining operations is not enough for building parametric fonts, even though it may be a useful starting point. Many characters can be decomposed into parts in the clean way implied by Figure 1, and such decompositions may be enough to support dictionary searches. It is easy to find enough well-behaved characters to put together a slide show or grant application, and to fool others or even oneself into thinking the whole character set will be easy.

But in order to produce high-quality fonts with full parameterization, with all the characters needed to typeset real documents, we must be able to override the simple descriptions and combinations of parts in arbitrarily complicated ways—per character and depending non-linearly on the parameters. To work at full scale, the font description language must have the power of a general-purpose programming language.

### 3 Tsukurimashou

My own attempt at building a METAFONT CJK font family is called the Tsukurimashou Project. The name means “Let’s make something!”; it is an *anime*

reference. As of version 0.8, released 26 August 2013, Tsukurimashou covers 1502 Japanese *kanji* (Han script) characters including all those taught in Japanese schools through Grade Four, as well as essentially complete coverage of *kana* (Japanese phonetic script), Latin, *hangul* (Korean alphabetic script), punctuation, and some miscellaneous ornaments and graphical characters. This is the work of one person, on a hobby basis while doing other things full-time for pay, since late 2010. It remains far from being a complete font family usable for typesetting general documents in Japanese, but it is already far past the point reached by any previous parameterized METAFONT-native CJK font project, and I believe my project is the first with a credible prospect of eventually reaching complete coverage.

Here are some points of reference distinguishing Tsukurimashou from other projects already discussed:

- Tsukurimashou is a parameterized meta-family, not a single font or a collection of independent fonts.
- Tsukurimashou is a font project, not primarily a dictionary of characters.
- Tsukurimashou is code, not data.
- Tsukurimashou is intended to achieve full coverage, at least of the characters needed for basic literacy in Japanese; it is not a proof of concept.
- Tsukurimashou is one person’s non-commercial project; not a for-profit corporate or large-scale collaborative effort.

Tsukurimashou is hosted as a free software project on SourceForge Japan, with the bilingual project home page at <http://tsukurimashou.sourceforge.jp/> featuring downloadable packages, a Subversion repository for the source code, a bug tracker, mailing list, and so on. The package as a whole is distributed under the GNU General Public License, version 3, with a clarifying paragraph added to explicitly permit embedding the fonts in documents.

#### 3.1 Motivation

The issues of human labour described in the previous section make it difficult for a CJK METAFONT project to reach complete coverage. Tsukurimashou’s solution to the amount of work involved in font design is to redefine that large amount of work as the *main goal* of the project instead of *an unfortunate cost* of the project. This point alone seems to be largely responsible for Tsukurimashou’s success to date.

I want to learn to read Japanese. Learning to read entails spending some time practicing and studying every character. But just studying a book and tracing copies on paper, as well as being boring, is

not a particularly effective way to learn. I would also like to become skilled at using METAFONT and related font technologies. I believe I acquire skills best by completing tasks that require the skills. Designing a font family for Japanese, as a project that requires knowledge of the *kanji* and of METAFONT, including concentration on every character in turn, is a good way to acquire that knowledge. And from that point of view, the actual finished fonts are not even important. The fonts are my excuse for spending time thinking about every character, which is the real goal. With that goal in mind, *avoiding* human attention to every character stops being necessary or even desirable.

Of course, the project may have desirable side effects. Work on Tsukurimashou has required me to invent new technology that may be useful in other projects. Some of it is publishable research in computer science, certainly welcome for someone hoping to establish an academic career. And because it places heavy (in some cases unprecedented) demands on other free software systems, Tsukurimashou has proven useful in the development of those systems. Given that I am already committing to spend some time per character on learning the language, the hope is to make that time pay off in as many ways as possible.

### 3.2 A brief tour of the fonts

Tsukurimashou as a software package generates OpenType font files as its main output. Those are intended for use in general typesetting and word processing, not only within the TeX world. I most often use them with XeTeX. The OpenType fonts are divided up into families, of which the main supported ones are named Tsukurimashou, TsuIta, and Jieubsida; then there is parameterization within each family for overall style, boldness, and monospace or proportional spacing. The main supported styles for the Tsukurimashou family are “Kaku” (a traditional sans-serif style), “Maru” (sans-serif with rounded stroke ends), “Mincho” (a less traditional version of the common Mincho serif style), and “Bokukko” (which somewhat resembles handwriting with a felt-tipped pen). Finer-grained parameters are used internally and could be made visible by modifying the code, much in the way that Computer Modern has internal parameters like “*stem\_corr*” as well as preset styles like “*Roman*”. Figure 5 shows a sample of the font styles; Figure 6 shows more of the Japanese characters in the Mincho style. Version 0.8 with all options enabled will build a total of 120 OpenType files, including some that are experimental and not intended for actual use.

These are outline fonts intended for printing at

### Tsukurimashou Font Meta-Family

さてさて、何が出来るかな？

Kaku 角	Extra Light 白字
Mincho 明朝	Light 軽字
Maru 丸	Normal 本
Bokukko 僕女	Demibold 半太字
Monospace	<b>Bold 太字</b>
Proportional	<b>Extra Bold 黒字</b>
<i>TsuIta Atama PS</i>	ツイタ頭 PS
<i>TsuIta Soku PS</i>	ツイタ足 PS
Jieubsida 지읍시다	Dodum 둘움
Batang 바탕	Sun-Moon 신문

Figure 5: A sample of the Tsukurimashou meta-family of fonts.

わらやまはなたさかあ ワラヤマハナタサカア  
あり みひにちしきい キリ ミヒニチシキイ  
るゆむふぬつすくうん ルユムフヌツスクウン  
ゑれ めへねてせけえ カレ メヘネテセケエ  
をろよもほのとそこお フロヨモホノトソコオ  
一七三上下中九二五人体先入八六円出力十千  
口右名四土夕大天女子字学小山川左年手文日  
早月木本村林校森正氣水火犬玉王生田男町白  
百目石空立竹糸耳花草虫見貝赤足車金雨青音

Figure 6: Kana and Grade One *kanji* in Tsukurimashou Mincho.

high resolution. They contain hinting for bitmap conversion, but it is done automatically and not expected to be extremely high quality. Japanese-language typesetting has traditionally used monospace metrics, simple scaling (i.e., no corrections for optical weight), and no slanting or italicization; Tsukurimashou currently offers a choice between monospace or proportional, no optical weight features, and italics for the Latin script only.

Although the largest use of Tsukurimashou fonts to date has been for typesetting the project’s own documentation in English, the design of the Tsukurimashou Latin glyphs, especially in the Mincho style, is intended primarily for setting the short fragments of English that sometimes occur in Japanese text. Tsukurimashou Mincho used for pure English text ends up looking like a display face and might not

be appropriate for entire sentences and paragraphs. Tsukurimashou Kaku is more suitable for extended settings in English.

The Jieubsida<sup>1</sup> family is intended to support the Korean *hangul* (alphabetic) script. *Hanja* (the Korean equivalent of *kanji*) are not included. This character set is relatively orthogonal: the main sequence of 11172 glyphs is algorithmically generated from a few tens of basic parts, though many less common letters had to be defined with more human intervention. Work on these fonts has proven useful in debugging the infrastructure at full scale, given that the Tsukurimashou series of fonts will eventually grow to a significant fraction of the size already reached by the Jieubsida series.

Beyond the main Tsukurimashou package, there are several smaller software packages called “parasites”, which appear in subdirectories of the distribution or may be detached. Some of these are font packages that share some of the Tsukurimashou infrastructure without really being part of the same meta-family; others are related software of other kinds. The only one discussed here will be the IDSgrep structural query system.

### 3.3 The infrastructure

Tsukurimashou’s infrastructure is designed like a typical free software project. It has source code that compiles into binary files, it has build scripts to accomplish that, and a would-be user can download a tarball, unpack it, and type `./configure` and `make`.

The build system is based on GNU Autotools. Choosing which source code files are needed for which font styles involves doing some logical inference that would not be convenient to do in a Makefile, so the Makefiles invoke additional code written in a subset of Prolog to evaluate the style selections, then run Perl scripts that scan the METAFONT sources to look for dependencies. The results of that computation are written into additional Makefiles, which guide the actual compilation process.

Knuth’s METAFONT creates bitmap fonts, while Tsukurimashou’s target is OpenType outline fonts. There are several METAFONT variants that can produce outline output from METAFONT source. I chose MetaType 1 [7] for Tsukurimashou. This package originates with the Polish TEX users group GUST and may be most famous for its use in the Latin Modern project [8]. It consists primarily of a macro package for MetaPost and a postprocessing script for GNU `awk`. One run of MetaPost generates the glyphs

---

<sup>1</sup> Intended as a translation to Korean of the name “Tsukurimashou”, but I am informed that “Mandeubsida” would be a better translation, and am considering changing it.

of a font as EPS files; another generates metrics; then the `gawk` script merges those and does some rewriting of the PostScript code to turn them into a single PostScript Type 1 font.

In recent versions, Tsukurimashou’s version of MetaType 1 has diverged somewhat from the one distributed by GUST. I started with the (very old) `mtype13` distribution, tried to upgrade it to use the latest MetaType 1 scripts, and ended up rewriting large sections of code. Many features of MetaType 1 are not used in Tsukurimashou (for instance, hinting; the “metrics” pass; and the entire processing chain in the reverse direction from PostScript back to METAFONT), and it proved useful to remove them, streamlining the code considerably. The core flow of information through Tsukurimashou’s version of MetaType 1 remains similar to that of the original, however: the MetaPost interpreter executes code in the METAFONT language, writing one EPS file for each glyph, and then those are postprocessed into PostScript Type 1 fonts.

Each PostScript font contains up to 256 glyphs (but usually far fewer than that), corresponding to a 256-character block of the Unicode character space. Many of these PostScript fonts are needed for each full-coverage OpenType font. The build system runs each individually through a FontForge script that removes overlapping sections of splines, this being an easier operation in FontForge than METAFONT. Once all PostScript fonts for an OpenType font have had their overlaps removed, it runs another FontForge script to combine them into the final OpenType font. Doing the overlap removal as a separate step is an optimization for the common case during development where only some of the PostScript fonts have changed: it reduces the amount of work needed to reassemble the updated OpenType font.

There are additional stages of processing in FontForge after the PostScript fonts are merged. The raw outlines generated by METAFONT may contain excessive or poorly-located spline control points; scripts in FontForge attempt to remove those. Similarly, some technical rules of the font formats (such as having points at the *x* and *y* extrema of each curve) need to be enforced. There is another processing chain for automated horizontal spacing and kerning of the proportionally-spaced styles. In that chain, the build system generates bitmap fonts in BDF format and a C program calculates spacing corrections, which are then applied back to the merged OpenType fonts. Other scripts are run on the side to do things like constructing OpenType glyph-substitution tables for Korean *hangul* support, and collecting data for proof generation. According to recent statistics



Figure 7: Three styles of “language” and “five”.

from Ohloh [2], 63% of the project’s code is written in MetaPost (the font descriptions proper), 8% is in L<sup>A</sup>T<sub>E</sub>X (documentation), and the remaining 29% is spread among 11 other programming languages: the infrastructure and some small spin-off packages.

### 3.4 The METAFONT code

Below is Tsukurimashou’s code defining the “language” glyph of Figure 1; three styles of it are shown at the top of Figure 7. This glyph is of about average complexity; some are even simpler, and a few involve much more complicated operations, such as calculating positions of strokes based on the intersections of other strokes, or doing interpolation and conditional processing on style parameters.

```
vardef kanji.grtwo.language =
push_pbox_toexpand(
  "kanji.grtwo.language");
build_kanji.level(build_kanji.lr(450,0)
  (kanji.grtwo.word)
  (tsu_xform(identity yscaled 0.95)
    (kanji.grnine.my)));
expand_pbox;
enddef;
```

This code is in a file named `tsuku-8a.mp`, which covers the Unicode code points U+8A00 to U+8AFF. A character like this one, which happens not to be used as part of any other character, is defined right there in the Unicode-range MetaPost file. Parts that are shared among more than one such file are moved to other files that can be included in multiple places; for instance, `kanji.grtwo.word` is in `gradetwo.mp`. Splitting macro definitions across many files like this makes it easier to avoid recompiling the whole system when something changes, but it also requires the build system to keep track of all the inter-file dependencies.

Tsukurimashou frequently uses a sort of functional programming via METAFONT’s concept of text arguments to macros. A global stack data structure contains several kinds of objects to eventually be rendered into the glyph. A macro receives one or more

arguments that are themselves fragments of code; it runs them, then examines the objects they added to the stack and possibly makes modifications. Macros that create *kanji* or parts of *kanji* normally put them into a square of arbitrary two-dimensional space defined by the coordinates from (50, -50) to (950, 850); the outer-level macros can then shift and scale that square into its final location in the finished glyph.

The macro `build_kanji.lr`, which combines things left-to-right, allows its two arguments to run, then scales and shifts their results to cover two smaller rectangles. The numeric arguments (450, 0) specify that in this case, the dividing line is at *x* coordinate 450, and the two rectangles overlap by an amount of 0. So the left side runs from (50, -50) to (450, 850) and the right side is from (450, -50) to (950, 850).

Many of the visual adjustments needed when parts are combined, can be had just by choosing the right values for the dividing line and overlap amount. But other macros seen in this sample include `build_kanji.level`, which adjusts the stroke widths in its argument to all be the same (which often, but not always, looks better) and `tsu_xform`, which applies an additional METAFONT transformation matrix to make `kanji.grnine.my` a little smaller. Even in this relatively simple glyph, some tweaking was necessary beyond just putting together existing pieces in a standardized way.

Now, let’s look at the code for the *kanji* numeral “five”, which is invoked indirectly by `kanji.grtwo.language` when it calls `kanji.grnine.my`. This glyph is shown at the bottom of Figure 7. It is a typical example of the basic shapes that are not made up of smaller components.

```
vardef kanji.grone.five =
push_pbox_toexpand("kanji.grone.five");
push_stroke((170,740)--(830,740),
  (1.6,1.6)--(1.6,1.6));
set_boserif(0,1,9);
push_stroke((500,740)--(350,20),
  (1.6,1.6)--(1.6,1.6));
push_stroke(
  (220,410)--(730,410)--(720,20),
  (1.5,1.5)--(1.5,1.5)--(1.4,1.4));
set_boserif(0,1,4);
set_botip(0,1,1);
push_stroke((120,20)--(880,20),
  (1.6,1.6)--(1.6,1.6));
set_boserif(0,1,9);
expand_pbox;
enddef;
```

The `push_stroke` macros save paths on the stack, with each stroke defined by one path for the spine of the stroke, and a second path describing how

the stroke weight (eventually translated to “width” through a style-dependent matrix) changes along the length of the stroke. Other macros, such as `set_boserif`, push other objects on the stack to indicate where serifs (*uroko*) should be added in styles that use them.

The whole thing, like `kanji.grtwo.language` before it, is bracketed by `push_pbox_toexpand` and `expand_pbox`, which respectively save, and adjust the size of, an object called a “proof box”.

After all the macros that specify a glyph have run, rendering code unwinds the stack and generates outlines for all the objects, writing them to the PostScript output. This code is where most aspects of the font style are applied. Styles define the pens used for stroking, transformations for calculating pen size, the shape of serifs and whether to use them, and can potentially override parts of the rendering code by defining hook macros to apply further effects.

I have never fully understood METAFONT’s traditional proof system based on greyscale fonts and “literate” programming, and in any case its reliance on the standard coordinate array `z[]` would not mix well with Tsukurimashou’s object stack concept. Tsukurimashou generates proofs in a completely different way. When unwinding the stack the rendering code writes a “proof file”, essentially a machine-readable log of all the things it is rendering. The build system collects the proof files and runs them through Perl scripts which generate TikZ/LaTeX files for an illustrated and cross-referenced edition of the source code. The proof boxes from `push_pbox_toexpand` result in annotations on the pictures, showing which part of each glyph came from which macro. Some information from the proof files also feeds into the kerning program, and is used for purposes like advising FontForge of white-on-black reversed glyphs, which represent exceptions to the overlap-removal rules otherwise applied.

#### 4 Character databases and IDSgrep

Adding characters to Tsukurimashou requires knowing what is already in the system and what is in the language. For example, when looking at something like the left side of “outlook”, I need to know whether such a thing already exists as a macro somewhere in the code base; whether many other characters in the language also include it (which would support the decision to create a new macro for future use); and which of its parts may be related to common shapes that could be used as guides for the new code. There are also simple coding questions to be answered, like “What was the name of that macro?” and “Which source code file is it in?”

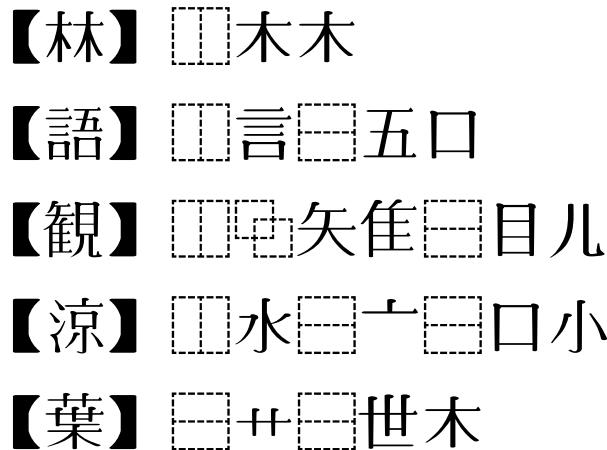
More generally, anyone working with Han characters who does not read them fluently may wish to search a dictionary on partial descriptions: “What is this character I don’t recognize that has ‘speak’ on the left and ‘five’ at the upper right?” Existing dictionaries sometimes offer what is called “multi-radical” search, whereby the user can specify one or more components and then see a list of all *kanji* that contain all those components. But multi-radical search features seldom if ever capture structural information like “on the left”; such a system would just show all the characters that contain “speak” in one pile for the user to dig through. In the initial stages of laying out Tsukurimashou’s *kanji* support, I frequently found myself wishing I could use the power of Unix regular expressions, or something like them, to make more precise queries: why not run `grep` on the writing system itself?

The IDSgrep package attempts to serve that need. With some irony intended, IDSgrep’s stated goal is to bring the user-friendliness of `grep` to Han character dictionaries. IDSgrep is one of the Tsukurimashou parasites: it comes included with the full distribution in a separate directory, or can be distributed on its own.

Recall the tree decomposition of Figure 1. That tree might be rendered into a simple ASCII-based prefix notation as “[lr] (speak) [tb] (five) (mouth)”: it is a left-right combination of two things, the first of which is “speak” and the second is a top-bottom combination of “five” and “mouth”. As argued earlier in this paper, such descriptions are not enough to render high-quality glyphs; but maybe if we include a few general catch-all categories like “overlap”, and accept that not all descriptions will be detailed enough for rendering graphics, we can come up with a description for every character sufficient to offer useful dictionary searches.

The Unicode standard specifies syntax for Ideographic Description Sequences (IDSes), intended to support exactly this kind of pursuit [14]. There are special characters defined in the range U+2FF0 to U+2FFB to represent the prefix operators. Figure 8 shows some examples of the notation. Note the way the IDS notation conceals some details: for instance, the two sides of “forest” are both denoted by the same character, even though they look different when rendered. This looks promising: maybe we could get away with “just running `grep`” on a database of such decompositions.

In practice there are some additional challenges. For theoretical reasons, namely the difference between regular and context-free languages, a true regular expression search on these descriptions may



**Figure 8:** Unicode Ideographic Description Sequences.

be less than satisfactory. IDSgrep implements a tree-matching query language in which the user can specify character components to search for explicitly, or use matching operators like wildcard, match-anywhere, Boolean operations, and so on. The IDS syntax is not quite sufficiently flexible and well-defined to encompass all the tasks IDSgrep demands of it, and the special Unicode combining operation characters are difficult to type (and to typeset in Computer Modern!); so IDSgrep defines extensions to the syntax and ASCII synonyms for the special characters, forming a language of Extended Ideographic Description Sequences (EIDSes) that subsumes the Unicode IDS syntax.

IDSgrep's user interface is a Unix command-line utility similar to `grep`. It reads a database of trees in EIDS syntax, from files or standard input, and writes out any that match the matching pattern specified on the command line: just like `grep`. The syntax for matching patterns is complicated because it is powerful, but no worse for skilled users than standard regular expressions. After learning the syntax, a user can easily and quickly compose queries like “What characters have this component in that location, but not that other component anywhere?”

The latest version, IDSgrep 0.4, uses Bloom filters and binary decision diagrams to speed up searches. Although the full tree-matching algorithm is not slow, a complete search of hundreds of thousands of *kanji* dictionary entries may take a few seconds. So during installation, IDSgrep precomputes bit vector indices for the databases being installed; when searching those databases, it can do quick tests on the bit vectors to reject the large majority of possible matches, running the more expensive tree match on the candidates that make it past the bit vector check. The amount of speed-up is variable,

but typically around a factor of 15.

But a critical question remains: where does the data come from? Databases of *kanji* marked up with structural data are not easy to find, let alone in IDSgrep's native format. The Tsukurimashou fonts generate (using information extracted from the proof files) a dictionary of character decompositions *as the characters appear in the fonts*. Querying how Tsukurimashou decomposes a character is often useful, but Tsukurimashou by definition does not cover the characters I have yet to add, and its decompositions may not reflect traditional etymology and other concerns. IDSgrep also ships with code to extract EIDS character decompositions from the KanjiVG Project's XML files [1] and from the CHISE IDS database [4]. It can do a “join” of any of the *kanji* databases with EDICT2 [3] to create an experimental dictionary of words and meanings with character decompositions. None of these databases is perfect; but especially by searching several at once, users can usually succeed in finding what they are looking for.

## 5 Conclusions and future work

There has been much past CJK METAFONT work, with few results and no finished fonts. I have described my own project, the Tsukurimashou parametric font meta-family, which is unfinished too. However, Tsukurimashou has made more progress than any similar system to date. I have described issues facing this kind of project, Tsukurimashou's solutions for some of them, and associated technology including the IDSgrep *kanji* structural query system.

The obvious direction for future work is to complete Tsukurimashou's *kanji* coverage. My hope, however, is that some of the code and ideas from this project will also be applicable in other languages and other projects.

## References

- [1] Ulrich Apel. KanjiVG. <http://kanjivg.tagaini.net/>.
- [2] Black Duck Software. The Tsukurimashou open source project on Ohloh: Languages page. [https://www.ohloh.net/p/tsukurimashou/analyses/latest/languages\\_summary](https://www.ohloh.net/p/tsukurimashou/analyses/latest/languages_summary).
- [3] Jim Breen. The EDICT dictionary file. <http://www.csse.monash.edu.au/~jwb/edict.html>.
- [4] CHISE Project. <http://www.chise.org/>.
- [5] John D. Hobby and Gu Guoan. A Chinese meta-font. *TUGboat*, 5(2):119–136, November

1984. <http://tug.org/TUGboat/05-2/tb10hobby.pdf>.
- [6] Don Hosek. Design of Oriental characters with METAFONT. *TUGboat*, 10(4):499–502, December 1989. <http://tug.org/TUGboat/10-4/tb26hosek.pdf>.
- [7] Bogusław Jackowski, Janusz Nowacki, and Piotr Strzelczyk. Programming PostScript Type 1 fonts using MetaType1: Auditing, enhancing, creating. *TUGboat*, 24(3):575–581, 2003. <http://tug.org/TUGboat/24-3/jackowski.pdf>.
- [8] Bogusław Jackowski and Janusz M. Nowacki. Latin Modern: Enhancing Computer Modern with accents, accents, accents. *TUGboat*, 24(1):64–74, 2003. <http://tug.org/TUGboat/24-1/jackowski.pdf>.
- [9] Koichi Kamichi. GlyphWiki. <http://en.glyphwiki.org/wiki/GlyphWiki:MainPage>.
- [10] Javier Rodríguez Laguna. Hóng-Zi: A Chinese METAFONT. *TUGboat*, 26(2):125–128, 2005. <http://tug.org/TUGboat/26-2/laguna.pdf>.
- [11] Tung Yun Mei. LCCD, a language for Chinese character design. Report STAN-CS-80-824, Stanford University, Department of Computer Science, 1980. <ftp://reports.stanford.edu/pub/cstr/reports/cs/tr/80/824/CS-TR-80-824.pdf>.
- [12] Tom Ridgeway. Poor Man’s Chinese and Japanese, 1990. <http://www.ctan.org/tex-archive/fonts/poorman>.
- [13] Tetsuro Tanaka. Wadalab-Toolkit. Web page in Japanese. <http://gps.tanaka.ecc.u-tokyo.ac.jp/wadalabfont/>.
- [14] Unicode Consortium. Ideographic description characters. In *The Unicode Standard, Version 6.2.0*, section 12.2. The Unicode Consortium, Mountain View, USA, 2012. <http://www.unicode.org/versions/Unicode6.2.0/ch12.pdf>.
- [15] Wenlin Institute. Character description language. <http://www.wenlin.com/cdl/>.
- [16] Candy L. K. Yiu and Wai Wong. Chinese character synthesis using MetaPost. *TUGboat*, 24(1):85–93, 2003. <http://tug.org/TUGboat/24-1/yiu.pdf>.

◊ Matthew Skala  
 Department of Computer Science  
 E2–445 EITC  
 University of Manitoba  
 Winnipeg MB R3T 2N2  
 Canada  
 mskala (at) ansuz dot sooke dot  
 bc dot ca  
<http://ansuz.sooke.bc.ca/>

## Braille fonts in Project Fandol

Clerk Ma

### 1 A short history

In China, Braille symbols were first introduced in 1874 by William Hill Murray, a preacher of the National Bible Society of Scotland. Collaborating with J. Crossette, Murray developed a system which can encode Braille with 408 different syllables in Mandarin Chinese. The shortcoming of Murray's system is that all the syllables are based on the Peking dialect of the Chinese language. This system cannot work properly for other dialects in North China. David Hill, a preacher of Wesleyan Methodist Missionary Society, developed another encoding system during 1888–1889. Hill's system influenced various Chinese Braille encoding systems in the 20th century. The current Braille encoding scheme used in mainland China was designed by Huang Nai (黃乃) in 1952.

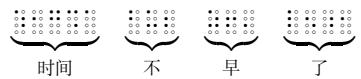
After the First Opium War, many preachers landed in China to do missionary work. The teaching of Braille was mainly in missionary schools. In 1928, the Braille Literature Association (BLA) was established in China, whose mission was to publish books in Braille. In 1933, a Braille version of John Bunyan's *The Pilgrim's Progress* by BLA became a best-seller among blind people. During the second half of the 20th century, after the foundation of the People's Republic of China, a national press named China Braille Press was established for publishing Braille books in Huang's Braille encoding scheme.

### 2 Standards for Braille

There are several GB (国家标准, National Standard) standards for Braille usage in China.

#### GB/T 15720-2008: Chinese Braille.

First published in 1995, revised in 2008. This standard has specified: (1) the form and size of Braille font, (2) the encoding of Braille, (3) hyphenation in Braille. According to this standard, a Braille sentence (meaning *It's getting late*) can be written as:



#### GB/T 18028-2010: Mathematical, physical and chemical symbols of Chinese Braille.

First published in 1995, revised in 2000 and 2010. This standard has specified the form and usage of mathematical, physical and chemical symbols. Adopting the Marburg and Nemeth systems, people can use Braille to express formulae, as in:

$$x' = f(x, t) \leftrightarrow \text{Braille formula}$$

### 3 Fandol's Braille fonts

More and more public facilities are marked with Braille in China. For example, the photo here shows Braille on a handrail:



Figure 1: Braille on handrail

Project Fandol has provided two Braille fonts which are inspired by Braille dots used in Chinese public facilities, and modified with rules from GB/T 15720-2008. All the Braille symbols are encoded in Unicode. The first is *FandolBraille-Display.otf* (fig. 2). Each glyph in this font has eight dots (empty or filled), and is designed for display screen.

The second font is *FandolBraille-Regular.otf* (fig. 3). This font is designed for printing.

### 4 Notes

- a. Unicode has a block called *Braille Patterns* (the range from U+2800 to U+28FF) assigned for these Braille glyphs. In fact, Chinese Braille only uses the range from U+2800 to U+283F, but I have no reason to abandon other code points which would be useful to other people in the world.
- b. The Braille glyphs in the GNU Project's FreeFont package (FreeMono) [4] are designed by Steve White. Braille glyphs in FreeMono conform to the proportions of a US Library of Congress standard [5], which are similar to the proportions in GB standards.
- c. William Park's *braille* package [6] also can produce Braille with Python scripts. But his implementation does not provide standalone Braille fonts. When using the *braille* package, every dot in every glyph is drawn via LATEX's \put command and *picture* environment.
- d. The picture is taken in a station of the Beijing subway. Other barrier-free structures, such as sidewalks for the blind, have also existed for a long time. These convenient facilities improve the safety of blind people in public.



## Case study: Typesetting old documents of Japan

NAKANO Ken and KOBAYASHI Hajime

This is a report on the typesetting of *Komonjo books* (reprint books compiling old documents of Japan) published by Shiryo Hensan-jo, with pTeX (pLATEX).

We would like to report that typesetting of *Komonjo books*, which are classified as difficult to typeset, can be made efficiently using TeX.

### About Shiryo Hensan-jo

From their web site (<http://www.hi.u-tokyo.ac.jp>): Shiryo Hensan-jo (the Historiographical Institute, HI), the University of Tokyo, has as its primary objective, rather than historiography in general, analysis, compilation, and publication of historical source materials concerning Japan.

The institute has become a major center of Japanese historical research, and makes historical sources available through its library, publications, and recently, databases.

### About us and TeX

Our company, Livretech, is a printing firm mainly producing textbooks and education-related books.

We've been using TeX since 1988, and have been concerned with some TeX-related topics. We developed a DVI driver program for the Shaken typesetter in 1989 (Shaken is a famous manufacturer in Japan making fonts and typesetting systems). We also typeset and output Japanese editions of *The TeXbook* (1992) and *The METAFONTbook* (1994).

### 1 Until TeX was chosen

Before TeX was chosen, *Komonjo books* were composed with hot metal types or phototypesetting systems developed for professional users. Several typesetting systems had been used, but commonly speaking, *Komonjo books* were difficult to typeset.

After typesetting was completed, contents of books were stored in the database named SHIPS (Shiryo hensanjo Historical Information Processing System), but another problem occurred.

In those days, Japanese typesetting systems worked with the creators' original character codes. For technical reasons and/or the creators' policies, typesetting data could not easily be re-used for another purpose. So, the final typesetting data would not automatically convert into database texts. To make the database text, typesetting contractor had to revise the final data manually, or in the worst case, input all texts again. And texts made in this way had to be proofread again.

To solve these problems, the professors of Shiryo Hensan-jo focused on TeX. This was in about 2001. They tried to typeset *Komonjo books* with TeX, and they found the possibility of typesetting and flexibility of text data.

And then, they contacted us and we started on this new challenge in 2002.

## 2 Typesetting problems

### 2.1 Problems with fonts

#### 2.1.1 Shortage of kanji characters

Massive numbers of kanji are used in *Komonjo books*. At that time, the Shaken system that was used to typeset *Komonjo books* had about 10,000 kanji characters. However, the fonts which could be used with pTeX had only about 6,500 kanji characters.

#### 2.1.2 Simplification of kanji shapes

In Japan, shapes of many frequently used kanji have been simplified to make them easier to read and write. Kanji of traditional shapes are called *Seiji* (正字), and those of simplified shapes are called *Ryakuji* (略字). Here are some examples of *Ryakuji* and *Seiji* character shapes:

<i>Ryakuji</i>	<i>Seiji</i>	meaning
医	醫	medical, doctor
円	圓	circle, yen
塩	鹽	salt
害	害	harm, damage
学	學	study, learn
国	國	country

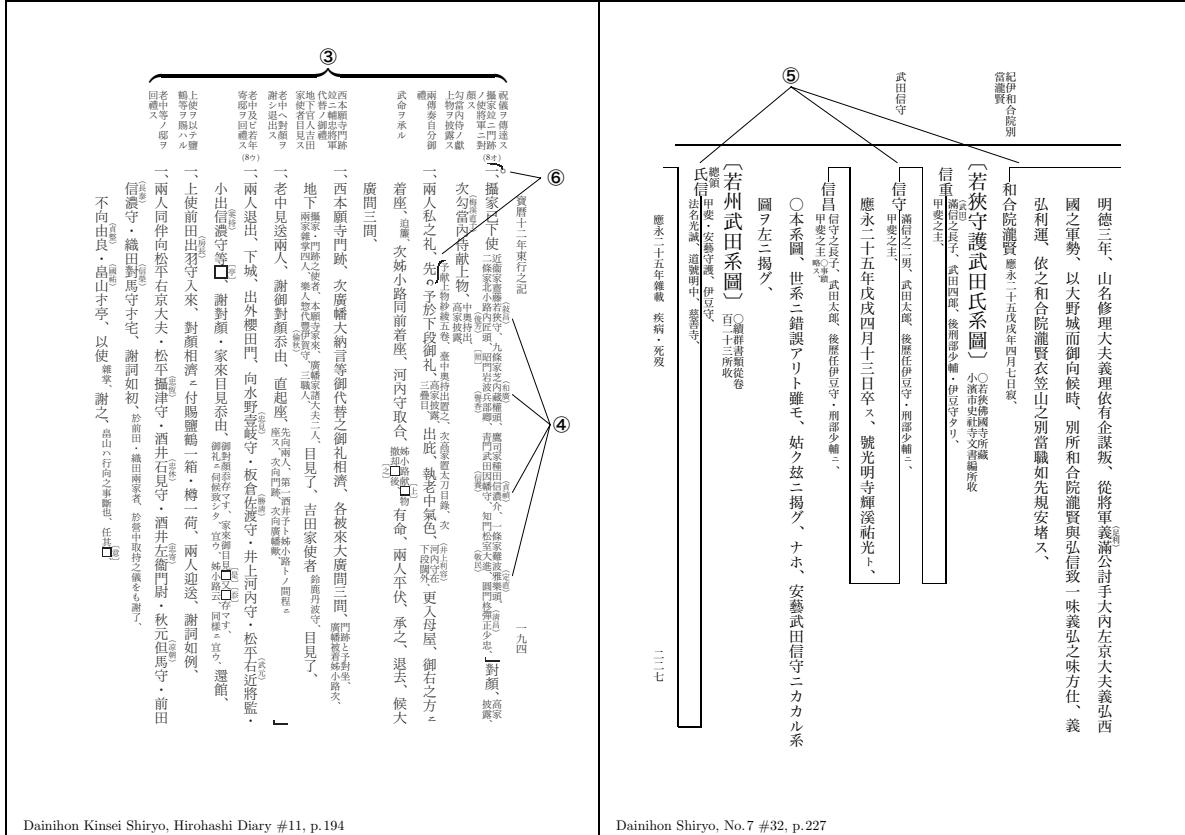
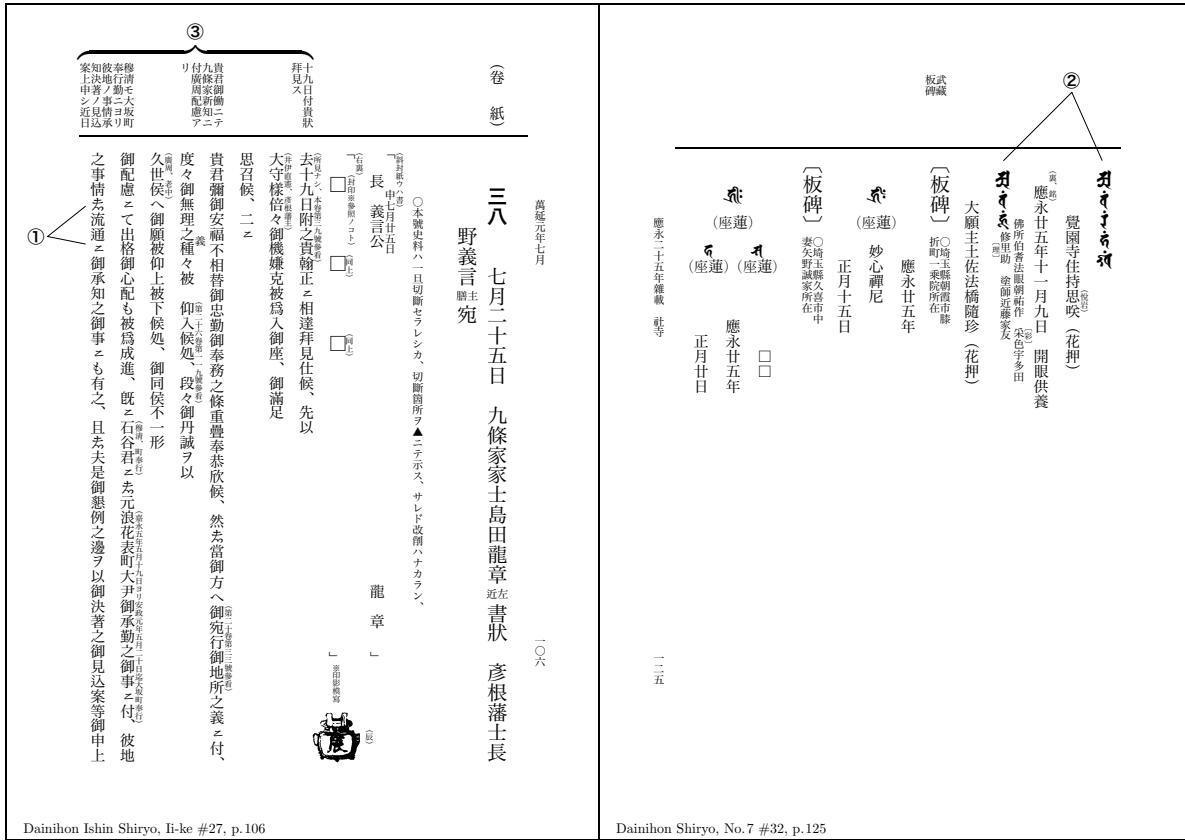
Like the letter "国," some *Ryakuji* characters have two or more *Seiji* characters.

In essence, *Komonjo books* are composed with *Seiji* characters. On the other hand, we use *Ryakuji* kanji in everyday life, and we rarely use *Seiji* characters. Because of this, *Seiji* characters have been put away in the dark recesses of fonts and kanji input methods. So it is difficult for us to input *Seiji* characters directly.

#### 2.1.3 *Hentai-gana*

変体仮名 (*Hentai-gana*, see ① in the figure on the next page) are Japanese old syllabary characters. *Hentai-gana* were made from kanji characters as well as modern *kana* characters. Here we show some *Hentai-gana* examples used in *Komonjo books*, with related kanji and *kana* characters.

original kanji	二 爾 而 八 者 者 者 江 三 茂
<i>Hentai-gana</i>	こ ろ へ も そ そ う み ま
modern Kana	に にて は は え み も

Examples of *Komonjo books* (reduced 53%).

*Hentai-gana* have different shapes with the same reading, while modern kana have one shape with one reading.

#### 2.1.4 Bonji

梵字 (*Bonji*, see ②) characters are mystical letters used in Japanese Buddhism. *Bonji* consist of consonants, vowels and derivative letters, so many characters can be created. The following are some typical *Bonji* characters and their meanings:

<i>Bonji</i> sound	meaning
𠂊	a, ア the most basic character
𠂑	āh, アーク Dainichi Buddha, 大日如来
𠂒	hrīh, キリク Amitabha, 阿弥陀如来

## 2.2 Problems with notes

*Komonjo books* contain many notes to aid readers' understanding. Two kinds of notes occur frequently.

One is a "head note" (③), similar to \marginpar but a long note appearing at the end of a page has to be split over pages.

The other is an "interline note" (④) which can occur anywhere in the text. To avoid overlapping of the interline note and text or other materials, position adjustment is needed.

## 2.3 Problems with picture-like materials

Picture-like materials such as family tree diagrams (⑤) and handwriting-like lines (⑥) interrupt the flow of text. Handling them is troublesome in TeX.

## 3 Solutions to problems

### 3.1 Solutions for font problems

#### 3.1.1 New kanji fonts are available

The Adobe-Japan1-5 character collection was published in 2002. This collection has 20,316 characters total, including 12,668 kanji characters. The number of kanji characters approximately doubled from the font that we used till then. AJ1-5 fonts could mostly cover the range of *Komonjo books*.

After that, the UTF package was developed by SAITO Shuzaburo. With this package, we could now use all characters of AJ1-5 fonts with pTeX.

In addition, we made an OpenType font named *ut-gaiji* to gather the additional characters which AJ1-5 did not have ("ut" means "u-tokyo" and "gaiji (外字)" means "external character").

#### 3.1.2 Input *Seiji* using *Ryakuji*

To avoid the difficulty of *Seiji* input, we decided to make *Seiji* texts from *Ryakuji* texts. Preparation in advance was as follows:

- Write a style file named *jitai.sty* ("jitai (字体)" denotes "character shape") to describe pairs of *Ryakuji* and *Seiji*, in order to specify *Seiji* kanji via *Ryakuji* kanji.

```
\def\seiji{\@ifnextchar[{\@seiji}{\@seiji[0]}}
\def \@seiji[#1]{\#2{\expandafter
\ifx\csname @SJ#1\#2 \endcsname\relax
\typeout{!!! Seiji #2(#1) undef.}\#2\relax
\else \csname @SJ#1\#2 \endcsname \fi }
\let\$=\seiji
\def\DefSeiji[#1]{\#2\#3{\expandafter
\gdef\csname @SJ#1\#2 \endcsname{\#3}}
\DefSeiji[0]{重}{亞}
\DefSeiji[0]{暁}{\CID{7633}}
\DefSeiji[0]{逢}{\CID{8266}}
\DefSeiji[0]{惡}{惡}
\DefSeiji[0]{医}{醫}
\DefSeiji[0]{学}{學}
\DefSeiji[0]{国}{國}
\DefSeiji[1]{国}{圈}
:
```

*jitai.sty* has about 1,000 such pairs. For example, after these definitions, the input \S{学} produces "學," and you can input \S[1]{国} to get optional *Seiji* character "圈."

The \CID command in a kanji pair specifies the internal character code from AJ1-5 (Character ID) directly.

- Write a Ruby program which converts *Ryakuji* texts into *Seiji* texts, referencing the *jitai* file. When the Ruby program finds *Ryakuji* kanji in a text file, it inserts the *Seiji* command.

For example, the kanji string "文学青年" ("Literary youth") is converted into "\S{文}\S{学}\S{青}年," and typeset as "文學青年."

Then, work steps are as follows:

1. Input text file with *Ryakuji* kanji.
2. Convert *Ryakuji* texts into *Seiji* texts.
3. Typeset *Komonjo books* with *Seiji* texts.

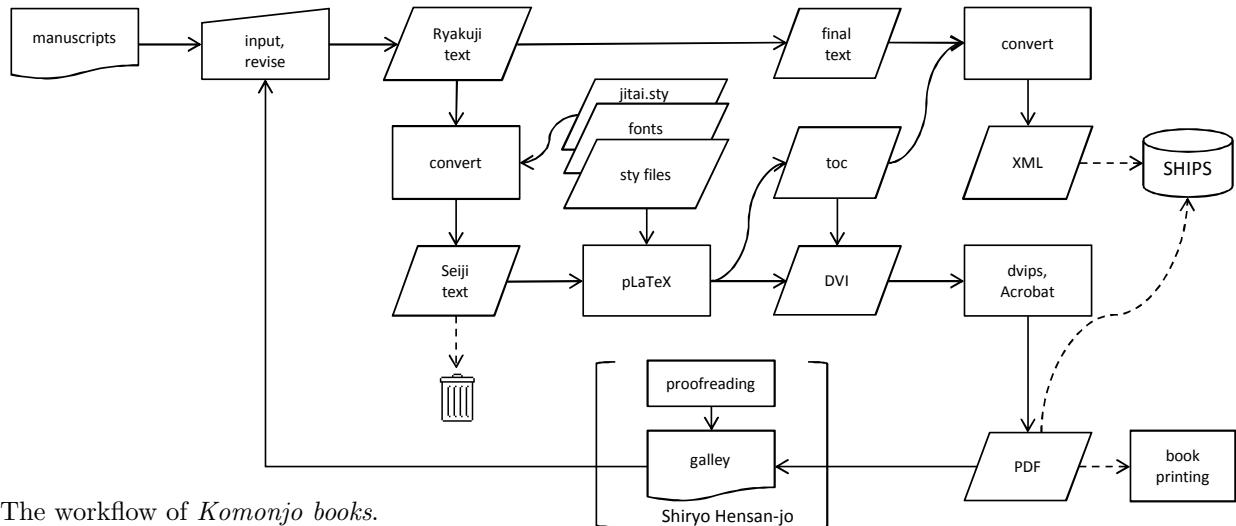
We revise only *Ryakuji* text files. We discard *Seiji* text files after typesetting has completed, since *Seiji* texts are hard to read due to the automatic insertion of *Seiji* commands.

#### 3.1.3 Hentai-gana

We made *Hentai-gana* as external characters and put them into *ut-gaiji*.

#### 3.1.4 Bonji

First, we used the free *Bonji* font, but it had only typical characters, so we often had to create additional characters. Therefore, now we use the 今昔文字鏡 (Konjaku-Mojikyo) font package which has about 2,000 *Bonji* characters. And we also use the *mojikyo* macro package developed by HONDA Tomoaki, to specify *Bonji* characters.



### 3.2 Solutions for notes

#### 3.2.1 Customize \marginpar for head note

A head note must be split when it “falls off” the end of a page. To implement this, we customized the definition of `\marginpar`. The original maintains a `\@marbox` to store a page of marginal notes. We just `\vsplit \@marbox` to `\textheight`, and put the remaining note text at the top of the `\@marbox` of the next page.

#### 3.2.2 Interline note

An interline note is placed at the right or left side of the text. We defined `\rnote` and `\lnote` respectively, and an optional position adjustment, as follows:

```
\rnote[v-adjust][h-adjust]{note text}.
```

### 3.3 Picture-like materials

We made the complicated family tree diagram as an integral number of line units, so that it can divide at the end of page.

For handwriting-like lines, we defined a `\Curve` macro, taking three points of a curve. Our `\Curve` is essentially the `\bezier` function with `\unitlength` of `1 zw` (= width of a kanji). The second curve of ⑥ (フ) was drawn with 3 lines as follows:

```
\Curve(.4,-.5)(.6,-.5)(.6,.2)
\Curve(.6,.2)(.6,.5)(.6,.5)
\Curve(.6,.5)(.6,1.1)(1.1,1.1)
```

### 3.4 Generate database texts

Another purpose of typesetting with `TeX` was to generate database texts from final `TeX` files. We wrote a Ruby program for this, and it works as follows: delete unnecessary information except text,

unify *Seiji* in *Ryakuji*, convert character code from Shift-JIS to Unicode, and convert from `TeX` file to the XML format of the database.

### 3.5 Result

We show the whole workflow of *Komonjo books* in the figure above. Using our methods, more than half of the *Komonjo books* of Shiryo Hensan-jo have been typeset with `TeX` now.

As a result, we think that the two purposes to be expected in `TeX` (efficiency of typesetting and re-using of input for the database texts) have been accomplished.

### Acknowledgements

First of all, we'd like to thank the professors of Shiryo Hensan-jo. They focused on `TeX` and selected us as a partner, and they were kind enough to teach everything about *Komonjo books*. We also give thanks to SAITO Shuzaburo for the development of the AJ1-5 font support packages. And also thanks to HONDA Tomoaki for the development of Mojikyo-related packages. Finally, many thanks to Karl Berry and Barbara Beeton for their kind support in reviewing this article.

### About rights

Products of this activity (style files, programs, etc.) belong to Shiryo Hensan-jo. They are not publicly available at this time.

- ◊ NAKANO Ken  
k-nakano (at) livretech dot co dot jp
- ◊ KOBAYASHI Hajime  
koba (at) livretech dot co dot jp

---

## upTeX — Unicode version of pTeX with CJK extensions

Takuji Tanaka

### Abstract

upTeX is a Unicode extension of ASCII's pTeX (a Japanese-localized TeX). It not only improves Japanese support, but also handles Chinese and Korean characters, i.e., Kanji (Hanzi, Hanja), Kana, CJK symbols, and Hangul with Unicode. Moreover, it can process multilingual typesetting of original LATEX with `inputenc` and Babel (Latin, Cyrillic, Greek, etc.) by switching its `\kcatcode` tables. This paper describes the main features of upTeX.

## 1 Introduction

### 1.1 Motivation

The objective of upTeX [1] is the “Unicodization” of ASCII Corp.'s pTeX [2]. p(LA)TeX is the most popular TeX system in Japan and is widely used as a typesetter for commercial printing. pTeX achieves professional quality Japanese typesetting [3] including Japanese hyphenation and vertical writing.

However, p(LA)TeX has some limitations due to the legacy encodings (double byte Japanese encodings, EUC-JP or Shift\_JIS). up(LA)TeX tries to improve this, while keeping the benefits of pTeX, as follows.

### 1.2 Enhancement of Japanese character set

The Japanese government has standardized several character set versions as JIS (Japanese Industrial Standard). In pTeX, the Japanese character set is basically limited to JIS C 6226 and JIS X 0208, namely JIS level-1 and -2 (6879 characters in 1990). In 2000, a new standard for character set JIS X 0213 was standardized, with an additional character set JIS level-3 and -4, 11233 characters in 2004. Moreover, additional characters which are useful in Japanese are defined in the Unicode standard. upTeX supports both sets, using the UTF-8 encoding.

### 1.3 Support of Chinese, Korean

pTeX basically supports Japanese and English (7-bit Latin). upTeX adds support of Chinese and Korean using procedures similar to Japanese typesetting. So now upTeX supports CJK (Chinese, Japanese and Korean).

### 1.4 Cooperation with Babel

pTeX has some difficulty handling 8-bit Latin due to the double byte legacy encodings. upTeX more easily treats 8-bit Latin, compared to pTeX. The

`inputenc` (option `utf8`) and Babel (Latin, Cyrillic, Greek, etc.) packages are available in upLATEX.

### 1.5 Compatibility with pTeX

Since upTeX keeps the typesetting procedures of pTeX, most macros for p(LA)TeX are expected to work in up(LA)TeX with minimal or no modifications.

### 1.6 Limitation

Current upTeX has several limitations.

In the area of multilingual support, upTeX handle CJK, Latin, Cyrillic and Greek. However, upTeX cannot directly treat complex characters, e.g., Arabic, Hindi.

upTeX includes the ε-Tex extensions, thanks to Kitagawa [4]. On the other hand, upTeX does not have pdftex extensions yet.

The (u)pTeX engine remains based on original TeX and written using WEB. It includes two special extensions: (1) (u)pTeX treats Japanese (CJK) glyph metrics with an extended format of `tfm`, called `jfm`; (2) (u)pTeX defines a special DVI command 255 for vertical writing.

Thus, (u)pTeX requires that related DVIware support `jfm` and this DVI command. Advanced features of OpenType fonts are hardly touched by pTeX and upTeX.

## 2 Implementation

This section describes the implementation of upTeX.

### 2.1 Unicodization

Table 1 compares the internal structure of original TeX, pTeX and upTeX. pTeX uses internally a 16-bit token for Japanese, which is not enough to cover the wide range of the Unicode character set. upTeX expands that to an internal 29-bit token for CJK, where the Ω (Omega) implementation was used for reference. upTeX internally “unicodizes” only CJK characters, treating 8-bit Latin characters the same as in original TeX.

Table 2 summarizes encoding in upTeX. upTeX I/O accepts UTF-8 with a variable length of one–four octets. Originally, upTeX was not so refined. The routine was cleaned up via the `ptexenc` library written by Tsuchimura [5]. upTeX assumes that characters beyond Unicode's Basic Multilingual Plane (BMP) have a fixed font metric to support Kanji on the Supplementary Ideograph Plane (SIP).

### 2.2 Extension of \kcatcode

Table 3 shows the classifications of the `\kcatcode` primitive in upTeX. pTeX defined `\kcatcode` values of 16 (Kanji), 17 (Kana), and 18 (Japanese symbol) for Japanese typesetting. upTeX extends the

\kcatcode of 15 (not CJK), defines 19 (Hangul), and 18 is redefined as CJK symbol.

When \kcatcode is set to 15 (not CJK), the character is treated like Latin characters in original TeX. This feature provides the improved Babel support in upTeX compared to pTeX. For example, users can switch \kcatcode to select whether Cyrillic and Greek characters are treated as proportional glyphs in their language, or treated as full-width glyphs in conventional Japanese fonts.

When \kcatcode is set to 19 (Hangul), end-of-line is treated as space.

### 2.3 Use of the DVI command set3

upTeX uses the DVI command set3 and supports Kanji of SIP (U+2xxxx), where some Kanji characters used daily in Japanese are included. The upTeX project prepared patches for some DVI software (dvips, dvipdfmx, dviout, xdvi and dvi2tty) to support SIP. In TeX Live, the patches for all but xdvi are (or will be) applied.

## 3 upTeX vs. other Unicode TeX

Table 4 compares features of original TeX, upTeX and other TeX families with Unicode support ( $\Omega$ , XeTeX [6] and LuaTeX-ja [7]).

The recommended practice for using upTeX is as a “better pTeX” with better Japanese support and/or better multilingualization with CJK and Babel support.

## 4 History and future

A very brief timeline of upTeX-related events:

1995	ASCII pTeX ver.2, pLATEX 2 $\varepsilon$ [2]
2007	upTeX first release, alpha version
2007	upTeX in W32TeX [8]
2008	$\varepsilon$ -upTeX by Kitagawa-san [4]
2012	upTeX 1.00
2012	upTeX in TeX Live

Currently, I believe upTeX has the capability of multilingual (CJK, Latin, Cyrillic, Greek) typesetting. Possible enhancements for the future:

- Support IVS (Ideographic Variation Sequence).
- Document classes for Chinese/Korean.
- Babel options for Chinese/Korean.
- Add pdfTeX extensions.
- Merge LuaTeX [9] or XeTeX [6].

## Acknowledgements

I deeply appreciate the work of ASCII Corporation (currently ASCII Media Works, Inc.). upTeX could not exist without the achievement of pTeX.

I thank Tsuchimura-san (土村展之氏), ZR-san (八登崇之氏) [10], Inoue-san (井上浩一氏), Kakutou-san (角藤亮氏), Okumura-san (奥村晴彦氏), Yasuda-san (安田功氏), Kuriyama-san (栗山雅俊氏), Kitagawa-san (北川弘典氏), Dora-san, Norbert Preining-san and the many people who discussed upTeX at the Japanese website TeX Q&A [11] and other places for their fruitful observations.

## References

- [1] Tanaka, Takuji. *upTeX, upLaTeX - Unicode version of pTeX, pLaTeX*, [http://homepage3.nifty.com/ttk/comp/tex/uptex\\_en.html](http://homepage3.nifty.com/ttk/comp/tex/uptex_en.html)
  - [2] ASCII Corporation (currently ASCII Media Works, Inc.). *ASCII Japanese TeX (pTeX)* (in Japanese), <http://ascii.asciiimw.jp/pb/ptex/>
  - [3] Okumura, Haruhiko. *pTeX and Japanese typesetting*, <http://oku.edu.mie-u.ac.jp/~okumura/texfaq/japanese/ptex.html>
  - [4] Hironori, Kitagawa, *e-pTeX Wiki* (in Japanese), <http://sourceforge.jp/projects/eptex/wiki/FrontPage>
  - [5] ptex Wiki, “UTF-8 対応 (4)” (in Japanese), <http://tutimura.ath.cx/ptex/?UTF-8%C2%D0%B1%FE%284%29>
  - [6] SIL International. *The XeTeX typesetting system*, <http://scripts.sil.org/xetex>
  - [7] *LuaTeX-ja*, <http://sourceforge.jp/projects/luatex-ja/wiki/FrontPage%28en%29>, <http://ctan.org/pkg/luatexja>
  - [8] Kakuto, Akira. *W32TeX*, <http://w32tex.org/index.html>
  - [9] The *LuaTeX* team. *LuaTeX*, <http://www.luatex.org/>
  - [10] Yato, Takayuki. *En toi Pythmeni tes TeXnopolos* 電脳世界の奥底にて (in Japanese), <http://zrbabbler.sp.land.to/upLaTeX> を使おう (in Japanese), <http://zrbabbler.sp.land.to/uplatex.html>
  - [11] *TeX Q&A* (in Japanese), <http://oku.edu.mie-u.ac.jp/~okumura/texwiki/>
- ◊ Takuji Tanaka  
Kokubunji  
Tokyo  
Japan  
KXD02663 (at) nifty dot ne dot jp  
[http://homepage3.nifty.com/ttk/comp/tex/uptex\\_en.html](http://homepage3.nifty.com/ttk/comp/tex/uptex_en.html)

**Table 1:** Comparison of structure among  $\text{\TeX}$ ,  $\text{p\TeX}$  and  $\text{up\TeX}$ .  
 † denotes that it works with the `inputenc` package.

			$\text{\TeX}$	$\text{p\TeX}$	$\text{up\TeX}$	
<b>Latin</b>	I/O		8 bit (multi-bytes)†	7 bit 1 byte	8 bit (multi-bytes)†	
		token    charcode catcode	8 bit 4 bit	8 bit 4 bit	8 bit 4 bit	
<b>CJK</b>	I/O		—	EUC-JP etc. 8 bit 2 bytes	UTF-8 8 bit 2–4 bytes	
		token    charcode kcatcode	— —	16 bit —	24 bit 5 bit	
<b>Latin/CJK classification</b>			—	fixed	customizable	
<b>inputenc</b>			ok	n/a	ok	
<b>Babel</b>			full	partial	full	

**Table 2:** Encoding in  $\text{up\TeX}$ .

	<b>Latin</b>	<b>CJK</b>		
	$\text{\TeX}$ compatible $<256$	up $\text{\TeX}$ extended BMP	over BMP	comment
.tex / .aux I/O buffer	1 byte	UTF-8 2–3 bytes	4 bytes	
token	12 bit	29 bit	with (k)catcode	
.dvi / .vf	set1 T1 etc. 8 bit	set2 UCS-2 16 bit	set3 UTF-32 24 bit	
.tfm	T1 etc. 8 bit	UCS-2 16 bit	—‡ ‡ treated as Kanji jfm for CJK	
.ps / CMap	T1 etc. 8 bit	UCS-2 16 bit	UTF-16 2×16 bit	

**Table 3:** \kcatcode in pTeX and upTeX. \* denotes upTeX extension.

\kcatcode	\catcode	kind	e.g.	control word	end of line
	...	...			
15*	10	space	_		
	11	char	azAZ	yes	as space
	12	other char	(.!?)	no	as space
	...	...			
16		Kanji	漢漢	yes	ignore
17		Kana	かナ	yes	ignore
18		CJK symbol	《』	no	ignore
19*		Hangul	한글	yes	as space

**Table 4:** Comparison of features among upTeX and other TeX families.

Symbols express the following: Better ... ◎ &gt; ○ &gt; △ &gt; ▽ ... worse.

		TeX	pTeX	upTeX	Ω	XeTeX	LuaTeX-ja
Compatibility	Latin	◎	○	◎	○	△	◎
	Japanese	—	◎	◎	▽	▽	△
Multilingual	Latin	◎	○	◎	◎	◎	◎
	Japanese	—	○	◎	△	△	○
	CK	—	—	◎	△	△	?
	others	—	—	—	△	◎	○
Integrity	(Japanese)	◎	◎	◎	△	△	○
Popularity	Japan	◎	◎	○	△	△	△
	World	◎	△	△	△	○	△

## Typesetting and layout in multiple directions—Proposed solution

John Plaice

### Abstract

I propose a new, general way of looking at typesetting and layout in multiple directions. It subsumes the left-to-right and right-to-left horizontal writing used in most of the world, as well as the vertical writing used in East Asia. The generality allows the development of layout schemes for situations when several writing directions appear on the same page.

The key to the approach is that managing multidirectional text requires a separation of writing style from box direction. It turns out that there are only three different kinds of writing style, and eight kinds of directional box, and that simple rules can be used to define how these different writing styles may appear in the different kinds of box.

### 1 Introduction

If  $\text{\TeX}$  or a successor thereof is to be used naturally for all of the world's languages, then the problem of typesetting in multiple directions must be solved in all of its generality.

Most of the languages around the world are printed horizontally, from left-to-right, with the first line of the page at the top of the page. In the Middle East, Arabic, Hebrew and several other scripts are written horizontally, from right-to-left. In East Asia, traditional writing and printing is done vertically, with the first line of the page on the right-hand side; in Japan, this practice is still common for literature. Uighur and Mongolian writing is also vertical, but the first line of the page is on the left.

The general problem does not consist in simply printing a text in each of the different directions. A typesetting system to be used for multilingual documents needs to be able to print, on the same page, multiple languages using different direction combinations, as needed. This means that headers, footers, columns, tables, paragraphs, and everything else appearing on a page can potentially appear in the different directions.

In this paper, I propose a general solution that solves not just the above-mentioned cases, but also even more complicated situations, such as for Ancient Greek *boustrophedon* and Rongorongo *reverse boustrophedon*, both with alternating line directions.

The solution, which supposes that the natural writing style of a script is separate from the boxes in which that writing will be embedded, is the result of much reflection examining existing approaches.

## 2 Previous work

### $\text{\TeX-X}\ddot{\text{E}}\text{T}$

In the  $\text{\TeX}$  world, the first work [5] in multidirectionality was made by Donald Knuth and Pierre MacKay, who developed  $\text{\TeX-X}\ddot{\text{E}}\text{T}$ , which allowed the mixing of left-to-right and right-to-left horizontal texts in the same paragraph. This was done by using nested  $\begin{L}$ – $\end{L}$  and  $\begin{R}$ – $\end{R}$  pairs to, respectively, embed left-to-right and right-to-left texts. However, their work supposes that all pages are left-to-right, so it is not suitable for true right-to-left documents.

### p $\text{\TeX}$

Mixed horizontal and vertical typesetting was introduced in the  $\text{\TeX}$  world with p $\text{\TeX}$  [2], a tool developed at ASCII Corporation in Japan. Still used in Japan, p $\text{\TeX}$  allows a vertical or a horizontal list to be begun with either of the  $\text{\yoko}$  and  $\text{\tate}$  primitives. In  $\text{\yoko}$  mode, horizontal and vertical boxes have the same meaning as in standard  $\text{\TeX}$ . In  $\text{\tate}$  mode,  $\text{\hboxes}$  are vertical and  $\text{\vboxes}$  are horizontal.

### CSS

The work on supporting multiple directions in Cascading Style Sheets (CSS) for HTML [1] has introduced some useful terminology:

**Block flow direction:** from first to last line;

**Inline base direction:** from first to last glyph;

**Line orientation:** direction towards “top” of line.

### Omega

Omega was the first successor to  $\text{\TeX}$  to attempt to solve the multidirectional problem in its generality [3, 4, 8]. It assumes that a box or a font's direction can be designated by three characters, where each is one of Top, Bottom, Left, and Right. These characters absolutely designate one of the edges of the physical page. Then a writing direction must designate:

**Primary part.** The “top” of each page.

**Secondary part.** The “left” of each line.

**Tertiary part.** The “top” of each character.

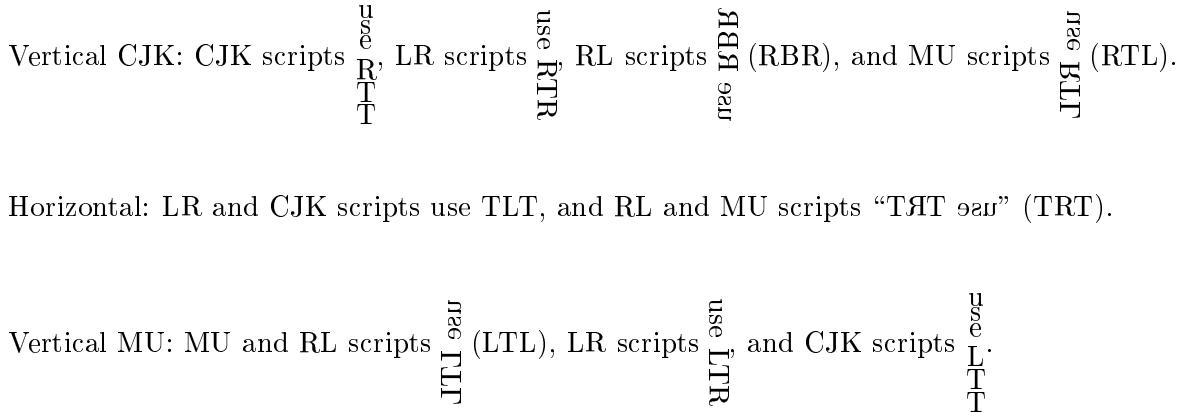
The secondary direction must be orthogonal to the primary direction. The tertiary direction can take all four values. Hence there are 32 possible directions. Here are the most common ones:

**TLT** — Left-right (LR) scripts, horizontal CJK.

**TRT** — Right-left (RL) scripts.

**RTT** — Vertical CJK, upright LR scripts in vertical CJK.

**LTL** — Mongolian and Uighur (MU).



**Figure 1:** Example of Omega text in many directions. The main flow of text is from left to right, but it includes parts that use a number of different directions. Each line explains the general structure for the most common cases of mixing writing directions.

**RTL** — MU scripts in vertical CJK.

**RTR** — Rotated LR scripts in vertical CJK.

**LTR** — Rotated LR scripts in MU.

**LTT** — Vertical CJK in MU.

Figure 1 demonstrates the use of several of these directions. In the text of a paragraph, a change of `\textdir`, which respects TeX's group nesting, will change the writing direction. In addition, boxes can have a direction definition, as can the page (for headers and footers), the body (of the page), the paragraph, and mathematics.

Notwithstanding the impressive number of possible writing directions, the proposed solution was not sufficiently general, as it did not make provisions for phenomena such as typesetting to a curve. Furthermore, it required different fonts for different writing directions, despite the fact that many of them simply involved rotating text.

### 3 The solution

In this paper, a completely separate approach, both simpler and more general, is taken: when typesetting, a particular writing style (there are three) is used, and the text will be placed in a secondary (text) box, of which there are eight kinds. Secondary boxes are lined up in a primary box.

The three writing styles are as follows:

- In *baseline left-to-right writing* (BL), there is a baseline which the glyphs are sitting upon or hanging from, and successive glyphs are placed to the right of previous ones. Most alphabetic scripts use this form of writing.

This is baseline left-to-right writing.

- In *baseline right-to-left writing* (BR), there is also a baseline, but successive glyphs are placed to

the left of previous ones. This form of writing is used for Arabic and Hebrew, but one can also consider the Uighur and Mongolian vertical scripts to be using the same style.

- In *axial writing* (A), there is an axis flowing through the glyphs, typically in the middle, and the glyphs are pinned onto the axis, one after the other. The vertical typesetting used in Japan, China and Korea is axial writing.

W	T
R	H
I	I
T	S
I	I
N	S
G	°
	A
	X
	I
	A
	L

The underlying assumption is that a writing system should be invariant to rotation, and not be defined by the box holding it. For example, typesetting along a curve requires no boxes.

TeX's `\vboxes` and `\hboxes` should be, respectively, generalized to *primary* and *secondary* boxes. Here, text will be placed in secondary boxes. Both primary and secondary boxes have a direction and an orientation. The direction is designated by one of T, B, L, or R, and the orientation is either positive (+) or negative (-), where positive or negative refers to the angle made between the direction vector of text and the normal vector pointing towards the part of the

line where annotations such as footnotes are made. Hence, English would normally be placed in a L+ secondary box, Arabic in a R- box, vertical Japanese in a T+ box, and Mongolian in either a T+ or T- box, depending on where the annotations are placed with respect to the line.

Any piece of text should be embeddable into a text of another style, and into any kind of secondary box. However, when doing so, the text must “adapt” to the box in which it will be placed, possibly by being mirrored or rotated so that it can fit therein.

#### 4 Example embeddings

In this section, we will consider a number of possible embeddings of a text *A* in a text *B*, where the two texts are using different writing styles. In all cases, these texts are:

***A*: pneumonoultramicroscopicsilicovolcanoconiosis**

***B*: I am glad ... is not well known.**

Text *A* was chosen because it is long, with many possible hyphenation points.

##### Embedding a baseline text in another baseline text

We consider two main cases. In the first, texts *A* and *B* are of the same style. In this case, depending on the fonts, vertical adjustment of the baseline of the embedded text may be necessary.

I am glad pneumonoultramicroscopicsilicovolcanoconiosis is not well known.

In the second case, texts *A* and *B* are of opposite orientation. There are two subcases. In the first, text *A* is mirrored to resemble the style of text *B*.

I am glad pneumonoultramicroscopicsilicovolcanoconiosis is not well known.

In the second subcase, the  $\text{\TeX}-\text{\XeT}$  bidirectional paragrapher is used to embed text *A*.

pneumonoultramicroscopicsilico- bas is laq pneumono-  
volcanoconiosis is not well known.

##### Embedding an axial text in a baseline text

We consider two main cases. In the first, with two subcases, the axial text is rotated,  $90^\circ$  or  $-90^\circ$ .

I am glad M O I C R P N S E U O C O U M I O N C O U L T I T R A  
V O L C A O N O C O O N I C O N I O S I T C R O  
is not well known.

I	am	glad	>	R	T	L	U	M	O	N	O	E	N	P
O	C	I	T	S	I	D	C	M	O	C	P	C	I	M
S	I	O	N	O	C	O	S	R	O	O	O	C	V	O
I	O	S	I	O	C	N	A	O	L	C	C	A	L	S

In the second case, the axial text is not rotated, but an inner paragrapher is called upon to create a set of lines perpendicular to the lines of the outer paragraph. For this to work, the length of these inner lines is declared. There are two subcases.

I	am	glad	P	O	T	C	O	I	V
			N	N	R	R	P	L	O
			E	O	A	O	I	I	L
			U	U	M	S	C	C	C
			M	L	I	C	S	O	A
N	I								
O	O								
C	S								
O	I								
N	S								

is not well known.

I	am	glad	V	I	O	C	T	O	P
			O	L	P	R	R	N	N
			L	I	I	O	A	O	E
			C	C	C	S	M	U	U
			A	O	S	C	I	L	M
I	N								
O	O								
S	C								
I	O								
S	N								

is not well known.

##### Embedding a baseline text in an axial text

In the first case, the embedded text is rotated.

K	niosis	ultramicroscopicsilicovolcano-	I	I
N	A	A	N	N
O	M	M	O	O
W	G	G	W	W
N	L	L	S	S
°	N	A	N	N
	O	D	O	O
	T	D	T	T
	W	pneumo-	W	W
	E		E	E
	L		L	L
	L			

In the second case, the embedded text is not rotated, and an inner paragrapher is called. Once again, the length of the lines of the inner paragraph must be stated.

K	volcano-	I
N	coniosis	A
O	I	M
W	S	G
N	N	L
°	O	A
	T	D
W	pneumo-	
E	noultra-	
L	microscop-	
L	icsilico-	

## 5 The importance of box orientation

The direction of a secondary box is not sufficient to completely describe it. Consider the following segment of an example presented by Ken Nakano [7]:



There are two annotation lines, one to the right, one to the left. The box orientation is positive, so the annotation line to the right (resp. to the left) would be considered to be the major (resp. minor) one.

Note that here, there are three parallel streams, each with its own rules for line breaking. This is in fact a special case of multiple interacting streams. See [6] for a more general discussion.

## 6 Implementation

Adapting existing  $\text{\TeX}$  engines to implement this proposal should be relatively straightforward. There would be a limited number of primitives (text mirroring, clockwise and counterclockwise rotation, vertical and horizontal displacement), using a limited

set of parameters: numeric (baseline shift), Boolean (to mirror or not, to rotate or not, clockwise or counterclockwise), and direction (text, mathematics, paragraph, page body, page).

I think that p $\text{\TeX}$ , X $\text{\TeX}$  and Lua $\text{\TeX}$  can all be extended in these ways, and that much of the relevant code from the Omega project is still useful. The approach is also extendable to new writing styles, e.g., axial bottom-up writing, or to new kinds of text embeddings. The new algorithms to be added include a paragrapher within the paragrapher (straightforward), and a line-breaking algorithm for multiple simultaneous streams (non-trivial).

## References

- [1] fantasai and Koji Ishii. CSS Writing Modes Level 3. <http://www.w3.org/TR/css3-writing-modes>, W3C Working Draft, 24 October 2013.
- [2] Hisato Hamano. Vertical typesetting with  $\text{\TeX}$ . *TUGboat*, 11(3):346–352, September 1990.
- [3] Yannis Haralambous and John Plaice. The Omega Typesetting and Document Processing System. *BIT*, 4:137–152, 2001. In Japanese.
- [4] Yannis Haralambous and John Plaice. Traitement automatique des langues et composition sous Omega. *Cahiers GUTenberg*, 39–40:139–166, May 2001.
- [5] Donald Knuth and Pierre MacKay. Mixing right-to-left texts with left-to-right texts. *TUGboat*, 8(1):14–25, April 1987.
- [6] Blanca Mancilla, Jarryd P. Beck, and John Plaice. Typesetting multiple interacting streams. In Cyril Concolato and Patrick Schmitz, editors, *ACM Symposium on Document Engineering*, pages 149–152. ACM, 2012.
- [7] Ken Nakano and Hajime Kobayashi. Case study: Typesetting old documents of Japan. *TUGboat*, 34(3):281–284, 2013.
- [8] John Plaice and Yannis Haralambous. Writing in multiple directions in Omega. In *Fourth International Symposium on Multilingual Computing*, Tokyo, Japan. 1–3 March 2001.

◊ John Plaice  
Montreal, Canada  
UNSW, Sydney, Australia  
`johnplaice (at) gmail dot com`  
<http://plaice.web.cse.unsw.edu.au>

## TEX Live Manager's rare gems: User mode and multiple repository support

Norbert Preining

### Abstract

This article describes two features of *TEX* Live Manager: *user mode*, where the *TEX* Live Manager manages an arbitrary directory instead of the system distribution, and *multi-repository support*, where multiple sources can be used to fetch *TEX* Live packages.

### 1 Introduction

The *TEX* Live Manager (`tlmgr`) is responsible for managing a *TEX* Live installation. It can be used to search for packages as well as doing usual package management tasks (install, update, remove, backup, configure). In the last couple of releases, two more features are available that have been requested for a long time: *user mode* and *multiple repository support*.

In user mode, rather than managing the system tree and installation, `telmgr` is used to manage an arbitrary `texmf` tree, for example the user's `TEXMFHOME` (this is in fact the default user mode tree). Although not all functionality is possible in user mode, the basic operations of package installation, removal, and updates can be handled by any user without requiring write permission to any system trees.

Multiple repository support was introduced to allow for easy handling of additional repositories of *TEX* Live packages. A few of them have come into existence (`tlcontrib`, `tlcritical`, `tlptexlive`, Korean support, ...), but until now one had to update packages one by one from these repositories. With multiple repositories this has been made a bit more convenient.

We will describe the background and operation of both of these features—why it was implemented, how it works, example runs, and warnings on usage.

### 2 User mode

This feature was probably one of the most requested in recent times: the ability to use `telmgr` to manage an arbitrary `texmf` tree. Although many points remain to be improved, the current implementation allows users to install most packages, without needing write access to the main *TEX* installation.

The *TEX* Live Manager `telmgr` has for many years been the main tool for configuration and management of a *TEX* Live installation. Most importantly, it is the main tool for updating existing packages and installing new ones.

In contrast to the original distribution of *TEX* Live from TUG, repackaged versions of *TEX* Live for

full operating systems (Debian, SuSE, Fedora, ...) normally do not ship `telmgr` (see article on pages 297–301), as it would interfere with the system package manager. But since distributions cannot feasibly keep up with the constant flow of *TEX* package updates, users have often asked for an option to use `telmgr` to handle packages within their `TEXMFHOME`.

#### 2.1 Background and description

Although I assume that the *TEX* Live Manager is by now sufficiently well known, here is a short reminder of some of the responsibilities of `telmgr` in managing the full installation:

- options: paper size, installation source, creation of formats, etc.
- platform: adds/removes support for different platforms.
- recreates core configuration files: `updmap.cfg`, `fmtutil.cnf`, hyphenation definitions.
- packages: installs, removes, backups, restores.
- information: search and browse through the *TEX* Live database.

Most of these operations need full write access to the installation, which, particularly in the cases of distribution-repackaged or shared installations, may well be unavailable.

In contrast to this, `telmgr` in *user mode* allows only a restricted set of functionality, and makes changes only below the home directory of the current users. Almost all actions are supported, with one important general restriction:

Only *relocatable* packages can be installed in user mode. A relocatable package resides completely within one `texmf` tree. This excludes all packages with binaries or scripts.

Otherwise, operations work similarly to normal mode, but changing the respective local files. For example, the `generate` action creates various configuration files in the user's `TEXMFVAR` instead of in the system-wide directory `TEXMFSYSVAR`.

In the following we will go through the necessary steps and give a few examples.

#### 2.2 Setup and operation

By default, user mode is not selected, so the command line argument `--usermode` has to be given on *all* operations.

##### 2.2.1 Initializing the user tree

Before starting to use user mode, a target tree has to be initialized. By default this is the user's `TEXMFHOME`, but any other tree can be specified with the command line option `--usertree`. This initialization is done by running `telmgr --init-usertree`, which sets up

the initial empty **T<sub>E</sub>X** Live database and creates necessary files. If the file **TEXMFHOME/tlpkg/texlive.tlpdb** already exists, the command will bail out. Other directories generated are **TEXMFHOME/web2c** and **TEXMFHOME/tlpkg/tlpobj**.

The initial **texlive.tlpdb** contains the very same options as the main installation concerning installation source, installation of documentation and source files, etc. After the initial setup, these options can be changed within the user tree in the normal way.

### 2.2.2 Installation of packages

As already mentioned, not all packages can be installed in user-managed trees; all the packages that contain files *outside* the **texmf-dist** hierarchy are not installable in user mode. Technically, only those packages tagged as **relocatable** are installable in user mode—which is most of them. Looking at the current **T<sub>E</sub>X** Live database, at the moment there are 2454 packages that can be installed in user mode.

Having set up the initial tree, installation is straightforward (some line breaks are editorial):

---

```
$ tlmgr --usermode install 12many
tlmgr: package repository /home/norbert/tlnet
[1/1, ???:??/??:??] install: 12many [376k]
tlmgr: package log updated: /home/tl/texmf/
    web2c/tlmgr.log
running mktexlsr ...
```

---

Dependencies of collections and of packages are installed automatically, which allows installation of a full collection (more precisely, all of the installable packages from a collection):

---

```
$ tlmgr --usermode install collection-xetex
tlmgr: package repositories: /home/norbert/
    tlnet
[1/28, ???:??/??:??] install: arabxetex [618k]
[2/28, 00:00/00:00] install: euenc [153k]
[3/28, 00:00/00:00] install: fixlatvian [123k]
...
Package xecyr is not relocatable, cannot
    install it in user mode!
...
[27/28, 00:07/00:08] install: collection-xetex
    [1k]
tlmgr: package log updated at /home/tl/texmf/
    web2c/tlmgr.log
running mktexlsr ...
```

---

Package removal, updates, etc., all work in the analogous way.

### 2.3 Warnings and future work

User mode support is still relatively new, and has not been used extensively, so expect bugs to creep

in. If you run across one, please let us know how to reproduce.

Another caveat with user mode is that all files in **TEXMFHOME** override system-wide files. In other words, if the system installation gets updated and the user installed packages become older, then the older version will still be used (which can be the desired behavior, after all).

Also, be aware that the size of **TEXMFHOME** can grow rapidly, especially when installing collections. Since this tree is not (by default) searched via **ls-R**, searching can become quite slow.

Last but not least, be warned if you are tempted to create copies of configuration files (see my other article in this volume cited earlier), as they are an endless source of problems—especially ones that are forgotten.

Concerning future developments of user mode, the set of supported actions in **tlmgr** is already sufficient. What remains to be done: support in the GUI, a distribution installation mode where there is no initial **T<sub>E</sub>X** Live database available, and perhaps an independent installation support, where there is not even a main **T<sub>E</sub>X** Live installation required.

## 3 Multi-repository support

Online updating of a **T<sub>E</sub>X** Live installation is done by fetching the **T<sub>E</sub>X** Live database of the remote repository, determining the set of updated packages, fetching the updated packages from that same remote repository, and finally unpacking and installing them. Multi-repository support allows for fetching from several sources, on a per-package basis.

### 3.1 Background and description

Over the years several additional repositories have come into existence. The first, as part of **T<sub>E</sub>X** Live itself, is called **tlccritical**, and provides testing releases of the **T<sub>E</sub>X** Live core infrastructure packages (e.g., **tlmgr**). Other notable repositories in use today are **tlptexlive** (for testing Japanese **T<sub>E</sub>X** integration and binary updates) and **tlcontrib** (for testing releases and items not distributable in **T<sub>E</sub>X** Live).

Since the very beginning, doing the full update circle as described above from any repository has been possible. But it was repetitive: a user wanting to update from different sources had to run **tlmgr** several times, once for each repository. The multi-repository support now allows configuring **tlmgr** to pull automatically from several repositories at the same time, somewhat similar to (though still different from), for example, **apt-get** in Debian.

There are a few points that set the TeX Live implementation of multi-repository support apart from similar features in other systems:

- There is a distinction between the main and subsidiary repository. The main repository should always be our `tlnet` distribution channel.
- By default everything is taken from the main repository. Subsidiary repositories are never used unless explicitly requested.
- To request a package from a subsidiary repository one has to *pin* this package to the respective repository, as explained below.
- Absolute revision numbers are *not* compared between repositories—only the pinning counts. Revision numbers are only used to compare between the local version and the selected version from the repository.

The necessity to pin a package explicitly to a subsidiary repository arose from our wish to avoid unnecessary splitting of repositories. In general, we want to have all freely available TeX-related packages to be available via TeX Live's `tlnet` channel. Furthermore, we want users to think twice before using packages from subsidiary repositories, as it can easily lead to problematic situations.

### 3.2 Setup and operation

By default `tlmgr` works in single-repository setup, well-known and well-tested. This repository can be set and changed with the invocation:

```
tlmgr option repository <url>
```

#### 3.2.1 Inspecting and adding repositories

To work with multiple repositories, a new `tlmgr` action `repository` has been added, with three sub-actions: `list`, `add`, `remove`. Initially there is only one repository:

---

```
$ tlmgr repository
List of repositories (with tags if set):
  /home/norbert/tlnet (main)
```

---

(The path here is my local copy of the `tlnet` directory from CTAN.) Giving `tlmgr repository` without any sub-action executes the `list` sub-action.

If we want to add a repository, we use the `add` sub-action:

```
tlmgr repository add <url> [tag]
```

where the `tag` is an optional short-hand for `<url>`. The main repository always has the tag `main`. As an example, let's add a local copy of the `tlptexlive` repository:

---

```
$ tlmgr repository add \
  /home/norbert/tlptexlive tlptexlive
tlmgr: added repository with tag tlptexlive:
  /home/norbert/tlptexlive
$ tlmgr repository
List of repositories (with tags if set):
  /home/norbert/tlnet (main)
  /home/norbert/tlptexlive (tlptexlive)
```

---

#### 3.2.2 Pinning

As mentioned above, before a subsidiary repository is used at all, it is necessary to pin the desired packages to the repository. Pinning is defined in the file `TEXMFLOCAL/tlpkg/pinning.txt`. This file consists of empty lines, comments starting with `#`, and lines of the form:

```
<repo>:<pkgglob>[,<pkgglob>] ...
```

where `<repo>` is a full URL or a repository tag given to `repository add`. `<pkgglob>` is a shell-style glob for package names, with multiple items separated by commas.

Editing this file manually is simple and perfectly ok to do. `tlmgr` also provides a convenience action to add, modify, and remove pinning data. The respective operations are:

---

```
tlmgr pinning [show]
tlmgr pinning add <repo> <pkgglob> ...
tlmgr pinning remove <repo> <pkgglob> ...
tlmgr pinning remove <repo> --all
```

---

As before, the `show` is optional (i.e., the default), and lists all current pins. To add new pinning data, use `pinning add` with the subsidiary's full URL or `<tag>` for `<repo>`, and then a list of package globs. As usual, when `<pkgglob>` contains shell meta characters they have to be quoted properly on the command line. The `pinning remove` invocation erases the listed `<pkgglob>`s from the list for `<repo>`, and does nothing if there is no such pair. Finally, one can remove all entries of a certain repository with the last of the above four invocations.

Continuing from the above example, let us see this in action and specify that we want to install *all* available packages from the `tlptexlive` repository. “All” in shell-glob syntax is just `*`, so:

---

```
$ tlmgr pinning add tlptexlive '*'
tlmgr: package repositories:
  main = /home/norbert/tlnet
  tlptexlive = /home/norbert/tlptexlive
tlmgr: new pinning data for tlptexlive: *
```

---

### 3.2.3 Installation of packages

Installation of packages from the selected repository is completely transparent. `tlmgr` checks which packages are pinned to which repository, compares the local revision with the revision in the pinned repository, and updates as necessary.

Let us see this first in action for the installation of a new package not present in the main repository:

---

```
$ tlmgr install pmetapost
tlmgr: package repositories:
  main = /home/norbert/tlnet
  tlptexlive = /home/norbert/tlptexlive
[1/2, ???:???:???:?] install: pmetapost.x86_64-
  linux @tlptexlive [671k]
[2/2, 00:00/00:00] install: pmetapost
  @tlptexlive [1k]
tlmgr: package log updated: /home/tl/texmf/
  web2c/tlmgr.log
running mktexlsr ...
```

---

We can see that `tlmgr` is quite verbose in telling the user from which repository the package is installed.

Updates work in the same general way. Here we see that `tlmgr list` shows all the candidates available, from both repositories along with the respective revision numbers:

---

```
$ tlmgr update --list
tlmgr: package repositories:
  main = /home/norbert/tlnet
  tlptexlive = /home/norbert/tlptexlive
tlmgr: saving backups to /home/tl/tug2013/tlpkg
  /backups
update: dvips.x86_64-linux [136k]: local: 30204,
  source: 31002@tlptexlive
  other candidates: 30204@main
update: ptex.x86_64-linux [530k]: local: 30519,
  source: 31001@tlptexlive
  other candidates: 30519@main
...
```

---

To actually perform the updates, we would run the usual:

---

```
$ tlmgr update --all
```

---

### 3.3 Warnings and future work

Here are the obligatory set of warnings, redoubled for an operation that so deeply changes the normal behavior of `tlmgr`:

*No purely numeric comparison for selecting the candidate:* Candidates are selected solely based on the pinning, and not by selecting the highest number

of the revisions in all repositories. This allows subsidiary repositories to have version numbers which are completely independent from the main `TEX Live` repository, where revision numbers are based on the Subversion revisions and thus are (other than being strictly increasing) unpredictable. As a consequence, this means that pinning a package to an alternative repository where the revision number is *smaller* than the one in `tlnet` will *not* automatically update the package the first time. An invocation of

`tlmgr install --reinstall <pkg>`

is needed. After having done that the first time all further updates will come automatically from the subsidiary repository.

*Support for actions:* Not all operations of `TEX Live Manager` can be supported, but those for which it is reasonable are done. The GUI has basic support in that sources can be added/removed, and information is displayed. Pinning is not possible in the GUI at present, and the presentation could be improved.

*Leftover packages:* Due to the fixed pinning, if an outdated package (like a development release) is *not* removed from a subsidiary repository, the user will remain stuck with the development version even if a newer version has found its way into the main repository. Removing the development package from the subsidiary repository makes sure that this does not happen, and is by far the cleanest and best solution.

Last but not least, multi-repository support is a recent addition and should thus be tried with special care and thought.

## 4 Closing

The development of `TEX Live Manager` has slowed, or more properly stabilized, as we come closer to the (asymptotic) point of feature completeness. The two additions described in this article have been in testing for about two years, and released in small steps to the general `TEX` audience, based on user requests and with attention paid to the design.

If you find problems, bugs, erratic behavior, unclear documentation, or you have further feature suggestions, please contact us at `texlive@tug.org`.

◊ Norbert Preining  
Japan Advanced Institute of  
Science and Technology  
Nomi, Ishikawa, Japan  
norbert (at) preining dot info  
<http://tug.org/texlive>

## Redistributing TeX and friends

Norbert Preining

### Abstract

Nowadays most TeX installations are based on TeX Live. TUG provides a platform-independent installer which can be used on many different platforms. But operating system distributors, such as Debian and Red Hat normally integrate TeX Live into their own packaging infrastructure.

Based on years of experience in packaging TeX Live for Debian, as well as upstream development, we give here a short introduction to the TeX Live ecosystem, list important files in need of special care when redistributing TeX Live, and give advice and warnings.

### 1 Introduction

The TeX environment has grown slowly but steadily into a huge collection of programs, fonts, macros, documentation, and more. TeX Live currently ships over 3 GB in more than 2500 different TeX Live “packages”, most of which are installed into TeX Live from CTAN [1]. Since teTeX development and support stopped several years ago, TeX Live has become the main TeX distribution on Unix, including Mac OS X (MacTeX is exactly TeX Live plus a few Mac-specific additions); it is also gaining on Windows (where MiKTeX is still strong).

Integrating TeX Live into any full operating system distribution is a non-trivial task due to the large number of post-installation tasks that have to be performed. Although over the last years the quality of packages has improved, the TeX Live development list still often gets bug reports that stem from incorrect packaging.

In the following we will give an overview of the structure of TeX Live and a list of important and special configuration files. Furthermore, based on the experience of packaging TeX Live over many years, we will give some advice and examples of best practices. Although the author maintains TeX Live for Debian, the information in this article is not targeted specifically for Debian, but at any distribution that redistributes TeX Live in one way or another.

The layout of the article is as follows: We will first give an overview of the structure of the TeX Live ecosystem. After that we discuss stacked versus non-stacked configuration files, followed by a discussion of the most important configuration files that need special handling. Finally, we collect some ideas concerning approaches to packaging TeX Live found in distributions.

## 2 Structure of TeX Live

TeX Live currently ships something like 130,000 files. To make this vast amount of material easier to handle we have introduced a hierarchical organization.

**Schemes** form the topmost level, with a dozen or so schemes defined. The default is `scheme-full`, which installs everything; at the other extreme is `scheme-minimal`, which installs only enough to run plain TeX. Schemes contain overlapping content; e.g., clearly everything in `scheme-minimal` is also contained in `scheme-full`.

**Collections** form the middle layer, with currently 45 collections. Each collection contains related (to some degree) packages. An example here is `collection-latex`. In contrast to the schemes, the collections form a mathematical partition of the content, that is, non-overlapping: every package is in exactly one collection.

**Packages** form the bottom layer, with currently around 2500 packages. As mentioned, most packages relate to an item available through CTAN. Examples are `pdftex` and `beamer`. A given file is in exactly one package.

### 2.1 TeX Live database

The TeX Live database, in short `tlpdb`, is a file usually located under the main installation’s root in `t1pg/texlive.tlpdb`. It is a simple text file where information is line based, and blocks (stanzas) are separated by blank lines. The structure is very similar to a Debian `Packages` file. Each stanza describes a package:

---

```
name beamer
```

```
...
```

---

```
name pdftex
```

```
...
```

---

Each non-empty line is either a `key value` pair or a file name, as we will see.

### 2.2 Package description

Each package contains various information: its name, a revision number, dependencies (`depends`), special things to be done when the package is installed (`execute`), and lists of files in three categories: runtime files (`runfiles`), binary (executable) files including scripts (`binfiles`), and documentation files (`docfiles`). The package description is also enriched with information obtained from the TeX Catalogue. A more or less complete example for a package stanza can be found in fig. 1.

---

```

name ascii-font
category Package
revision 29989
shortdesc Use the ASCII "font" in LaTeX.
longdesc The package provides glyph and font ...
longdesc ... and R.W.D. Nickalls.
execute addMap ascii.map
containersize 48984
containermd5 8e922125b755694d21b45e9644265611
doccontainersize 552
doccontainermd5 7b0c7918dadaca7665f8d1bd61677254
docfiles size=1
texmf-dist/doc/fonts/ascii-font/README.TEXLIVE
srccontainersize 4444
srccontainermd5 82f12b5dbe4107bada602b7f0dc5561
srcfiles size=5
texmf-dist/source/fonts/ascii-font/ascii.dtx
texmf-dist/source/fonts/ascii-font/ascii.ins
runfiles size=17
texmf-dist/fonts/map/dvips/ascii-font/ascii.map
texmf-dist/fonts/tfm/public/ascii-font/ASCII.tfm
texmf-dist/fonts/type1/public/ascii-font/ASCII.pfb
texmf-dist/tex/latex/ascii-font/ascii.sty
catalogue-ctan /fonts/ascii
catalogue-date 2013-04-15 01:42:14 +0200
catalogue-license lpl1
catalogue-version 2.0

```

---

**Figure 1:** Stanza for the `ascii-font` package

We will come back to this example later, after discussing the various configuration files.

### 3 Types of configuration files

Most of the files in a TeX system are normal input files. These files are searched for using the well-known Kpathsea library. Normally, only the first-found file is read (details of the file search algorithm are in the Kpathsea manual [3]). This is the normal case, and we will refer to it henceforth as the *non-stacked* case.

In contrast, a few configuration files are read in a *stacked* manner, where *all* files found by Kpathsea are read and evaluated, not just the first.

The difference can be seen in a TeX Live installation by comparing the `kpsewhich updmap.cfg` output to that of `kpsewhich -all updmap.cfg`. On my system (which is installed in `/tl/2013` and is a bit unusual with respect to `texmf-local`) I get:

---

```

$ kpsewhich updmap.cfg
/tl/2013/texmf-config/web2c/updmap.cfg
$ kpsewhich -all updmap.cfg
/tl/2013/texmf-config/web2c/updmap.cfg
/usr/local/share/texmf/web2c/updmap.cfg
/tl/2013/texmf-dist/web2c/updmap.cfg

```

---

In the non-stacked case only the first file would be read; in the stacked case, all of them.

Not many files are treated in a stacked way. In the next section we will discuss the most important

configuration files and mention for each whether it is read in a stacked or non-stacked way.

### 4 Important configuration files

The TeX Live configuration files discussed here are the most important, especially for distributors, as they need special attention. Other configuration files (there are plenty more) can be treated transparently, as they should generally work without any changes.

The configuration files we will discuss are:

`texmf.cnf` Central configuration file for path searching and parameters of the engines.

`updmap.cfg` Configuration file for font embedding from which configurations for driver programs are produced.

`fmtutil.cnf` TeX formats (and METAFONT bases).

`language.dat` Several files controlling the inclusion of hyphenation patterns in format dumps.

For each of these we give advice on what distributors can (should?) change and how they can be handled.

#### 4.1 `texmf.cnf`

The `texmf.cnf` file defines the available trees, among many other things. It has always been treated as a *stacked* configuration file—all the `texmf.cnf` files found are evaluated. This feature is used in the native `install-tl` to adjust settings via a file `texmf.cnf` at the root of the TeX Live installation.

By default the following trees are defined and used, where `R` is the root of the installation:

TEXMFDIST	files from TeX Live	<code>R/texmf-dist</code>
-----------	---------------------	---------------------------

TEXMFHOME	user tree	<code>~/texmf</code>
-----------	-----------	----------------------

TEXMFLOCAL	site-wide additions	<code>R/..../texmf-local</code>
------------	---------------------	---------------------------------

TEXMFSYSVAR	cached data	<code>R/texmf-var</code>
-------------	-------------	--------------------------

TEXMFSYSCONFIG	config. data	<code>R/texmf-config</code>
----------------	--------------	-----------------------------

TEXMFVAR	per-user cached data	<code>~/.texlive2013/texmf-var</code>
----------	----------------------	---------------------------------------

TEXMFCONFIG	per-user modified configuration data	<code>~/.texlive2013/texmf-config</code>
-------------	--------------------------------------	--

VARTEXFONTS	location of generated fonts	<code>TEXMFVAR/fonts</code>
-------------	-----------------------------	-----------------------------

In recent years, when packaging for Debian I haven't needed to change anything outside of these path definitions. In particular, distributors might want to change the definition of `TEXMFSYSCONFIG`. In Debian, we change that to `/etc/texmf` in accordance with our policies.

Another possible adjustment is adding an additional tree. In Debian, we ship TeX Live in `/usr/share/texlive` and add a tree called `TEXMFDEBIAN` in `/usr/share/texmf`, searched before `TEXMFDIST`.

To effect such changes, distributors can either patch the main `texmf.cnf` in `texmf-dist/web2c`

or add another `texmf.cnf` in one of the searchable trees. Of course, one cannot change the location of, say, `TEXMFSYS CONFIG` to a different path in a `texmf.cnf` file *within* the new location. So in Debian we patch the main configuration file to adjust *only* `TEXMFSYS CONFIG`, and add all other changes to `/etc/texmf/web2c/texmf.cnf`.

## 4.2 updmap.cfg

The `updmap.cfg` file has probably caused the most grief, so we will go to great length in the explanations.

Many of the fonts shipped in TeX Live are PostScript Type 1 fonts. TeX itself does not know anything about these fonts, and only uses the metrics (`.tfm`). Output drivers, on the other hand, need to know how the metrics are mapped to external fonts. Some notable output drivers:

`pdftex` The TeX engine with PDF output. Since producing PDF clearly needs the actual fonts, `pdftex` is also an output driver.

`dvips` A classic output driver converting `.dvi` (Device Independent) files to PostScript. Again, the fonts have to be embedded.

(x)`dvipdfm(x)` The family of dvi-to-pdf converters.

These programs support direct translation from DVI to PDF. XeTeX uses one of these in the background. Japanese users often use `dvipdfmx`, since it has good support for Japanese fonts.

(p)`xpdf` Display programs, of course need access to the fonts. `pxdvi` is `xpdf` patched for Japanese support.

Unfortunately, different drivers need the font mapping in different formats. Here is where `updmap` comes into play: It reads a list of specifications and creates configuration files for the above programs, in the necessary formats.

### 4.2.1 Different layers of configuration

The files generated by `updmap` have a long chain of provenance:

- A “font map definition” maps a `.tfm` file name to an external font with optional transformations.
- A “font map file” collects font map definitions; normally there is one font map file per package, collecting all fonts in that package.
- An “`updmap` config file” lists options and font map files.
- The “output driver configuration files” are read by the output drivers; these files are generated by `updmap`.

### 4.2.2 Configuration of fonts in updmap.cfg

`updmap`'s central configuration file is `updmap.cfg`. In former times, only the first one found by Kpathsea

was used, but now all of them are read (see below). Each `updmap.cfg` file can contain either empty lines, comment lines starting with the comment char #, or one of the following settings, in the format key value:

`dvipsPreferOutline true or false`; whether `dvips` uses bitmaps or outlines, where possible.

`dvipsDownloadBase35 true or false`; whether `dvips` embeds the standard 35 PostScript fonts.

`pdftexDownloadBase14 true or false`; whether `pdftex` embeds the standard 14 PDF fonts.

`pxdviUse true or false`; whether `updmap` controls `pxdvi`'s maps.

`kanjiEmbed,kanjiVariant` arbitrary strings, controlling kanji font embedding

`LW35 URWkb, URW, ADOBEkb, ADOBE`; file naming scheme assumed for the base PostScript fonts.

`map directives` One of `Map foo.map`,

`MixedMap bar.map`, or `KanjiMap baz.map`.

`Map` is used for fonts that are available only in PostScript format; `MixedMap` for fonts where Metafont and PostScript variants are present, and `KanjiMap` for special kanji support (see [5]).

### 4.2.3 Operation mode

As of TeX Live 2013, `updmap` reads all `updmap.cfg` files found, i.e., all the files given by `kpsewhich -all updmap.cfg`, in contrast to the former method of only reading the first one found.

We made this change for several reasons. First, it supports having the font map configuration in the same tree as the fonts themselves. Before, the activation of a map file did not survive when (re)installing a new release of TeX Live. Now, if for example `TEXMFLOCAL` contains local fonts, and they are listed in `TEXMFLOCAL/web2c/updmap.cfg`, they will automatically be picked up. A second reason is to support users without write permission to the system installation. This way, they can manage their fonts without needing a copy of the system's `updmap.cfg`.

More specifics, such as enabling and disabling of maps, can be found in the manual page of `updmap` and a blog post [2].

### 4.2.4 Recommendations for distributors

Distributors must be aware that changing the set of available fonts requires a change to one of the `updmap.cfg` files, followed by running `updmap-sys`. Otherwise, the fonts will not be available to users, even though they are present in the system. Also, distributors should *not* ship the `updmap.cfg` file included in TeX Live, since it is only valid for a full installation of TeX Live. (The TeX Live installer

itself does not install this file, but generates it from the set of installed packages.)

Since `updmap.cfg` is read in a stacked manner, changes can be localized to the tree where fonts are installed. In Debian we have one `updmap.cfg` for the TeX Live packages in `/usr/share/texlive/texmf-dist/web2c/updmap.cfg`, and one for additional font packages with files in `TEXMFDEBIAN`.

### 4.3 fmtutil.cnf

The configuration files discussed so far have been read in a stacked way; the following files are all non-stacked. To repeat that important difference, only one instance of the following files will be used, namely the one that is returned by a normal `kpsewhich` call.

`fmtutil.cnf` is the main configuration file for the `fmtutil` program, which generates format dumps for the various engines. Thus, a change in available formats needs to change `fmtutil.cnf`, and then call `fmtutil-sys`.

Fortunately, it is rare that a user wants to create his own format dumps (and such users can take care of themselves); so distributors need only make sure that the configuration file stays properly updated.

### 4.4 language.dat family

The last group of configuration files relates to the definition of hyphenation patterns. Many engines load hyphenation patterns for different languages at format dump time (see above), and proper hyphenation is possible with only those languages. These files are:

`language.dat` for L<sup>A</sup>T<sub>E</sub>X-based formats  
`language.def` for ε-T<sub>E</sub>X-based plain formats  
`language.dat.lua` for LuaT<sub>E</sub>X-based formats

The first two files are loaded at format dump time, thus a change in the available hyphenation patterns needs to (again) trigger a call to `fmtutil-sys`, best in combination with the `--byhyphen` command line option to specify explicitly the location of the hyphenation file.

The last of the three is easier, since LuaT<sub>E</sub>X loads the patterns at runtime. So no action on the side of the distributors is necessary.

## 5 Gluing it together

### 5.1 Execute statements

Many times above I have written ‘change in availability’. But how can a distributor detect such a change? The answer lies in the `execute` statements in the package stanzas, as shown in fig. 1. There are three different execute actions: one for font maps, one for formats, and one for hyphenation patterns.

### 5.1.1 Font map execute action

Activating a font can happen in three different ways, trivially corresponding to the three different map types in `updmap.cfg`:

```
execute addMap <mapname>
    Add a line 'Map <mapname>'.
execute addMixedMap <mapname>
    Add a line 'MixedMap <mapname>'.
execute addKanjiMap <mapname>
    Add a line 'KanjiMap <mapname>'.
```

For distributors, this means that part of creating the TeX packages for distribution is determining the maps to be activated from the `tlpdb`, and adding the respective lines to the appropriate `updmap.cfg` file. The semantic differences between the three invocations are explained in the `updmap` documentation.

### 5.1.2 Format execute action

The information involved in defining a format is a bit more complex than for font maps. Each `execute` statement contains again a list of `key=value` pairs, all on the same line in the `tlpdb`. The possible keys are `name`, `engine`, `patterns`, and `options`. A typical line (breaks are due to *TUGboat*):

---

```
execute AddFormat name=pdflatex engine=pdfTeX
patterns=language.dat
options="-translate-file=cp227.tcx *pdflatex.ini"
```

---

The value of `options` often contains spaces and thus needs to be quoted.

The meaning of the above line is that a line `name engine patterns options` with the respective values should be added to the `fmtutil.cnf` file (without any quotes). In the above case, that would be:

---

```
pdflatex pdfTeX language.dat
-translate-file=cp227.tcx *pdflatex.ini
```

---

### 5.1.3 Hyphenation execute action

The most complicated `execute` statement regards the activation of hyphenation patterns. As in the previous case, it is a line of `key=value` pairs. The possible keys this time are `name`, `synonyms`, `lefthyphenmin`, `righyphenmin`, `file`, `file_patterns`, and `file_exceptions`. How to generate the three language definition files (`.dat`, `.def`, `.dat.lua`) from this information is beyond the scope of this article; there are functions available in the Perl modules distributed with TeX Live (in `tlpkg/TeXLive`).

### 5.2 Distribution paradigms

When it comes to distributing such a huge piece of software, several options have been used. The first

question is perhaps *if and how* to split the full TeX Live before repackaging it for a distribution. This gives rise to choosing one of the following paradigms:

- all-or-nothing* all of TeX Live is distributed as one distribution package
- collection-splitting* one distribution package per TeX Live collection
- package-splitting* one distribution package per TeX Live package
- mixed-mode* overlapping/ad hoc splitting

The *all-or-nothing* approach has the advantage that, in principle, no changes to the various config files are needed. Just ship them as they should be and that's it. Unfortunately, this is no longer the case as soon as there are fonts shipped independently from TeX Live in the distribution. Even worse, downloading a few Gb for any update will not make the users of your distribution happy. I don't know of any distribution using this method.

The *collection-splitting* approach converts TeX Live collections into distribution packages. This approach has many advantages: first, since the content of collections do not overlap, there will be no file conflict (double inclusion) which is a basic requirement for package managers. Furthermore, TeX Live collections try to group related packages together, so users can potentially eliminate collections not of interest. Finally, the number of distribution packages is not too big. On the negative side, splitting by collection requires a bit more work on the packaging side. Debian and its derivatives (such as Ubuntu) use this approach.

The *package-splitting* approach converts each TeX Live package to one distribution package. While this is conceptually the cleanest approach, and allows for fine-grained installations, it requires a near-fully automatic packaging system due to the huge number of packages. Having thousands of distribution packages itself might be regarded as a disadvantage. Furthermore, since we do not track inter-package dependencies in TeX Live, dependencies between distribution packages will be incomplete. Distributions using this paradigm include Fedora and SuSE.

Finally, as I understand it, the *mixed-mode* paradigm is used in some BSD packaging, but I don't know the details and so cannot comment on advantages or disadvantages.

## 6 Closing

I want to close with some warnings and common pitfalls.

❖ *I've never used TeX Live and I don't know what TeX does, but I package it!*—it sounds crazy, but we have heard from people who want to package TeX without the slightest knowledge. Just say no.

❖ *Improper configuration file handling*—by far the biggest problem, and the reason I wrote this article. A common error is shipping the TeX Live `updmap.cfg` instead of generating its content based on the fonts actually installed.

❖ *What is upstream?*—since TeX Live has (approximately) one release per year and daily updates, it is rather a moving target. Build scripts that require a stable target (such as some BSD ports) need to take extra care.

❖ *Binaries and sources*—we almost never update compiled binaries after a release, but our development sources are changing continually. Thus, it's a mistake to base distribution binaries on them.

❖ *Shipping tlmgr*—distributions have their own package manager, thus subsuming the most important part of `tlmgr` functionality. Even if users are crying for it, `tlmgr` should not be used to update packages (also not by root). The only reasonable approach is to ship `tlmgr` working in *user mode* only, where it manages `TEXMFHOME`.

In addition to the above, I highly recommend creating a working installation of TeX Live and actually use it yourself; and to learn Perl, since most of the functionality of our installer and `tlmgr` are implemented in Perl modules and available in the TeX Live distribution; and finally, to contact us—we have a designated mailing list for distributors [4].

## References

- [1] CTAN (Comprehensive TeX Archive Network). <http://ctan.org>.
- [2] Internals of TeX Live 2: multi-updmap. <http://www.preining.info/blog/2013/07/internals-of-tex-live-2-multi-updmap/>.
- [3] Kpathsea manual. <http://tug.org/kpathsea>.
- [4] TeX Live distributors mailing list. <http://lists.tug.org/tldistro>.
- [5] Updmap and Kanji embedding in TeX Live. <http://tug.org/texlive/updmap-kanji.html>.

◊ Norbert Preining  
Japan Advanced Institute of  
Science and Technology  
Nomi, Ishikawa, Japan  
`norbert (at) preining dot info`  
<http://tug.org/texlive>

---

## A gentle introduction to Python<sub>TEX</sub>

Andrew Mertz and William Slough

### Abstract

The Python<sub>TEX</sub> package allows authors to combine computational and typesetting tasks by embedding Python code in <sub>TEX</sub> documents. This package allows access to many powerful Python modules, providing support for such things as symbolic mathematics, plotting, arbitrary precision numerical calculations, and networking. Python's intuitive syntax, popularity, and extensibility along with <sub>TEX</sub>'s formatting strengths make them a logical combination for programming documents. By examining a variety of examples, we will provide an overview of the capabilities and possibilities of Python<sub>TEX</sub>.

### 1 Motivation and overview

As widely appreciated by its users, <sub>TEX</sub> is a typesetting system with numerous strengths and capabilities, providing the ability to create beautiful documents. Although it provides a capability for the definition of macros, designing and implementing them can be a daunting experience, especially for non-experts.

Python [9] is a programming language which has attracted a large number of users. As a further enhancement, scientific and technical computing is supported by an extensive collection of modules and utilities. These provide capabilities for numerical integration, linear algebra, linear programming, sparse matrix manipulation, symbolic mathematics, and plotting, for example.

Python<sub>TEX</sub> [8] allows authors to combine the computational power of Python with the typesetting capabilities of <sub>TEX</sub>. This marriage of computational and typesetting worlds yields some exciting possibilities, as we intend to show in this paper.

Using the Python<sub>TEX</sub> package, Python code may be placed directly into a <sub>LAT</sub>E<sub>X</sub> document. During processing of this document—and “behind the scenes”—wherever Python code appears, a Python interpreter is executed, producing results which are then injected into the document in place of that code.

Python<sub>TEX</sub> provides a variety of macros and environments with various optional arguments. It also has a tool, `depythontex`, for creating merged documents consisting of the original <sub>LAT</sub>E<sub>X</sub> source and the Python output. This resulting document can then be processed without Python<sub>TEX</sub>. Our aim is to provide an introduction, so we limit ourselves to a small, yet powerful, subset of Python<sub>TEX</sub>.

### 2 Getting started

To begin, some installation will probably be needed. The Python<sub>TEX</sub> package can be found at CTAN and installed, for example, by use of a package manager such as <sub>TEX</sub> Live's `tlmgr`.

In addition, a Python installation is needed, with Python 2.7 being the recommended version. The exact details of how this is done depend on your system, but one relatively simple way to obtain it is to download and install Anaconda [1]. (We are grateful to Richard Koch, from whom we learned about this resource.) Anaconda is a free Python distribution which supports GNU/Linux, Windows, and Mac OS X.

Not surprisingly, a document to be processed with Python<sub>TEX</sub> will need to indicate this in its preamble:

```
\usepackage{pythontex}
```

A number of optional arguments can be supplied, though none of these are needed for what is being described in this introduction. For full details, refer the Python<sub>TEX</sub> documentation.

Three steps are needed to process a Python<sub>TEX</sub> document: first, <sub>LAT</sub>E<sub>X</sub>, then Python<sub>TEX</sub>, and finally <sub>LAT</sub>E<sub>X</sub>. (Various engines are possible, including pdf<sub>LAT</sub>E<sub>X</sub>, Lua<sub>LAT</sub>E<sub>X</sub>, and X<sub>F</sub><sub>LAT</sub>E<sub>X</sub>.) For example, the document `sample.tex` could be processed with the following sequence of commands.

```
pdflatex -interaction nonstopmode \
-draftmode sample.tex
pythontex sample.tex
pdflatex sample.tex
```

The first step extracts the Python code from the document (to the file `sample.pytxcode`). In the second step, this code is given to the Python interpreter and the results are saved to a variety of files within the subdirectory `pythontex-files-sample`. In the final step, the results from Python are merged with the original document.

### 3 Fundamental Python<sub>TEX</sub>

We begin our exploration by considering two macro commands intended for inline code: `\py` and `\pyc`. To use `\py`, a single-line Python expression is supplied as an argument:

```
\py{expression}
```

In response, the Python interpreter evaluates the expression, computes the result and stores the result as a string. This string then takes the place of the `\py` command which is subsequently typeset. For example, `\py{2**26}` produces 67108864, the value of  $2^{26}$ .

```
\begin{pycode}
print(r"\begin{tabular}{c|c}")
print(r"$m\$ & $2^m \$ \\ \hline")
print(r"%d & %d \$ \\ % (1, 2**1)")
print(r"%d & %d \$ \\ % (2, 2**2)")
print(r"%d & %d \$ \\ % (3, 2**3)")
print(r"%d & %d \$ \\ % (4, 2**4)")
print(r"\end{tabular}")
\end{pycode}
```

```
\begin{tabular}{c|c}
$m\$ & $2^m \$ \\ \hline
1 & 2 \\
2 & 4 \\
3 & 8 \\
4 & 16 \\
\end{tabular}
```

**Figure 1:** Generation of a table using the `pycode` environment; resulting L<sup>A</sup>T<sub>E</sub>X code shown on the right

A related macro is `\pyc`, which has a subtle yet important difference. To use `\pyc`, a single-line Python statement is given:

```
\pyc{statement}
```

Here, the given statement is executed and anything *printed* by it takes the place of the `\pyc` command, which is subsequently formatted by TeX. As an example, `\pyc{print(2**26)}` yields 67108864.

At this point, it may appear that `\pyc` does not add much beyond what `\py` provides. However, this is far from the truth, as we intend to show. But before we can illustrate the power of `\pyc`, we need to discuss some additional features of PythonTeX and Python itself.

An analog of the `\pyc` command is the `pycode` environment:

```
\begin{pycode}
Python statements
\end{pycode}
```

Unlike `\pyc`, this environment allows multiple-line Python statements to appear. As with `\pyc`, all of the printed output gets inserted into the document at that point, to be subsequently typeset.

To illustrate, consider Figure 1. This example shows how a `pycode` environment may be utilized to generate a table of values consisting of the pairs  $(m, 2^m)$ , using Python to generate powers of 2. Although this example does not use sophisticated Python code, it does illustrate an important idea: the code embedded within a `pycode` environment should generate the appropriate typesetting markup to achieve the desired effect. The typeset result of this code follows:

$m$	$2^m$
1	2
2	4
3	8
4	16

Like the C language, Python uses escape sequences (such as `\\"`, `\n`, `\f`, etc.) to describe certain characters. Python uses “raw” strings, denoted with

an `r` prefix, to disable escape sequences, allowing their content to appear verbatim. So, for example,

```
print("\\")
```

would output a single `\`, whereas

```
print(r"\\")
```

outputs `\`. Since Python is being used to generate L<sup>A</sup>T<sub>E</sub>X code, the use of raw strings is often needed.

Another feature of Python being used here is the `%` operator, which is used for formatting strings. The `%d` specifies a placeholder, to be filled by a decimal integer value obtained from an expression. For example,

```
"%d and %d" % (3, 2**3)
```

produces the string `"3 and 8"`. These two features, raw strings and formatted strings, allow for the understanding of the example shown in Figure 1.

With this background, we can improve the code by introducing a loop which iterates over the desired values of  $m$ . The Python `range` function generates a list of integer values over a specified interval. Given integers  $l$  and  $h$ , `range(l, h)` generates a list of the integers from  $l$  to  $h - 1$ . The example code shown in Figure 2 produces the same tabular output as before, but adds flexibility. The multiple assignment

```
lo, hi = 1, 4
```

allows an arbitrary range of table values to be specified; the `for` loop generates the table entries, one row per iteration. As a side note, Python provides arbitrary precision integer arithmetic; thus, tables of powers of 2 involving a large number of digits can be produced by simply adjusting `lo` and `hi`. For example, with `lo = 100` and `hi = 102` the following table is produced:

	$2^m$
100	1267650600228229401496703205376
101	2535301200456458802993406410752
102	5070602400912917605986812821504

Python provides the ability to define functions as a way to promote program modularity. By defining a function within a `pycode` environment, we can

```
\begin{pycode}
lo, hi = 1, 4
print(r"\begin{tabular}{c|c}")
print(r"$m$ & $2^m$ \\ \hline")
for m in range(lo, hi + 1):
    print(r"%d & %d \\ % (m, 2**m)")
print(r"\end{tabular}")
\end{pycode}
```

**Figure 2:** Generation of a table using a loop

subsequently evaluate it using `\py` or a similar command. To illustrate this capability, consider the well-known Fibonacci sequence

$$0, 1, 1, 2, 3, 5, 8, 13, \dots$$

defined by  $F_0 = 0$ ,  $F_1 = 1$ , and  $F_k = F_{k-2} + F_{k-1}$  for  $k \geq 2$ . Figure 3 provides the definition of a function which computes the  $n^{\text{th}}$  Fibonacci number.

Since `range(n)` generates the list of integers from 0 through  $n - 1$ , the `for` loop in `fib` makes exactly  $n$  iterations. One way to understand this function is to imagine scrolling across the Fibonacci sequence with a window capable of exposing two adjacent numbers in the sequence. Initially, the window is positioned over  $F_0$  and  $F_1$ ; to expose  $F_n$  the window is advanced  $n$  times.

---

```
\begin{pycode}
def fib(n):
    a, b = 0, 1
    for i in range(n):
        a, b = b, a + b
    return a
\end{pycode}
```

**Figure 3:** A Python function to compute the  $n^{\text{th}}$  Fibonacci number

---

With this function definition in place, we can use `\py` to evaluate arbitrary values of the Fibonacci sequence. For example, `\py{fib(10)}` produces 55, the value of  $F_{10}$ . As we saw earlier, arbitrary precision arithmetic is available “out of the box”. so we can use this function with equal ease on larger values. For example, the claim

$$F_{100} = 354224848179261915075$$

is produced by `$F_{100} = \py{fib(100)}$`.

#### 4 Getting fancier

So far, we have considered just two commands, `\py` and `\pyc`, and one environment, `pycode`. Even with this limited collection, we have many possibilities.

However, an awareness of a few more features of PythonTeX will allow for improved processing times and additional flexibility. One such feature is

the concept of *sessions*. Without naming sessions, as we have done up to this point, all Python code runs sequentially in one default session. This can have several advantages. For example, variables and functions defined in one `pycode` environment are available to subsequent `pycode` environments, which avoids redundant code.

On the other hand, running all Python code in one session has the disadvantage that multiple cores are not utilized. As the amount of Python code in a document increases, there is a time penalty to be paid. By utilizing multiple Python sessions, code can be executed in parallel, providing a welcome speedup. All of the PythonTeX commands and environments provide for an optional session name. If no such specification appears, it runs in the default session. Judicious use of sessions can have dramatic improvement in processing time.

Another speed-related benefit of sessions derives from the fact that Python will run only for those sessions where the code has recently changed. This allows the user to place time-intensive Python code in their own sessions—and if that code doesn’t need to be modified, then it is executed just once.

Multiple sessions are independent. They do not share variables or function definitions, for example. Sometimes this will be exactly what we want, but other times not. It is for these latter situations that the `pythontexcustomcode` environment exists. This environment allows a code block to be specified, which is then made available to *all* sessions, irrespective of session name. Let’s look at an example, shown in Figure 4, to explore this idea further.

---

```
\begin{pythontexcustomcode}{py}
def makeTable(lo, hi):
    print(r"\begin{tabular}{c|c}")
    print(r"$m$ & $2^m$ \\ \hline")
    for m in range(lo, hi + 1):
        print(r"%d & %d \\ % (m, 2**m)")
    print(r"\end{tabular}")
\end{pythontexcustomcode}
```

**Figure 4:** Informing all sessions how to generate a table of powers of 2

---

As a small detail, we first note that the custom code in this example specifies `py`, which indicates the `py` *family* of commands and environments to which it applies. As an introduction to PythonTeX, we have chosen to focus exclusively on the `py` family, but more sophisticated uses of PythonTeX may benefit from other families of commands. Full details are given in the PythonTeX manual.

Notice that the custom code provided here is simply an abstraction based on the earlier example from Figure 2. In the present case, starting and ending rows can be specified. So,

```
\pyc{makeTable(1, 4)}
```

would generate a table of powers with  $m$  between 1 and 4, whereas

```
\pyc{makeTable(4, 10)}
```

would generate a similar table, with  $m$  between 4 and 10. Using independent sessions would allow for potential speedup:

```
\pyc[one]{makeTable(1, 4)}
\pyc[two]{makeTable(4, 10)}
```

The session names, `one` and `two`, are arbitrary. Admittedly, the time gains for this example are likely to be negligible, but these examples were chosen for their simplicity and ability to illustrate sessions.

As a further illustration, we can extend our code so that it generates tables of arbitrary functions. To do this, we include two additional parameters: one to specify the function and one for the desired table heading. These two parameters, `f` and `hd`, appear in the revised version shown in Figure 5.

---

```
\begin{pythontexcustomcode}{py}
def makeTable2(lo, hi, f, hd):
    print(r"\begin{tabular}{c|c}")
    print(r"$m$ & %s \\ \hline" % hd)
    for m in range(lo, hi + 1):
        print(r"%d & %d \\ % (m, f(m))")
    print(r"\end{tabular}")
\end{pythontexcustomcode}
```

**Figure 5:** Informing all sessions how to generate a table for an arbitrary function

---

With this abstraction, we can produce a portion of the Fibonacci sequence displayed as a table, using the command

```
\pyc{makeTable2(4, 8, fib, "F_m")}
```

This call produces the table:

$m$	$F_m$
4	3
5	5
6	8
7	13
8	21

Python has a wealth of predefined functions, made available from its library of modules. These can be accessed with an appropriate `import` statement. For example, to make the factorial function available to all sessions, we could write:

```
\newif\ifprime \newif\ifunknown % booleans
\newcount\n \newcount\p \newcount\d
\newcount\a % integer variables
\def\primes#1{2,~3% assume #1 is at least 3
\n=#1 \advance\n by-2 % n more to go
\p=5 % odd primes starting with p
\loop\ifnum\n>0
    \printifprime\advance\p by2 \repeat}
% we will invoke \printp if p is prime
\def\printp{,
    \ifnum\n=1 and^~fi % "and^~" precedes last value
    \number\p \advance\n by -1 }
\def\printifprime{\testprimality
    \ifprime\printp\fi}
\def\testprimality{\d=3 \global\primetrue
\loop\trialdivision
    \ifunknown\advance\d by2 \repeat}
\def\trialdivision{\a=\p \divide\a by\d
\ifnum\aa>\d \unknowntrue\else\unknownfalse\fi
\multiply\aa by\d
\ifnum\aa=\p \global\primefalse\unknownfalse\fi}
```

**Figure 6:** Knuth's code for generating prime numbers (editorial changes to line breaks and comments)

---

```
\begin{pythontexcustomcode}{py}
from math import factorial
\end{pythontexcustomcode}
```

With this import in effect, the table generation call  
`\pyc{makeTable2(10, 17, factorial, "m!")}`  
produces the following result:

$m$	$m!$
10	3628800
11	39916800
12	479001600
13	6227020800
14	87178291200
15	1307674368000
16	20922789888000
17	355687428096000

These examples illustrate how a wide assortment of tables can be generated and typeset with relatively little effort.

## 5 A table of primes

A recent post appeared on T<sub>E</sub>X Stack Exchange [12] asking how one might generate a collection of prime numbers using L<sup>A</sup>T<sub>E</sub>X. Among the responses was the comment that Knuth himself had provided code for this [5, p. 218]. Figure 6 shows his implementation.

Knuth's code is not for the timid — he gives it his most difficult rating, a double dangerous bend. Some years later, Roegel [10] explains this 16-line macro, in the span of four pages, providing further evidence of the subtlety involved in its implementation.

```
\begin{pythontexcustomcode}{py}
from sympy import prime

def generatePrimes(n):    # Assume n >= 3
    for i in range(1, n):
        print("%d, " % prime(i))
    print("and %d%" % prime(n))
\end{pythontexcustomcode}
```

**Figure 7:** How to generate the first  $n$  prime numbers using Python $\text{\TeX}$

For comparison, we show an equivalent using Python $\text{\TeX}$  in Figure 7. This code hides the computational details within a function `prime` which computes the  $i^{\text{th}}$  prime number. To use this we might write

Thirty primes: \pyc{generatePrimes(30)}.

which generates the following output:

Thirty primes: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, and 113.

We think this provides a nice illustration of the suitability of Python $\text{\TeX}$  for documents that can benefit from programmed output, especially for those  $\text{\TeX}$  users who do not intend to become highly skilled in the art of macro writing.

This function has a small subtlety. The final line of output produced by `generatePrimes(30)` is  
and 113%

followed by a newline supplied by the `print` statement. (A single % is produced by the Python specifier `%%`.) We want the terminating % to appear so `generatePrimes` can be included in the context of other text—such as the terminating period in the preceding example.

## 6 Python data structures

Python has several useful native data structures, such as lists, sets, and dictionaries. Furthermore, list objects have methods that allow them to be treated as stacks or queues. This section demonstrates the basic syntax for working with lists and dictionaries. These capabilities will be needed for examples in subsequent sections.

A list is an integer-indexed sequence of items, possibly of different types. In other words, a list can contain a mixture of numbers, strings, and other objects. Items can be removed from, added to, or retrieved from lists. Lists can be defined with a comma separated sequence of items within square brackets. For example:

```
\pyc{myList=["Iris", "Azalea", "Rose"]}
```

defines a new list named `myList` that contains three strings. The indexing operator, [ ], can be used to retrieve items from a list. Lists in Python are indexed from zero. Thus, `\pyc{myList[0]}` is “Iris” and `\pyc{myList[1]}` is “Azalea”.

A `for` loop can be used to iterate over any sequence. For example:

```
\begin{pycode}
for name in myList:
    print(name)
\end{pycode}
```

becomes “Iris Azalea Rose”.

The `enumerate` function may be used if both the index and the value of an item are needed. For example,

```
\begin{pycode}
for index, name in enumerate(myList):
    print(r"%d: %s" % (index, name))
\end{pycode}
```

prints both the index and value each item in the list, that is, “0: Iris 1: Azalea 2: Rose”.

While lists are indexed by integers, dictionaries can be indexed by any immutable type, often strings. Dictionaries can be thought of as sets of key-value pairs where the keys are unique. Dictionaries can be defined with a comma-separated sequence of key-value pairs within braces. For example,

```
\begin{pycode}
myDict = {"Illinois": "Violet",
          "New Mexico": "Yucca",
          "Indiana": "Peony"}
\end{pycode}
```

defines a dictionary named `myDict` with three entries. The indexing operator, [ ], can be used to retrieve values from a dictionary with keys used as the index. So, `\pyc{myDict["Illinois"]}` yields “Violet”.

## 7 Symbolic mathematics

While the preceding sections attempt to be a relatively simple introduction to Python and Python $\text{\TeX}$ , the remaining sections are more complex. The goal is to demonstrate some of the cases where Python $\text{\TeX}$  can provide useful capabilities that would be difficult using only  $\text{\TeX}$ .

Aside from the built-in functionality, Python has many powerful modules for mathematics. For instance, SciPy [11] is a rich collection of open source Python-based software for science, engineering, and mathematics. SciPy includes SymPy [13], a Python module for symbolic mathematics with features similar to other computer algebra systems like Mathematica [15] and Maple [6]. Using SymPy with  $\text{\TeX}$

allows mathematics not only to be beautifully typeset, but also to be manipulated and evaluated.

While a full exploration of SymPy is beyond the scope of this paper, an introduction to its features will be provided to highlight how well it can integrate with TeX through PythonTeX. For more information see the SymPy tutorial [14].

Like other modules, SymPy must be imported before use. For example:

```
\begin{pythontexcustomcode}{py}
from sympy import *
\end{pythontexcustomcode}
```

imports all of the functions and objects defined in the `sympy` module. SymPy defines numerous functions, many with common names such as `sin`, `cos`, and `var`. This can interfere with other modules, such as the plotting module `pylab`, which will be discussed in the next section. Thus, it can be safer to import the module with an `import sympy` statement:

```
\begin{pythontexcustomcode}{py}
import sympy
\end{pythontexcustomcode}
```

or to import inside of a session that will be used only for SymPy:

```
\begin{pycode}[sympy-session]
from sympy import *
\end{pycode}
```

PythonTeX also defines macros (`sympy`, `sympyc`) and related environments (such as `sympycode`) which simplifies this process.

The `var` function can be used to declare symbolic variables. For example, `\pyc{var("x, y")}` declares two symbolic variables `x` and `y`. Such variables can be used to form symbolic expressions that can be manipulated by SymPy. The examples in this section assume this variable declaration has been performed and that a

```
from sympy import
```

statement was used.

The `latex` function returns the L<sup>A</sup>T<sub>E</sub>X code representing a given SymPy expression. For example,

```
$\pyc{\text{\texttt{ latex}}((x + y)^{**5})}$
```

yields  $(x + y)^5$ . Without the `latex` function,

$\$\\pyc{(x + y)^{**5}}\$$

simply becomes  $(x + y) * ^5$ , since the Python exponentiation operator is not converted into its L<sup>A</sup>T<sub>E</sub>X equivalent. This difference is more pronounced as the expressions become more complex. Also, L<sup>A</sup>T<sub>E</sub>X code is just text to SymPy and cannot be manipulated as symbolic expressions can. It is important to remember to use the `latex` function only when typesetting SymPy expressions.

Symbolic expressions can be saved in ordinary Python variables. For example,

```
\pyc{z = (x + y)**5}
```

stores an expression in the variable `z`. SymPy has many functions for manipulating symbolic expressions, including: `simplify`, `factor`, `collect`, and `expand`. These functions are applied to symbolic expressions like any other function call. For instance, `z` can be expanded with

```
\[ \pyc{\text{\texttt{ latex}}(\text{\texttt{ expand}}(z)) + ". "} \]
```

which becomes

$$x^5 + 5x^4y + 10x^3y^2 + 10x^2y^3 + 5xy^4 + y^5.$$

Figure 8 shows a more complex example that forms a table of binomials and their expansions. This example also shows how to build a list of expressions and iterate over them.

SymPy exports many trigonometric and calculus related functions, some of which are illustrated in Figure 9. Several SymPy objects, such as limits, integrals, sums and products are not evaluated automatically. To evaluate such objects the `doit` method may be used.

SymPy also includes combinatorial functions, for such things as Bernoulli, Catalan, Fibonacci, and Stirling numbers. Figure 10 details the creation and use of a function for formatting tables of Stirling numbers of the second kind, denoted  $\{n\}_k$ , which counts the number of ways to partition a set of  $n$  elements into  $k$  nonempty subsets. This function relies on a macro to format Stirling numbers, for example:

```
\usepackage{amsmath}
\newcommand{\Stirling}[2]{
\begin{Bmatrix} #1 \\ #2 \end{Bmatrix}}
```

```
\begin{pycode}
# Start with an empty list
binomials = []

# Add a few symbolic expressions to the list
for m in range(2, 6):
    binomials.append((x + y)**m)

# Start an align environment to hold the
# results
print(r"\begin{align*}")

# Add the original expressions and their
# expansions to the table
for expr in binomials:
    print(r"%s &= %s\\ % (latex(expr),
        latex(expand(expr)))")

# End the align environment
print(r"\end{align*}")
\end{pycode}
```

$$\begin{aligned}(x+y)^2 &= x^2 + 2xy + y^2 \\ (x+y)^3 &= x^3 + 3x^2y + 3xy^2 + y^3 \\ (x+y)^4 &= x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4 \\ (x+y)^5 &= x^5 + 5x^4y + 10x^3y^2 + 10x^2y^3 + 5xy^4 + y^5\end{aligned}$$

**Figure 8:** Binomial expansions

```
\begin{pycode}
# Define a list of functions
functions = [sin(x), cos(x), tan(x)]

print(r"\begin{align*}")

# For each function build a symbolic expression of its derivative and integral
for f in functions:
    d = Derivative(f, x)
    i = Integral(f, x)

    # Print a row in the table displaying the derivative and integral. Note the
    # use of the "doit" method to evaluate the derivative and integral. Also
    # string concatenation, +, is used to join strings in this example.
    print(latex(d) + "&= " + latex(d.doit()) + "&" +
          latex(i) + "&= " + latex(i.doit()) + r"\\")

print(r"\end{align*}")
\end{pycode}
```

$$\begin{array}{ll} \frac{d}{dx} \sin(x) = \cos(x) & \int \sin(x) dx = -\cos(x) \\ \frac{d}{dx} \cos(x) = -\sin(x) & \int \cos(x) dx = \sin(x) \\ \frac{d}{dx} \tan(x) = \tan^2(x) + 1 & \int \tan(x) dx = -\frac{1}{2} \log(\sin^2(x) - 1) \end{array}$$

**Figure 9:** Building a table of derivatives and integrals

```
\begin{pycode}
# Import a function for computing Stirling numbers
from sympy.functions.combinatorial.numbers import stirling

# Define a function to print a table of all of the Stirling numbers for sets
# of size 1 to maxN.
def stirlingTable(maxN):

    # Print the start of the table using a triple quoted string. Triple quoted
    # strings can span lines and are useful when including long strings.
    print(r"""\\begin{displaymath}
\\begin{array}{c|*{8}{c}} \hline
\multicolumn{8}{c}{\\textbf{Stirling's Triangle for Subsets}} \\\hline
n & \left\{ \begin{matrix} n \\ 1 \end{matrix} \right\} & \left\{ \begin{matrix} n \\ 2 \end{matrix} \right\} & \left\{ \begin{matrix} n \\ 3 \end{matrix} \right\} & \left\{ \begin{matrix} n \\ 4 \end{matrix} \right\} & \left\{ \begin{matrix} n \\ 5 \end{matrix} \right\} & \left\{ \begin{matrix} n \\ 6 \end{matrix} \right\} & \left\{ \begin{matrix} n \\ 7 \end{matrix} \right\} & \left\{ \begin{matrix} n \\ 8 \end{matrix} \right\} \\
\hline
1 & 1 & & & & & & & \\
2 & 1 & 1 & & & & & & \\
3 & 1 & 3 & 1 & & & & & \\
4 & 1 & 7 & 6 & 1 & & & & \\
5 & 1 & 15 & 25 & 10 & 1 & & & \\
6 & 1 & 31 & 90 & 65 & 15 & 1 & & \\
7 & 1 & 63 & 301 & 350 & 140 & 21 & 1 & \\
8 & 1 & 127 & 966 & 1701 & 1050 & 266 & 28 & 1 \\
\hline
\end{array} \\end{displaymath}""")

    # Print each of the column headings using the previously defined Stirling
    # macro.
    for k in range(1, maxN + 1):
        print(r" & \\Stirling{n}{%d} " % k)

    # Add some phantom space so the braces are not touching the hlines.
    print(r"\vphantom{\\parbox[c][7ex]{0in}{}} \\\hline")

    # Start each row with the current n value
    for n in range(1, maxN + 1):
        print("%d" % n)

    # Add each of the Stirling numbers to the row
    for k in range(1, n + 1):
        print(" & %d" % stirling(n, k))

    # End the row
    print(r"\\")

    # End the table
    print(r"\hline \\end{array}\\end{displaymath}")

stirlingTable(8)
\end{pycode}
```

Stirling's Triangle for Subsets								
$n$	$\left\{ \begin{matrix} n \\ 1 \end{matrix} \right\}$	$\left\{ \begin{matrix} n \\ 2 \end{matrix} \right\}$	$\left\{ \begin{matrix} n \\ 3 \end{matrix} \right\}$	$\left\{ \begin{matrix} n \\ 4 \end{matrix} \right\}$	$\left\{ \begin{matrix} n \\ 5 \end{matrix} \right\}$	$\left\{ \begin{matrix} n \\ 6 \end{matrix} \right\}$	$\left\{ \begin{matrix} n \\ 7 \end{matrix} \right\}$	$\left\{ \begin{matrix} n \\ 8 \end{matrix} \right\}$
1	1							
2	1	1						
3	1	3	1					
4	1	7	6	1				
5	1	15	25	10	1			
6	1	31	90	65	15	1		
7	1	63	301	350	140	21	1	
8	1	127	966	1701	1050	266	28	1

**Figure 10:** Building a table of Stirling numbers

```
\begin{pycode}
from pylab import *

# Define f(t), the desired function to plot
def f(t):
    return cos(2 * pi * t) * exp(-t)

# Generate the points (t_i, y_i) to plot
t = linspace(0, 5, 500)
y = f(t)

# Begin with an empty plot, 5 x 3 inches
clf()
figure(figsize=(5, 3))

# Use TeX fonts
rc("text", usetex=True)
rc("font", family="serif")

# Generate the plot with annotations
plot(t, y)
title("Damped exponential decay")
text(3, 0.15, r"$y = \cos(2\pi t) e^{-t}$")
xlabel("time (s)")
ylabel("voltage (mV)")

# Save the plot as a PDF file
savefig("myplot.pdf", bbox_inches="tight")

# Insert LaTeX code to include the plot.
print(r"\begin{center}")
+ r"\includegraphics[width=\textwidth]{myplot.pdf}"
+ r"\end{center}")
\end{pycode}
```

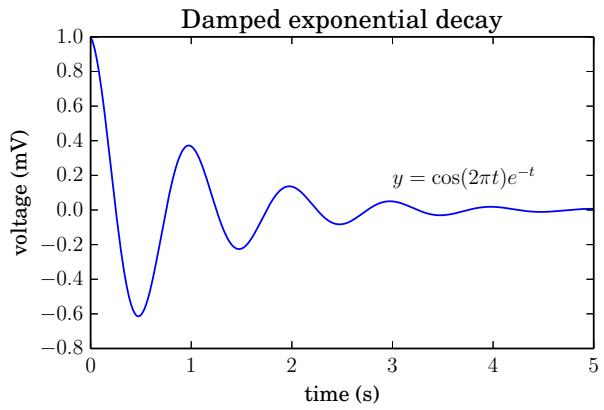


Figure 11: Plotting a function with Matplotlib

## 8 Plotting with Matplotlib

Matplotlib is a two-dimensional Python plotting library with an object-oriented interface and a set of functions similar to MATLAB [7]. To use it, the `pylab` module must be imported. Figure 11 shows an example of plotting a function with annotations. This example is inspired by a plot from the Matplotlib gallery [4] which contains many examples and tutorials. Such plots are desirable since they can use fonts which blend with the rest of the document.

Matplotlib's plots are saved in an external file such as a PDF, which can then be included in the current document. However, this can be problematic as the file may not exist the first time the document is processed. To avoid this problem, the Python code generates the required `\includegraphics` statement. In this way, the `\includegraphics` is not present the first time TeX processes the document, but is on subsequent processing.

## 9 Web services

There are many powerful and freely available web services which return JSON (Java Script Object Notation) or another easily parsed format. Python's excellent parsing and networking libraries make using such services relatively simple. Furthermore, some web services have Python modules made specifically for them.

Accessing such a web service typically requires some authorization. This may involve requesting an account with the service provider and agreeing to their terms of service. Often a key is provided to identify a client to the service and this key must be presented each time the service is used. Using the service can normally be broken into following tasks.

- Encoding the request as a URL
- Fetching the URL
- Parsing the response
- Using the result

```
\begin{pycode}
# Import functions to save the contents of a URL to a file and
# encode a dictionary as a string suitable for URL requests
from urllib import urlretrieve, urlencode

# See Google's documentation for service usage information and examples
# https://developers.google.com/maps/documentation/staticmaps/
def showGoogleMap(address, filename, zoom=10, width=640, height=680):
    # Build a dictionary with the key-value pairs required by the service
    query = {"key":googleKey, "center": address, "zoom": zoom,
              "size": "%dx%d" % (width, height), "sensor": "false"}

    # Convert the query into a url
    url = "https://maps.googleapis.com/maps/api/staticmap?" + urlencode(query)

    # Save the image to a file and include it in the document
    urlretrieve(url, filename + ".png")
    print(r"\begin{center}\includegraphics[width=0.45\textwidth]{%s}\end{center}" %
          % filename)
\end{pycode}
```

Figure 12: Displaying a map of Tokyo using Google’s Static Maps web service

The code of Figure 12 illustrates how the Google Static Maps API [3] can be used. This web service returns a PNG image of a map centered at a given address. The `showGoogleMap` function defined in this example has default values for the `zoom`, `width`, and `height` arguments. As a result, when invoking this function these arguments do not need to be specified. The results of invoking

```
showGoogleMap("Tokyo", "tokyoMap")
```

are shown in Figure 13.

To use the Google Static Maps service, an API key is needed. Such keys can be created with the Google APIs console (<https://code.google.com/apis/console>). The examples assume such a key has been stored in a global variable `googleKey`. For instance:

```
\begin{pycode}
googleKey = "Put API Key Here"
\end{pycode}
```

As a second example, Google’s “URL Shortener” [2] web service takes long URLs and converts them into URLs with fewer characters, yielding links that can be easier to share. This service sends and receives data as JSON (a simple plain text format for transmitting information as key-value pairs). This format has dictionaries, lists, numbers, and strings for data types. Figure 14 shows an example of what JSON looks like, while Figure 15 shows a sample response from the URL Shortener. Note the similarity to the declaration of a Python dictionary. See Figure 16 for the details of using this service.

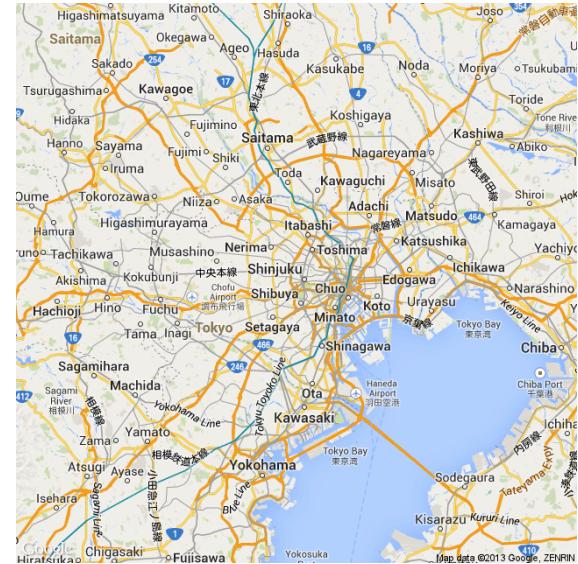


Figure 13: Map output from Static Maps service

## 10 Conclusions

The ability to include arbitrary computations within a L<sup>A</sup>T<sub>E</sub>X document holds much appeal. As a programming language, Python has relative simplicity with broad expressive power. The examples presented in this paper provide a glimpse of what is possible with this software combination.

```
{
  "debug": "on",
  "window": {
    "title": "Main View",
    "width": 640,
    "height": 480
  },
  "image": {
    "src": "Images/Icon.png",
    "hOffset": 10,
    "vOffset": 10
  }
}
```

**Figure 14:** JSON sample

```
{
  "kind": "urlshortener#url",
  "id": "http://goo.gl/fbsS",
  "longUrl": "http://www.google.com/"
}
```

**Figure 15:** Response for a successful use of the URL Shortener API

```
\begin{pycode}
# A function and object for fetching URLs and
# customizing requests
from urllib2 import urlopen, Request

# JSON/Python conversion functions
from json import load, dumps

def shortenURL(longURL):
  # The base URL for shortening requests
  url = ("https://www.googleapis.com/" +
         "urlshortener/v1/url")

  # For this service the query is sent as JSON.
  # So the query is converted to JSON and the
  # content type is set in the request's header.
  query = dumps({"longUrl": longURL,
                  "key": googleKey})
  request = Request(url, query,
                     {"Content-Type": "application/json"})

  # Fetch the request and parse the returned JSON
  result = load(urlopen(request))

  # Retrieve the shortened URL from the result
  shortURL = result["id"]

  # Add the shortened URL to the document
  print(r"\url{\%s}%" % shortURL)

shortenURL("http://mirror.ctan.org/macros/latex/" +
           "contrib/pythontex/pythontex.pdf")
\end{pycode}
```

**Figure 16:** Shortening a long URL to <http://goo.gl/sfT8S5>

## References

- [1] Continuum Analytics. Anaconda. <http://store.continuum.io/cshop/anaconda/>.
- [2] Google. The URL Shortener API. <https://developers.google.com/url-shortener/>.
- [3] Google. The Google Static Maps API. <https://developers.google.com/maps/documentation/staticmaps/>.
- [4] John Hunter. Matplotlib gallery. <http://matplotlib.org/gallery.html>.
- [5] Donald E. Knuth. *The T<sub>E</sub>Xbook*. Addison-Wesley Professional, 1984.
- [6] Maplesoft. Maple. <http://www.maplesoft.com/products/maple/>.
- [7] MathWorks. Matlab. <http://www.mathworks.com/products/matlab/>.
- [8] Geoffrey Poore. PythonT<sub>E</sub>X. <http://www.ctan.org/pkg/pythontex>.
- [9] Python Software Foundation. Python. <http://python.org>.
- [10] Denis Roegel. Anatomy of a macro. *TUGboat*, 22:78–82, 2001. <http://tug.org/TUGboat/tb22-1-2/tb70roeg.pdf>.
- [11] SciPy Developers. SciPy. <http://scipy.org>.
- [12] T<sub>E</sub>X Stack Exchange. How to produce a list of prime numbers in L<sub>E</sub>T<sub>E</sub>X. <http://goo.gl/903u75>.
- [13] SymPy Development Team. SymPy. <http://sympy.org>.
- [14] SymPy Development Team. Sympy tutorial. <http://docs.sympy.org/latest/tutorial/>.
- [15] Wolfram Research. Mathematica. <http://www.wolfram.com/mathematica/>.
  - ◊ Andrew Mertz and William Slough  
Department of Mathematics and  
Computer Science  
Eastern Illinois University  
Charleston, IL 61920  
aemertz (at) eiu dot edu,  
waslough (at) eiu dot edu

## Online publishing via pdf2htmlEX

Lu Wang and Wanmin Liu

### Abstract

The Web has long become an essential part of our lives. While web technologies have been actively developed for years, there is still a large gap between web and traditional paper publishing. For example, the PDF format, the *de facto* standard for publishing, is not supported in the HTML standard; and the most powerful typesetting system, **T<sub>E</sub>X**, cannot be integrated perfectly.

Despite of the long history of people trying to convert **T<sub>E</sub>X** or PDF into HTML, some are focused on only a small fraction of features, *e.g.* text, formulas or images; some are too old to support new features in the HTML standard such as font embedding or linear transformations (*e.g.* rotation); some display everything in images at the cost of larger sizes.

In this article, while we survey and compare existing methods of publishing **T<sub>E</sub>X** or PDF documents online, a new approach is attempted to attack this issue. We introduce an open source program, called pdf2htmlEX, which is a general PDF to HTML converter and publishing tool with high fidelity. It presents PDF elements with corresponding native HTML elements, in order to achieve high accuracy and small size. The flexible design also makes it useful for a variety of use cases in online publishing. Obviously **T<sub>E</sub>X** users can immediately benefit with zero learning cost, just like dvipdf while people were still using DVI. More information is available at the home page:

<https://github.com/coolwanglu/pdf2htmlex>

### 1 Introduction

 Arguably, for many people the World Wide Web *is* the Internet. Indeed, web technologies have been so actively developed in the past few years, nowadays web pages far surpass plain text and images. HTML5 brings audio, video, 3D graphics and many other rich features; CSS3 defines brand new visual effects, and JavaScript allows different kinds of user interactions. Modern web browsers are literally operating systems, and the boundary between web apps and local software has been blurred. Today, we can access the WWW with all kinds of devices such as watches, phones, tablets, computers and even glasses. It has become an essential part of our lives.

The web technologies provide brand new user experiences compared to traditional media. Taking Wikipedia as an example, it has *rich contents*: inside

an article, besides plain text, there are often images, animations, audio and video that are relevant to the topic; it is *well organized*: users may jump to relevant articles by clicking links; it is *interactive*: users may create or edit an article; it is *personalized*: the appearance of the web site respects users' preferences such as language, theme or format; it is *social*: users may leave comments and have discussions regarding an article.

Compared with traditional publishing media, it is more convenient and easier for users to obtain, view and share the contents. While most features in HTML are targeting visual effects, multimedia and rich Internet applications, there is still a large gap between the Web and traditional publishing. Many existing publishing technologies cannot be perfectly integrated online—especially two of them focused on in this article, PDF and **T<sub>E</sub>X**, which are the most popular format and typesetting system respectively.

**PDF** The Portable Document Format, developed by Adobe, is one of the most popular formats for digital documents. PDF is known for its wide support of different types of fonts, encodings, raster images, vector graphics, and many other features from pre-press processing to user interaction. It is widely supported in different operating systems and devices. Nowadays, almost all documents can be exported to PDF. Notably, with a virtual PDF printer, any document that can be printed on paper can be converted to PDF. It has become the *de facto* standard for academic articles, technical reports, manuals, newspapers and ebooks. As an example, the final format for *TUGboat* is PDF.

PDF is a print-ready format; it is designed to completely describe a fixed-layout flat document. A PDF file clearly defines the appearance of the document, independent of particular devices or viewers.

PDF is not supported in the HTML standard, but it can be viewed directly in several web browsers. Users of other web browsers usually have to read PDF documents with web browser plugins, or download the files and then read them with a local PDF reader. In all these cases, PDF files are viewed in a closed environment where users cannot utilize most web features.<sup>1</sup>

**T<sub>E</sub>X** Designed and written by Professor Donald Knuth, **T<sub>E</sub>X** is one of the most powerful typesetting systems in the world.<sup>2</sup> It is well-known for its capability of producing high quality formulas and figures

<sup>1</sup> PDF does include features such as external links and interactive functions within a document, but these are quite limited compared to HTML.

<sup>2</sup> When using ‘**T<sub>E</sub>X**’ in this article, most of the time we will be referring to the whole **T<sub>E</sub>X** family.

in many different areas. While it is most popular in academia, it is also used for typesetting books, magazines and sheet music.

**T**EX is a source format for *authors*. It contains structured contents including text, formulas, figures and possibly cross references between them. Users can define their own concepts by writing macros. Typically the layout of the document must be determined by *compiling* the file with a TEX compiler; different compilers may produce different results from the same TEX source file.

People started trying to connect TEX to the Web nearly since the Web began. There were some early overviews, such as [31, 32], [34, chapter 7], but we are not aware of any recent surveys on the topic. Early works were mainly focusing on correctly displaying formulas produced by TEX. Different methods include using images, Unicode characters, MathML or HTML5. However the power of TEX is far more than formulas, it is also famous for its capability of handling mathematical spacing, hyphenation and justification, which is often ignored in these cases.

In the following sections, we are going to describe and compare some popular existing approaches. We will also introduce a new program, pdf2htmlEX [25], and discuss its advantages and limitations with examples.

## 2 Preliminaries

The target audience of this article includes those who need to publish both online versions and print versions of their documents at the same time, especially those who want to publish existing documents online.

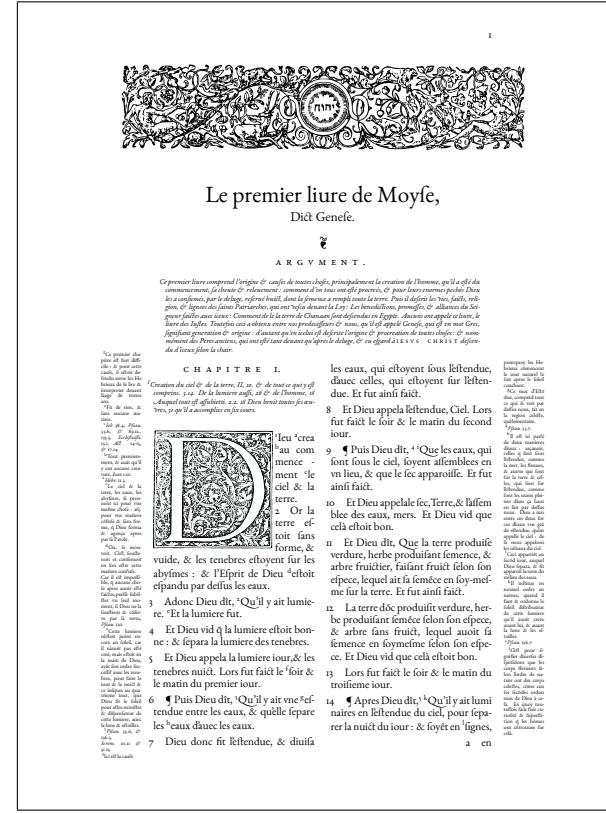
We assume that the existing document is in PDF format. It could be generated from TEX or any other tool. We do not assume that the publisher is the author, *i.e.* the source files may not be available to the publisher.

We believe that the following requirements are essential for most users. They are also the criteria we will use to discuss and compare existing approaches.

**Convenience** The publishing process should be automated, with minimal manual adjustments involved, such that publishers need focus on only one version, while the other can be generated accordingly.

**Consistency** Both the online version and the print version should have a consistent appearance, sometimes including the same layout and format.

Evidently the contents should never vary between the two versions, but one may argue that screen and paper are two completely different kinds of media, and so fonts, spacing and even layouts should be optimized individually. For example, users



**Figure 1:** Bible de Genève, 1564 [8], typeset by Raphaël Pinson with XeTTeX. The drop cap, fonts, spacing and layout were carefully tuned in order to duplicate the 16th century French Bible.

might want text in the document to be reflowed according to the screen size of their mobile devices.

However, in many cases, fonts, spacing and layouts are carefully designed to assist reading, and sometimes they have already become essential parts of the document, see Figures 1 and 2 as examples. In such cases, a complete redesign may be involved in order to optimize for specific media.

In our opinion both situations are important, and we will try to cover both of them in this article.

**Flexibility** An important purpose of the online version is to provide better services and user experience. This version of the document should be flexible enough for front-end designers to design interactive web pages.

For example, text and other elements in the documents should be accessible such that extra styles or effects can be specified; the whole document should be able to be embedded into existing frameworks with well-defined behaviours and themes applied.

**Optimization** There are concerns for web services which may not be covered by traditional media: the



**Figure 2:** One page from a product catalogue generated with LATEX. Text paragraphs, images and tables are well-organized for each category. The whole catalogue contains more than 70 pages, including information on 800–1000 products. Courtesy of Jason Lewis [35].

size of the files should be as small as possible in order to save storage space; the cache mechanism of web browsers should be utilized when possible, in order to save bandwidth; the readers should not need to wait long before viewing the first few pages, even if there are thousands of pages in the document.

Therefore special optimizations are necessary when producing an online version from a traditional document.

### 3 Existing approaches

Quite a number of approaches have been developed to publish TeX or PDF contents online. Possible workflows are shown in Figure 3. It is possible to *compile*<sup>3</sup> a TeX file into HTML; or to *convert*<sup>4</sup> a PDF document into HTML.

<sup>3</sup> To determine the layout based on the information from the source.

<sup>4</sup> To transform between two presentation formats, in both of which layout and appearance are clearly defined.

Converting a large TeX file with complicated layouts into PDF is usually not a fast process. Because of this, it is a common practice to convert the source format into web pages on the server side, the results can be stored on the servers, and sent to users upon request.

On the other hand, nowadays JavaScript is already powerful enough for many tasks, and it can be *embedded*<sup>5</sup> into HTML, in which case HTML is used as a container—the embedded files are to be parsed and rendered on the client side with JavaScript.

In order to utilize existing technologies, it is also common to introduce intermediate formats, which are to be converted or embedded into HTML. In particular, PDF may be viewed as an intermediate format while compiling TeX to HTML,

In this section we try to describe the most popular approaches and discuss them from different aspects. Although some of them might not be originally designed for publishing, they are still listed here because they can be used to facilitate the process.

#### 3.1 Raster image-based approaches

Approaches of this type render source files into raster images (*e.g.* PNG, JPEG), usually one image per page, which are then embedded into HTML. Popular tools of this type include:

- pdftocairo from Poppler [27]
- ImageMagick [20]
- mathTeX [9]
- “fallback” mode of pdf2htmlEX

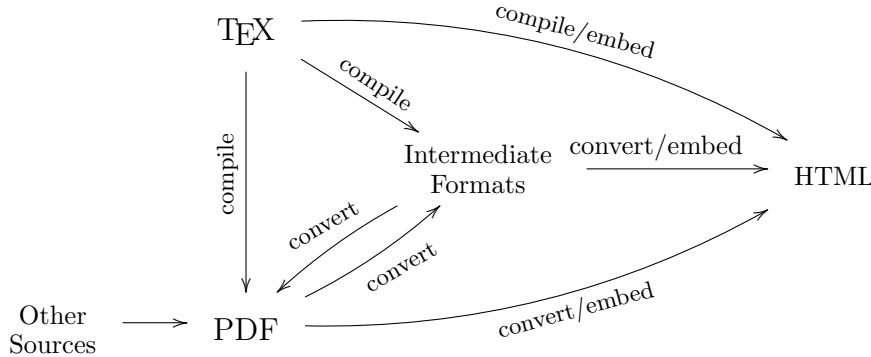
**Pros** Raster images were introduced in a very early stage of HTML, and so are highly compatible with old web browsers. All visual elements can be displayed correctly.<sup>6</sup>

**Cons** The main disadvantage of this type of approach is that the image sizes are usually huge. It is costly to convert text into images and it is usually not easy to balance quality and size. Large files consume large bandwidth of both server and client, which also cause delays. Another issue is that all semantic information is lost, users can no longer copy text out from the document, nor follow the links.

Raster image-based approaches are “universal”, in that they are widely used to publish many different formats, not limited to TeX or PDF. Famous examples include the *Look Inside* feature of SpringerLink [11] and Google Docs Viewer [3].

<sup>5</sup> To keep the source format as it is inside the target format.

<sup>6</sup> For TeX and PDF, there are also advanced features like audio, video, animation or annotation, *etc.*, which are beyond the scope of this article.



**Figure 3:** Different approaches to publishing online.

The disadvantages can be compensated for to some extent: A hidden text layer can be overlaid above the images in order to simulate user text selection, however generating this text layer itself actually involves other conversion technologies; for a responsive user experience, the input document may be converted into images with different resolutions, and images with high resolutions can then be split into small blocks. When the document is rendered on the client side, only the block being viewed by the user is needed to transfer. However much more disk storage and network bandwidth is required in this way, which might not be affordable for all publishers, especially individuals.

### 3.2 SVG-based approaches

Scalable Vector Graphics, developed by W3C, is an XML-based format for presenting 2D graphics. It supports a similar set of features as PDF, including color, gradients, patterns, paintings and raster images. It also supports font definition within SVG as well as external fonts defined in CSS.

Due to the large feature set, most visual elements can be rendered with SVG counterparts. Popular tools in this category include:

- Inkscape [21]
- pdftocairo from Poppler [27]
- pdf2svg [26]
- dvisvgm [17]
- “fallback” mode of pdf2htmlEX

**Pros** Due to the similar nature between SVG and PDF, it is relatively easy to find an SVG counterpart for each PDF element. SVG is one of the few methods that support advanced layout features such as characters along a curved path and image clipping.

SVG is based on XML, hence it can be easily parsed or edited manually. SVG can be well inte-

grated with HTML/CSS, and it can be easily accessed and manipulated by JavaScript.

**Cons** Old web browsers do not support SVG, and the degree of support for SVG varies for modern web browsers.

While SVG-based approaches are powerful when integrated with HTML, CSS and JavaScript, most tools in this category do not support such integrations, probably because they were designed as an SVG converter instead of an online publishing tool.

### 3.3 Semantic HTML-based approaches

Approaches of this type try to find the matching HTML element for each `\TeX` element, for example `\section` and `\textbf` in `\TeX` might become `<h1>` and `<b>` in HTML respectively. Popular tools in this category include

- HEVEA [19]
- `\LaTeX2HTML` [1, 37]
- `\LaTeXML` [24]
- `plasTeX` [5, 29]
- `\TeX4ht` [6, 33]
- `\TeX2page` [14]
- `TtH` [12]

all of which are designed to process general `\TeX` files. There are also programs designed for particular documents, for example

- The Feynman Lectures on Physics [28]
- The Stacks Project [38]
- The TUG Interviews Project [30]

**Pros** Semantic HTML files are normally expected by most users. Semantic information is retained in an XML-like format, such that they can be read or edited by a human or further processed by other programs.

Basic elements such as colors, font family and sizes, paragraphs, links, images can all be supported.

CSS can be used to specify the layout and appearance. Math formulas may be semantically retained via Unicode characters, MathML or embedded  $\text{\TeX}$  snippets (see Section 3.5).

Approaches of this type can be used when the publisher does not rely on the layout produced by a  $\text{\TeX}$  compiler. They are often used for simple text-based files without complicated layouts. The final layout is determined by the web browser based on the semantic structure and CSS rules.

**Cons** Approaches of this type usually don't work well for PDF files. In general, PDF files do not contain semantic information, and so *recognition* is inevitable to detect semantic meanings, which is considered to be hard. This is also true for other intermediate files.

On the other hand,  $\text{\TeX}$  users do not necessarily expect the same appearance as compiled by  $\text{\TeX}$ . Most advanced layouts in  $\text{\TeX}$  cannot be used, for example, double columns. Specific layouts might be simulated, but it is hard in general due to the essential differences between the page model of  $\text{\TeX}$  and HTML. While font embedding is possible nowadays, most tools of this kind do not support it.

Furthermore, this type of approach can be considered a re-implementation of  $\text{\TeX}$ , as these tools parse and process  $\text{\TeX}$  syntax in their own engines, and therefore some macros and packages may not work with them, especially those related to drawing, page layout or PDF-specific features. Sometimes the authors have to prepare different versions of  $\text{\TeX}$  files for both HTML and PDF, and HTML knowledge might also be required.

It is possible to achieve HTML documents in rather good quality, while reserving not only semantic information, but also well organized links, elegant styles and MathJaX-based math formulas. Good examples are [28] and [38]. However, most of them employ project-specific tools and lots of engineering work, and there are also limitations or paradigms for authors. Therefore their methods might not work for general documents.

### 3.4 Presentation HTML-based approaches

Approaches of this type focus on the layout and appearance of the result, utilizing CSS rules to set accurate position and size for each element, mostly text. Non-text elements are usually converted into images, raster or vector, which are embedded in the HTML. They are still significantly different from raster image based approaches or SVG based approaches mentioned above, as they do not convert the whole document into images.

Prior to pdf2htmlEX, the pdftohtml utility from Poppler [27] is probably the best known tool that is freely available to the community. While pdftohtml focuses more on extracting semantic information, pdf2htmlEX focuses on precise layout and appearance. There seems not to be any tools that directly produce presentation HTML files from  $\text{\TeX}$ , because of course  $\text{\TeX}$  users may produce PDF files before further converting it into HTML.

**Pros** Comparing this output with images, text is now represented with native HTML elements, such that they can be selected by users or easily extracted by programs; the file size is heavily reduced in this way. Also it is easier to apply CSS and JavaScript to tweak the appearance.

Comparing with semantic HTML files, the appearance of presentation HTML output is often closer or even identical to the original document.  $\text{\TeX}$  users are free to use any advanced layout, macro or package, fine-tuning will also be reflected in the output, as the  $\text{\TeX}$  page model is simulated in HTML.

**Cons** While text is still available, the semantic meanings (e.g. title, section, table etc.) are likely to be lost. The content may be too complicated to be further processed. Precise layout and appearance rely on advanced CSS features, like font embedding, absolute positioning and linear transformation, which might not be supported by old web browsers.

### 3.5 JavaScript-based approaches for $\text{\TeX}$

While  $\text{\TeX}$  is not directly supported in HTML, modern JavaScript technologies allow us to *embed* these files in HTML, such that they will be parsed and rendered directly in the web browsers. Similar technologies for PDF are covered in Section 3.6.

MathJaX [7] is a JavaScript display engine which parses and renders  $\text{\TeX}$  snippets on web pages with HTML/CSS, SVG or MathML. MathJaX is designed for online communications where users want to directly input formulas in the  $\text{\TeX}$  syntax. Similar projects include jsTeX [2], and jsMath [4].

I<sup>A</sup>T<sub>E</sub>X2HTML5 [22] is able to produce interactive diagrams from PSTricks macros; it also utilizes MathJaX for rendering math formulas.

**Pros** This kind of approach is best for dynamic content, especially that intended to be created or modified by users. Any modification to the source can be reflected in the result promptly without any network transmission. It can also handle documents with simple layout, while formatting can be specified with CSS.

**Cons** Approaches of this kind are usually focusing on specific elements; while they may support a

small set of  $\text{\TeX}$  syntax, they are not designed as a JavaScript implementation of  $\text{\TeX}$ . Therefore advanced commands or macros may not be supported, and usually they are not capable of typesetting general documents with complicated layout.

### 3.6 JavaScript-based approaches for PDF

PDF.js [13] is a JavaScript library for rendering PDF; it is now a part of Mozilla Firefox. It is like the raster image-based approaches except that all the parsing and rendering are done on the client side. Recent web browsers are necessary to support the technologies used by the library.

PDF.js is one of a kind; there are no similar alternatives to the best of our knowledge.

**Pros** PDF.js renders PDF files into an HTML5 canvas, which is similar to a raster image; most PDF elements can be rendered correctly. Furthermore, it does not suffer from a huge network cost as only the original PDF file need be transferred.

The library can be embedded into web pages, and can be extended by publishers if needed.

**Cons** PDF.js relies heavily on the computation power on the client side, which might cause performance issues in some environments.

It is designed as a PDF reader, and it does not optimize for online publishing; for example users still have to wait for the entire file to be downloaded before they can read any page.

PDF elements are rendered into an HTML5 canvas, which may not be flexible enough for publishers.

### 3.7 Plugin-based approaches

Many web browsers support plugins to add new features, especially plugins can be used to display  $\text{\TeX}$ , PDF, or other formats converted from them. Publishers may also develop plugins for their own formats, which are otherwise not supported by web browsers.

Adobe Reader includes plugins to display PDF files within different web browsers. There are also similar third-party plugins based on Adobe Flash. There are also plugins to display math formulas inside web browsers, *e.g.* MathPlayer [23].

**Pros** Plugins are not limited to web technologies, thus they are usually better in term of rendering quality or supported features. For example, Adobe Reader should be the plugin with the most complete support for PDF features.

**Cons** The crucial downside is that plugins usually create closed environments, which prevent an interactive user experience on the web sites. Most plugins are not easily customizable, except for a few commercial ones. Due to the active development of

HTML5 technologies, plugins nowadays are no longer so popular as before, for security, compatibility and performance reasons.

### 3.8 Third-party services

Third-party services also exist such that embedded  $\text{\TeX}$  or PDF can be redirected to their servers for processing and rendering, for example:

- Quick $\text{\LaTeX}$  [10]
- math $\text{\TeX}$  [9]
- Google Docs [3]
- Crocodoc [15]

This kind of service accepts input uploaded by publishers, converts it internally with their own implementations and redirects the result to users. The technologies behind them, open or proprietary, should still fall into the categories mentioned above.

**Pros** This kind of service is usually easy to deploy, and convenient for publishing a few simple documents. The conversion process relies only on the computation power of the third party. Some services also provide an API for better integration with publishers' sites.

**Cons** A crucial issue here that may concern a lot of publishers is that the files must be accessible by the third party. This may not be acceptable for private, confidential or copyrighted materials. In addition, the publisher's service has to depend on the availability of the third party. Further development may also be limited by the API provided or other issues such as different domains.

### 3.9 Discussion

In this section we categorized a number of popular approaches, among which the most popular three types of output are:

**Image** is best for publishers who can afford large volume of storage and bandwidth. There is no need to fine-tune or even redesign the document, as layout and format are already accurately preserved in the output. Yet this highly compatible result can be viewed by most users.

**Semantic HTML** is best for simple  $\text{\TeX}$  source files. All semantic information is preserved in the output, which can be further processed by other tools. In particular, math formulas may be rendered interactively with latest web technologies.

**Presentation HTML** is best when complicated layout, advanced  $\text{\TeX}$  macros and packages are used. It is also suitable when the source files are not available at all. PDF files produced from other tools can

also be supported. Flexibility of semantic HTML and accuracy of image are cleverly balanced.

In general there is no best approach for all situations. Users should carefully choose the best matching one according their specific concerns.

While we tried our best to be complete and accurate, it is quite possible that we have missed or misunderstood some approaches due to the limitations of our knowledge. Please contact us for corrections or suggestions, and thank you.

#### 4 A tour of pdf2htmlEX

 In this section we introduce pdf2htmlEX, created and mostly written by Lu Wang, which is an open source PDF to HTML converter. It generates presentation HTML documents, utilizing modern Web technologies such as HTML5, CSS3, JavaScript, etc., such that most PDF features can be retained. Especially fonts, math formulas and images can all be displayed correctly.



Figure 4: Logo of pdf2htmlEX

pdf2htmlEX is not only a *converter*, but also a *publishing tool*. It is designed for many different situations, for example:

**Scenario 1** My sister wants to put her resumé on her online homepage. She wants the resumé to be stored into one single file such that it can be easily downloaded by others. She also needs to add JavaScript code to track how many people have read her resumé.

**Scenario 2** A book publisher wants to put some sample books online to attract readers. The publisher does not want readers to wait for too long before they can read any page, thus pages are better converted and stored separately. Images and fonts should also be stored in individual files such that users may benefit from web caches.

**Scenario 3** A cloud storage service provider wants to provide a PDF preview feature to their service, such that users may read their files online. The service provider needs to design their own viewer to match the theme and behaviour of their web site. They also want to attach advertisements based on the contents of the files. Advanced users may be allowed to leave marks and discuss with others about particular parts of the documents. In this case the

service provider needs the finest control—they need to access every single element of the document for their customizations.

We can see that different forms of HTML files are desired in different scenarios, and flexibility is always necessary. pdf2htmlEX is indeed designed for all these scenarios and many others; some features have been requested or implemented by users.

In this section we will introduce a few useful features and explore some internal mechanisms of pdf2htmlEX. More information, including source code and license terms, is at the project home page: <https://github.com/coolwanglu/pdf2htmlEX>

##### 4.1 Quick start

Throughout this section, a sample PDF file is used to demonstrate different features of pdf2htmlEX. The file, `integral.pdf`, contains 4 pages from the book *Differential and Integral II* [16, 36], which consists of Japanese characters, mathematical symbols and formulas, figures, images and delicate layouts. We believe that it reflects common elements used in real use cases.

To start with, we simply execute

```
$ pdf2htmlEX --fit-width 1024 integral.pdf
```

which produces a *single* HTML file `integral.html`. The result is shown in Figure 5,<sup>7</sup> and we challenge the readers to find any evidence or clues that the screenshot shows an HTML file instead of PDF. (Except for the title bar of course.)

The `--fit-width 1024` option specifies that each page should be squeezed or stretched to the width of 1024 pixels. The zoom ratio can be adjusted with similar options: `--fit-height` and `--zoom`.

It is possible to convert only a few pages of a PDF file, for example

```
$ pdf2htmlEX -f 2 -l 3 integral.pdf
```

converts only the second page and the third page.

##### 4.2 Separating resource files

By default everything is combined into one single HTML file, which is good for creating archives or performing tests. However, it is not a good practice when publishing HTML documents online; often we want resource files (fonts, CSS, JavaScript, images etc.) to be stored separately in order to reduce size and improve efficiency.

With the `--embed` option, we can decide which types of resource files are embedded and which are not. For example,

```
$ pdf2htmlEX --embed fi integral.pdf
```

<sup>7</sup> Mozilla Firefox 24 on Ubuntu 13.04 is used for all the demonstrations.

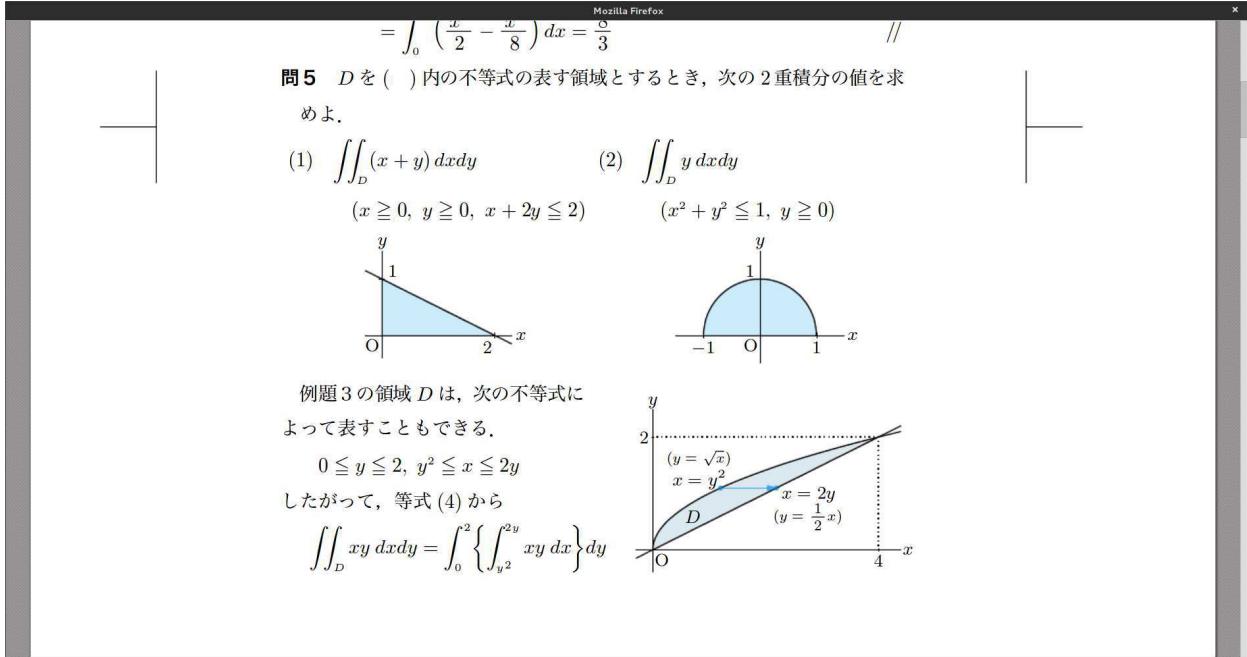


Figure 5: An HTML document produced by pdf2htmlEX.

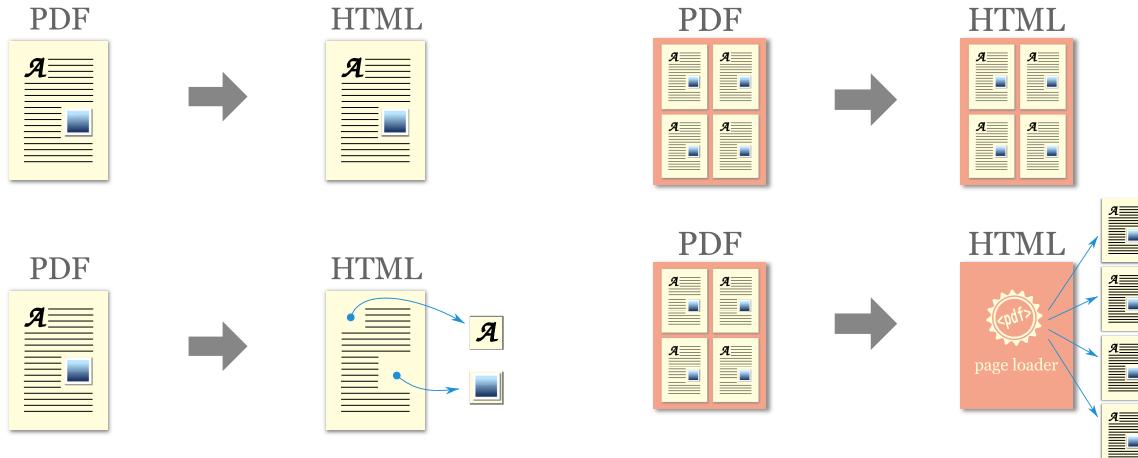


Figure 6: Above: by default all resources are embedded in the HTML file. Below: with the `--embed-fi` option, fonts and images are stored into separate files and linked to the HTML file.

stores all fonts and images in separate files, as shown in Figure 6. There are also specific options including `--embed-css`, `--embed-font`, `--embed-image`, etc.

### 4.3 Splitting pages

With a large PDF file containing hundreds of pages, often we have to download the whole file even if we want to take a look at only a few pages inside. On the other hand, web pages are usually stored into separate files, such that we just need to download the pages we request.

Figure 7: Above: by default all pages are embedded in the HTML file. Below: with `--split-pages 1`, pages are stored in separate HTML snippets, which can be dynamically loaded to the main HTML file.

With the `--split-pages` option, it is possible to store PDF pages into separate HTML snippets. In this way, when the main HTML file is loaded on the client side, only necessary pages will be dynamically loaded via Ajax, as shown in Figure 7.

### 4.4 Image format for backgrounds

For each page, pdf2htmlEX generates a background image to present all non-text elements. By default

all images are generated in the PNG format, and different formats can be specified via the `--bg-format` option. For example,

```
$ pdf2htmlEX --bg-format jpg integral.pdf
```

would generate all images in the JPEG format.

Currently pdf2htmlEX supports PNG and JPEG. There is also preliminary support for SVG. Users can also convert the images into other formats.

#### 4.5 Customizing the output

`integral.html` contains a default set of HTML, CSS and JavaScript, which is designed for average use cases. All of them can be found in the so-called `data-dir` (run `pdf2htmlEX -v` to see the location), and they can be tweaked by the users.

**HTML template** The `manifest` file determines how pages should be combined into an HTML document. It is a template for the output and users may add their own HTML snippets into it. A typical use case is enabling a traffic statistics service on the page.

**CSS** Quite a number of features of pdf2htmlEX rely on CSS; the default CSS styles determine the correct appearance and behavior of the elements. Advanced users can override existing properties by modifying the CSS files.

**JavaScript** A simple UI is implemented in the default `pdf2htmlEX.js` file. This also serves as a demonstration of accessing and manipulating HTML elements produced by pdf2htmlEX. It can be a good reference for advanced users who want to implement their own UIs.

#### 4.6 Secrets of pdf2htmlEX

Here we briefly introduce some internal mechanisms of pdf2htmlEX for the curious readers.

`integral.html` consists of two layers: the text layer and the image layer, as shown in Figures 8 and 9. pdf2htmlEX parses `internal.pdf` and extracts elements from it. The elements are then processed and put into one of the layers.

**Text** Unlike in HTML, in PDF text is set in fixed positions. Text extracted from the PDF is translated into native HTML text elements, and put into the same position in the HTML as they were in PDF. In this way text can be selected and copied by users, while preserving the layout. Many fixed-position text elements in HTML make the file very large in size and very slow to render; to compensate, pdf2htmlEX tries to recognize and merge text lines according to their geometric metrics.

**Font** Font embedding is one of the most important features of PDF, without which it is nearly impos-

sible to preserve the appearance of PDF in HTML. No similar feature has been supported in the HTML standard until recently. Figure 10 shows a few fonts used in `integral.pdf`. pdf2htmlEX is able to extract all the fonts from PDF and convert them into web fonts via FontForge [18]; converted fonts are then embedded or referred to in the HTML file. All font formats supported in PDF are supported by pdf2htmlEX, and different web font formats can be specified for output.

**Encoding** Unlike HTML, PDF uses two sets of encodings for text rendering, one for choosing correct glyphs to display, and the other for meaningful text that can be selected and copied by users. pdf2htmlEX is able to combine both sets into one and re-encode the font accordingly, such that text in HTML is correct both visually and meaningfully. This is another essential feature of pdf2htmlEX, like font processing.

**Images** PDF supports graphical instructions such as drawing and image embedding. Such elements are all rendered into images, and then put into the image layer.

#### 4.7 Future work

Several features are planned in the future versions of pdf2htmlEX.

**Reflowable text** Comparing with HTML files directly converted from TeX, text in HTML files generated by pdf2htmlEX is generally not reflowable, *i.e.* the width of paragraphs cannot be self-adapting to the size of the viewer. After all, that information is generally not available in a PDF file, and it is not easy for pdf2htmlEX to recover it.

On the other hand, reflowable text may be extracted for specific document types and layouts. Extracting such information would make it much easier to further process HTML files generated by pdf2htmlEX, such as to edit manually, to embed accessibility information or to convert into other formats like EPUB.

**Preserving semantic information** While much semantic information is lost in PDF as mentioned above, theoretically it is possible for authors to embed additional information into PDF, such that it may be further recognized and used by pdf2htmlEX. This kind of PDF file is called a *tagged* PDF, which can also be generated with other tools.

Especially for TeX users, it is possible to mark text paragraphs such that text will be reflowable in HTML to some extent; also, mathematical formulas may be marked such that they will be rendered with MathJaX in HTML.

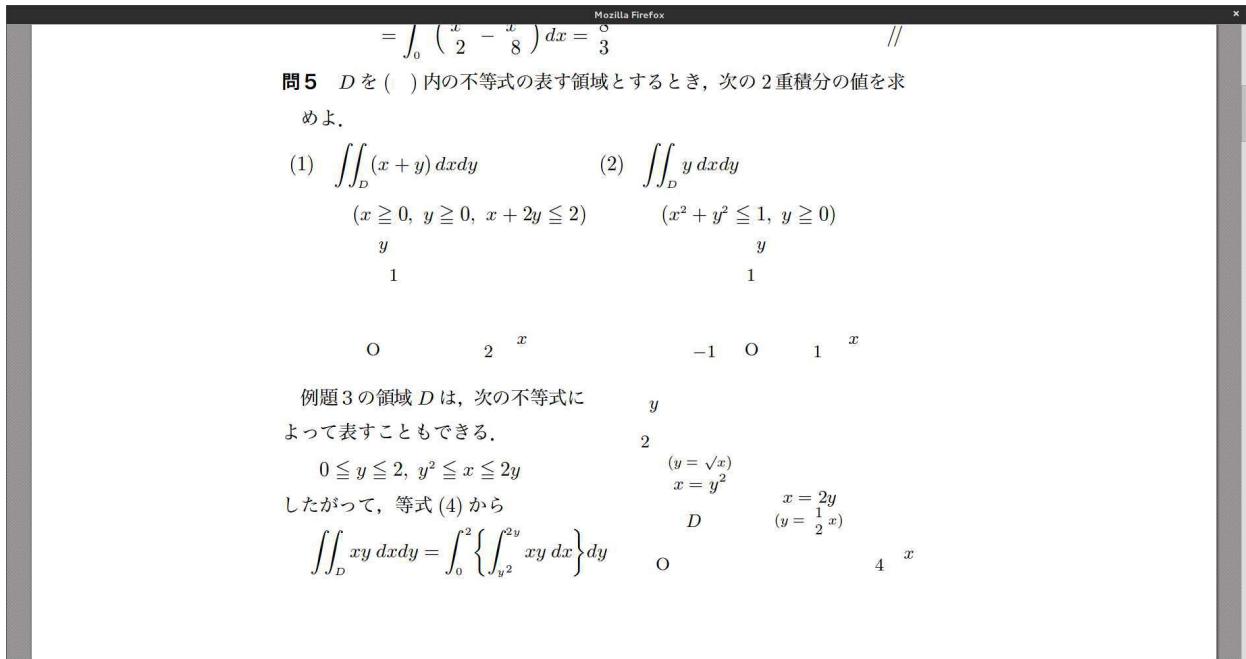


Figure 8: The text layer

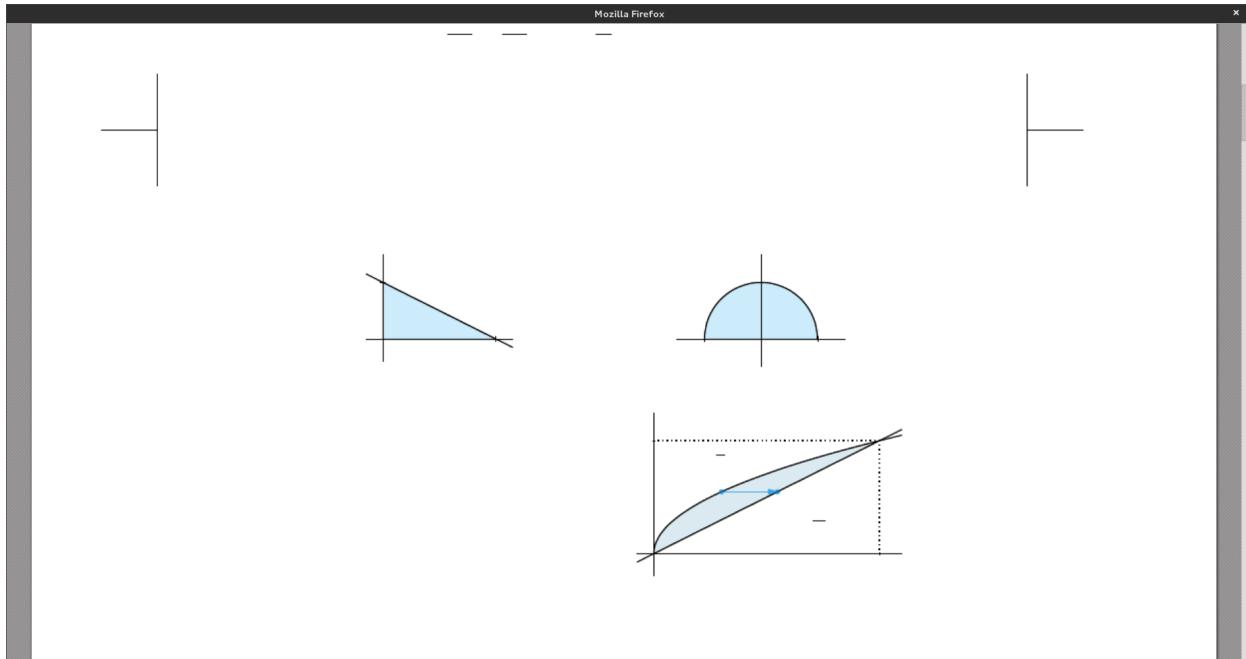


Figure 9: The image layer

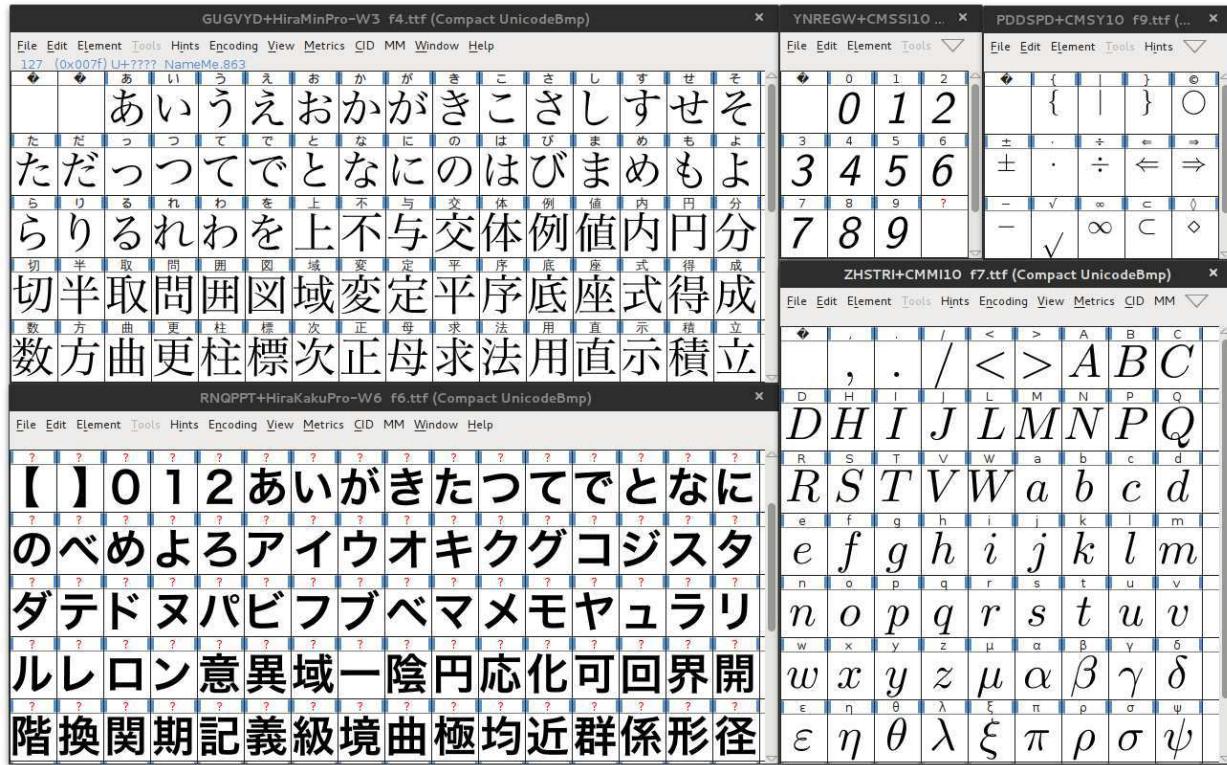


Figure 10: Fonts embedded in the HTML file

**Image overlay** For each PDF page, pdf2htmlEX puts all non-text elements into a background image; this image is then put behind all the text in that page. However, it is possible that some text is in fact covered by an image in the PDF, in which case in the corresponding HTML file produced by pdf2htmlEX, the text will be visible due to the superimposition.

We are still looking for efficient solutions for this issue; fortunately, this issue is not common, especially for TeX users. A workaround is to use the *fallback* mode of pdf2htmlEX at the cost of larger file size.

**Image optimization** When generating the background image, pdf2htmlEX calculates the bounding box of all non-text elements in that page, and renders everything inside. However, if there are only a few images which are far away from each other, most parts in the image are actually blank, which will be a waste of bandwidth. It is possible to recognize and split those small images and pack them into one small image, such that they will be loaded using the CSS sprite technique. In this way significant bandwidth and computation can be saved.

#### 4.8 Discussion

In this section we introduced pdf2htmlEX from several perspectives. Due to space limitations here, we cannot present everything—there are nearly 50 different options in total, and there are also tricky implementations regarding font conversion, text handling and image processing. Interested readers are encouraged to visit the project web site for detailed and up-to-date documentation.

#### 5 Conclusion

In this article we tried to categorize and compare existing methods of publishing TeX or PDF online. We hope that readers may use this article as a guide to choose the proper tool for their specific use cases, or be inspired to create their own implementations.

We also introduced our program pdf2htmlEX, a PDF to HTML converter and publishing tool which is accurate and flexible for many different use cases. We encourage interested users to get involved.

#### Acknowledgement

We thank Professor Haruhiko Okumura for his help and great advice. We also thank Professor Makasata Kaneko, Mr Raphaël Pinson and Mr Jason Lewis for the nice sample files used in this article.

## References

- [1] L<sup>A</sup>T<sub>E</sub>X2HTML. <http://www.latex2html.org>, 2001.
- [2] jsTeX. <http://simile.mit.edu/wiki/JsTeX>, 2008.
- [3] Google Docs Viewer. <https://docs.google.com/viewer>, 2009.
- [4] jsMath: A Method of Including Mathematics in Web Pages. <http://www.math.union.edu/~dpvc/jsmath>, 2009.
- [5] plasTeX. <http://plastex.sourceforge.net>, 2009.
- [6] TeX4ht: L<sup>A</sup>T<sub>E</sub>X and TeX for Hypertext. <http://tug.org/tex4ht>, 2010.
- [7] MathJax: Beautiful math in all browsers. <http://www.mathjax.org>, 2011.
- [8] Bible de Genève, 1564. [https://github.com/raphink/geneve\\_1564](https://github.com/raphink/geneve_1564), 2012.
- [9] mathTeX. <http://www.forkosh.com/mathtex.html>, 2012.
- [10] QuickL<sup>A</sup>T<sub>E</sub>X — advanced L<sup>A</sup>T<sub>E</sub>X web rendering service. <http://quicklatex.com>, 2012.
- [11] SpringerLink. <http://link.springer.com/>, 2012.
- [12] TtH: The TeX to HTML translator. <http://hutchinson.belmont.ma.us/tth>, 2012.
- [13] PDF.js. <https://github.com/mozilla/pdf.js>, 2013.
- [14] TeX2page. <http://www.ccs.neu.edu/home/dorai/tex2page/index.html>, 2013.
- [15] Crocodoc: HTML5 Document Embedding. <https://crocodoc.com>, 2013.
- [16] *Differential and Integral II*. Dai-Nippon Toshō Publisher, 2013.
- [17] dvisvgm: A DVI to SVG converter. <http://dvisvgm.sourceforge.net>, 2013.
- [18] FontForge: A font editor. <http://fontforge.org>, 2013.
- [19] HEVEA: A L<sup>A</sup>T<sub>E</sub>X to HTML translator. <http://hevea.inria.fr>, 2013.
- [20] ImageMagick: Convert, Edit, And Compose Images. <http://www.imagemagick.org>, 2013.
- [21] Inkscape: An open source scalable vector graphics editor. <http://inkscape.org>, 2013.
- [22] L<sup>A</sup>T<sub>E</sub>X2HTML5—interactive math equations and diagrams. <http://latex2html5.com>, 2013.
- [23] MathPlayer: Display MathML in your browser. <http://www.dessci.com/en/products/mathplayer>, 2013.
- [24] L<sup>A</sup>T<sub>E</sub>XML: A L<sup>A</sup>T<sub>E</sub>X to XML Converter. <http://dlmf.nist.gov/LaTeXML>, 2013.
- [25] pdf2htmlEX: Convert PDF to HTML without losing text or format. <https://github.com/coolwanglu/pdf2htmlex>, 2013.
- [26] pdf2svg. <http://www.cityinthesky.co.uk.opensource/pdf2svg>, 2013.
- [27] Poppler. <http://poppler.freedesktop.org>, 2013.
- [28] The Feynman Lectures on Physics. <http://www.feynmanlectures.info>, 2013.
- [29] Tim Arnold. Getting started with plasTeX. *TUGboat*, 30(2):180–182, 2009. <http://tug.org/TUGboat/tb30-2/tb95arnold.pdf>.
- [30] Karl Berry and David Walden. TeX People: The TUG interviews project and book. *TUGboat*, 30(2):196–202, 2009. <http://tug.org/TUGboat/tb30-2/tb95berry-interviews.pdf>.
- [31] Peter Flynn. L<sup>A</sup>T<sub>E</sub>X on the Web. *TUGboat*, 26(1):66–67, 2005. <http://tug.org/TUGboat/tb26-1/flynn.pdf>.
- [32] Stephen A. Fulling. Keynote: TeX and the Web in the higher education of the future: Dreams and difficulties. *TUGboat*, 20(3):371–372, 1999. <http://tug.org/TUGboat/tb11-3/tb29fulling.pdf>.
- [33] Eitan Gurari. TeX4ht: HTML production. *TUGboat*, 25(1):39–47, 2004. <http://tug.org/TUGboat/tb25-1/gurari.pdf>.
- [34] Steven G. Krantz. *Handbook of Typography for Mathematical Sciences*. Chapman and Hall/CRC, 2000.
- [35] Jason Lewis. How I use L<sup>A</sup>T<sub>E</sub>X to make a product catalogue that doesn't look like a dissertation. *TUGboat*, 34(3):263–267, 2013.
- [36] Yoshifumi Maeda and Masataka Kaneko. Making math textbooks and materials with TeX+KETpic+hyperlink. Presentation at TUG 2013.
- [37] Ross Moore. Presenting mathematics and languages in Web-pages using L<sup>A</sup>T<sub>E</sub>X2HTML. *TUGboat*, 19(2):195–203, 1998. <http://tug.org/TUGboat/tb19-2/tb59moore.pdf>.
- [38] The Stacks Project Authors. The Stacks Project. <http://stacks.math.columbia.edu>, 2013.

◇ Lu Wang  
 Department of Computer Science  
 and Engineering  
 The Hong Kong University of  
 Science and Technology  
 Hong Kong  
 coolwanglu (at) gmail dot com  
<http://coolwanglu.github.io/>

◇ Wanmin Liu  
 Department of Mathematics  
 The Hong Kong University of  
 Science and Technology  
 Hong Kong  
 wanminliu (at) gmail dot com

## The X<sup>M</sup>TeX system for publishing interdisciplinary chemistry/mathematics books

Shinsaku Fujita

### 1 X<sup>M</sup>TeX Version 5.01

I have recently released X<sup>M</sup>TeX Version 5.01 for drawing chemical structural formulas, where its zip file (`xymtx501.zip`) is available from my personal homepage (<http://xymtex.com/>). I have more recently uploaded this version to the CTAN archives.

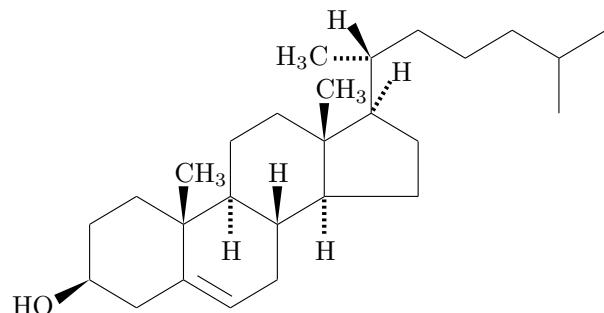
The X<sup>M</sup>TeX system supports three modes for drawing:

1. the L<sup>A</sup>T<sub>E</sub>X-compatible mode, which is based on the L<sup>A</sup>T<sub>E</sub>X picture environment along with the `epic` package,
2. the PostScript-compatible mode, which is based on the `PSTricks` package, and
3. the PDF-compatible mode, which is based on the `pgf/TikZ` package.

The three modes can be switched by loading the `xymtex`, `xymtexps`, or `xymtexpdf` package with the `\usepackage` command. If structural formulas of high quality are necessary, the latter two modes should be selected. A typical template for switching the three modes is shown below:

```
\documentclass{article}
%\usepackage{xymtex} %LaTeX mode
%\usepackage{xymtexps}%PostScript mode
\usepackage{xymtexpdf}%PDF mode
\usepackage{graphicx}
\begin{document}
\cholestane[e]{3B==HO}%XyMTeX command
\end{document}
```

The X<sup>M</sup>TeX command `\cholestane` with the arguments `[e]` and `{3B==HO}` generates the chemical structural formula of cholest-5-en-3 $\beta$ -ol as follows:



Because PDF is now a default standard for exchanging digital documents, it is usually highly desirable to convert DVI files obtained by the PostScript-compatible mode or the PDF-compatible mode to

PDF files. To obtain a PDF file of printing quality, the following routes are typical:

1. PostScript-compatible mode:

As the more classical process, a `dvi` file produced by the PostScript-compatible mode is converted into a `ps` file. The resulting `ps` file is in turn converted into a `pdf` file.

`tex`  $\xrightarrow{\text{L}\text{\^A}\text{T}\text{\^E}\text{X}}$  `dvi`  $\xrightarrow{\text{dvips}}$  `ps`  $\xrightarrow{\text{distiller}}$  `pdf`

2. PDF-compatible mode:

- Because PDF technology has become predominant over the PostScript technology, a `dvi` file produced by the PDF-compatible mode is directly converted into a `pdf` file by using the `dvipdfmx` converter.

`tex`  $\xrightarrow{\text{L}\text{\^A}\text{T}\text{\^E}\text{X}}$  `dvi`  $\xrightarrow{\text{dvipdfmx}}$  `pdf`

- The PDF-compatible mode can take an optional argument `pdftex` as follows:

`\usepackage[pdftex]{xymtexpdf}`

Thereby, a `tex` file is directly converted into a `pdf` file by using the `pdflatex` engine:

`tex`  $\xrightarrow{\text{pdflatex}}$  `pdf`

It should be emphasized that common code written for the X<sup>M</sup>TeX system can be used in any of the routes itemized above.

### 2 Techniques for drawing complicated structural formulas

X<sup>M</sup>TeX commands are equipped with facilities for drawing complex structures, i.e., the substitution technique for attaching substituents, the addition technique for drawing fused rings, and the replacement technique for drawing spiro rings. The detailed documentation of the X<sup>M</sup>TeX system [1] is available from my homepage located at <http://xymtex.com/>.

#### 2.1 The substitution technique

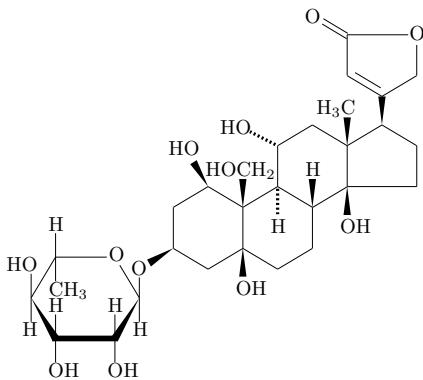
The X<sup>M</sup>TeX system supports the substitution technique, which is based on (yl)-functions for linking complicated substituents. An intervening divalent unit can be inserted by using a command `\ry1` or `\ly1`.

For example, the structural formula of *g*-strophanthin (ouabain) as a poisonous cardiac glycoside is drawn by the code [1]:

```
\begin{XyMcompd}(2000,1850)(-550,-300){}{}
\steroid{1SB==\lmoiety{HO};5B==OH;8B==H;%
9A==H;{11}A==HO;{10}B==\llap{HO}CH$_{2}$;$;%%
{14}B==OH;{13}B==\lmoiety{H$_{2}$O$C$};%%
{17}B==\fiveheterov[e]{3==0}%;%
{4D==0;1==\y1};3B==\y1(3==0){8==%}
```

```
\pyranosew{1==(y1);1Sa==H;2Sb==H;2Sa==OH;%  
3Sb==H;3Sa==OH;4Sb==HO;%  
4Sa==H;5Sb==H;5Sa==CH$_{3}$$)}  
\end{XyMcompd}
```

In this code, a steroid skeleton (due to the \steroid command) is substituted by a five-membered heterocycle (due to a (yl)-function in the \fiveheterov command) and by a pyranose moiety (due to a (yl)-function in the \pyranosew command and a further use of the \yl1 command). The XyMcompd environment secures a drawing area for the structure to be drawn. This code typesets the following structural formula:

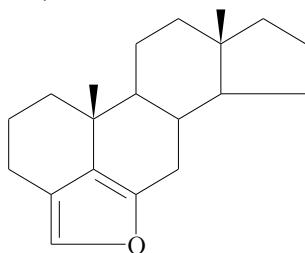


## 2.2 The addition technique

The X<sup>f</sup>M<sub>T</sub>E<sub>X</sub> system supports the addition technique, where the attachment mechanism permits a given fusing unit to be attached onto an edge of a parent skeleton.

For example, furo[40,30,20:4,5,6]androstane as a fused steroid is drawn by the addition technique, where the \fivefusevi command for drawing a 5-membered fusing unit is declared in the bond list of the \steroid command for drawing a steroid skeleton:

```
\steroid  
[{\c{\fivefusevi[ad]{3==0}{\e}[a]}},  
 {\{10\}B==\null,\{13\}B==\null}]
```

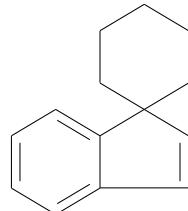


## 2.3 The replacement technique

The X<sup>f</sup>M<sub>T</sub>E<sub>X</sub> system supports the replacement technique, where a spiro unit is drawn on the basis of a (yl)-function and attached to a vertex of a parent skeleton.

For example, spiro[cyclohexane-1,1'-indene] is drawn by the replacement technique, where a six-membered spiro unit is produced by declaring a (yl)-function in the \cyclohexanev command:

```
\begin{XyMcompd}(600,800)(250,250){}  
\nonaheterovi[bdfh]  
\{1s==\cyclohexanev{4==(y1)}}  
\end{XyMcompd}
```



## 3 Interdisciplinary chemistry/mathematics books

The development of the X<sup>f</sup>M<sub>T</sub>E<sub>X</sub> system largely reflects the personal history of my research aiming at the integration of chemistry and mathematics, e.g., the concept of imaginary transition structures (ITSS) [2], the USCI (unit-subduced-cycle-index) approach [3, 4], the concept of stereoisograms [5, 6], the proligand method [7], and the concept of mandalas [8].

### 3.1 Manual drawing without using the X<sup>f</sup>M<sub>T</sub>E<sub>X</sub> system

In 1991, I published an interdisciplinary monograph on the combinatorial enumeration of chemical compounds as three-dimensional structures (the USCI approach) [9]. This book contains many structural formulas of organic compounds along with mathematical equations because of its interdisciplinary nature. Such mathematical equations were successfully typeset by means of the original programs of the (I<sup>A</sup>)T<sub>E</sub>X system. However, the structural formulas contained in this book were drawn manually and pasted on the camera-ready manuscript, because (I<sup>A</sup>)T<sub>E</sub>X at that time had no reliable utility for drawing structural formulas, and commercially available systems such as ChemDraw were too expensive to be used for personal purposes.

The concept of imaginary transition structures (ITSSs), which serve as computer-oriented representations of organic reactions, was developed mainly during the 1980s. In 2001, rather belatedly, I published a monograph on the concept of ITSSs [10]. Although such ITSSs can be regarded as extended structural formulas with colored bonds (par-bonds, out-bonds, and in-bonds), the X<sup>f</sup>M<sub>T</sub>E<sub>X</sub> system at that time did not support utilities of coloring bonds. It follows that the ITSSs contained in this book were drawn manually and pasted on the camera-ready manuscript.

### 3.2 Drawing by the X<sup>M</sup>TEX System

The X<sup>M</sup>TEX system was developed and released in 1993 as a L<sup>A</sup>T<sub>E</sub>X tool for drawing structural formulas. The manual was published as a book in 1997 [11]. However, it was not until version 4.00 that the X<sup>M</sup>TEX system supported the PostScript-compatible mode for drawing structural formulas for high-quality printing [12].

The PostScript-compatible mode was applied to prepare a book for surveying organic compounds for color photography [13]. Along with chemical or mathematical equations, this book contains 480 figures, each of which consists of several structural formulas drawn by the X<sup>M</sup>TEX system.

The book published in 2007 deals with a new concept, *mandalas*, which I have proposed as a basis for rationalizing enumeration of three-dimensional structures [14]. This book contains many mathematical equations as well as structural formulas because of its interdisciplinary nature; the mathematical equations were again typeset by the original (L<sup>A</sup>)T<sub>E</sub>X utilities, but this time the structural formulas were drawn by the X<sup>M</sup>TEX system.

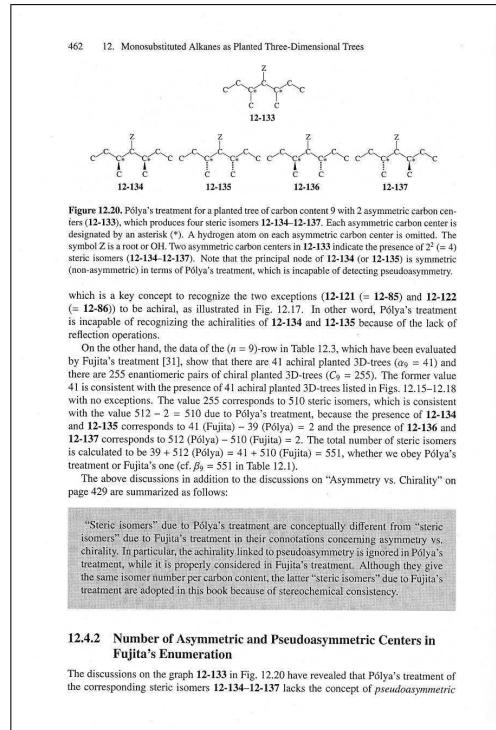
The book published in 2013 is concerned with the *proligand method*, in which I have proposed to enumerate three-dimensional structures [15]. This book indicates that the proligand method for enumerating three-dimensional structures can be degenerated into Pólya's method for enumerating graphs.

A sample page shown in Fig. 1 (page 462 of [15]) contains structural formulas drawn by the X<sup>M</sup>TEX system, while another sample page shown in Fig. 2 (page 463 of [15]) contains mathematical equations typeset by the original utilities of the L<sup>A</sup>T<sub>E</sub>X system.

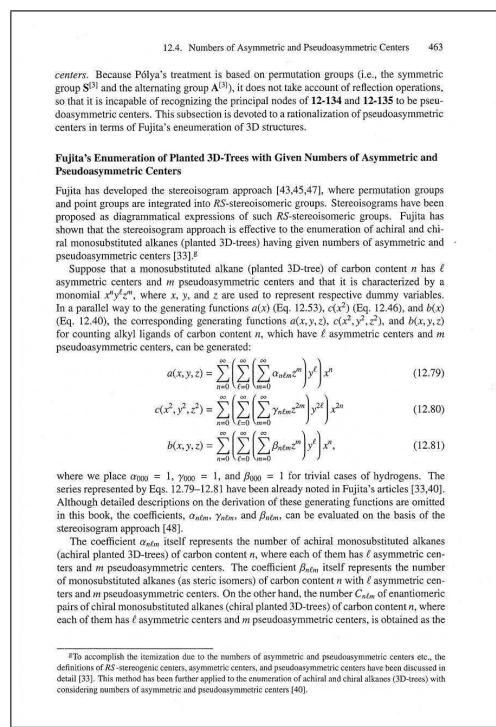
These sample pages from [15] demonstrate that the combination of the X<sup>M</sup>TEX system with the L<sup>A</sup>T<sub>E</sub>X system is an efficient tool for publishing interdisciplinary chemistry/mathematics books.

Moreover, the on-line manual [1] of the X<sup>M</sup>TEX system itself provides us with an illustrative example for publishing a book which contains both chemical structural formulas and mathematical equations. For example, several structural formulas drawn by the X<sup>M</sup>TEX system are aligned in an `align` environment of the `amsmath` package bundled with the L<sup>A</sup>T<sub>E</sub>X system, so as to generate a reaction scheme, as shown in Fig. 3 (page 647 of [1]).

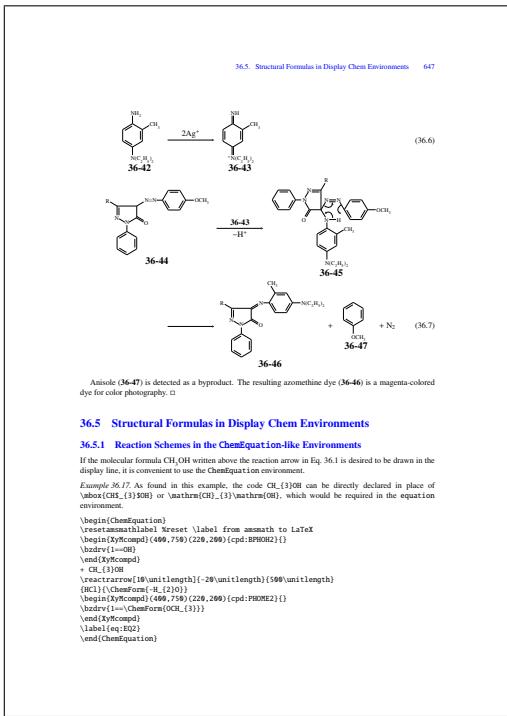
Because X<sup>M</sup>TEX version 5.01 supports utilities for coloring structural formulas, the book published in 2001 would be rewritten with maintaining bond colors (par-bonds, out-bonds, and in-bonds). This has been briefly discussed in Section 39.4 of the on-line manual [1].



**Figure 1:** Sample page containing structural formulas drawn by the X<sup>M</sup>TEX system (page 462 of [15]).



**Figure 2:** Sample page containing mathematical equations (page 463 of [15]), which are typeset by the original utilities of the L<sup>A</sup>T<sub>E</sub>X system.



**Figure 3:** Sample page of the XyMTEX manual [1] (page 647), which contains a reaction scheme drawn by the XyMTEX system and aligned in an `align` environment of the `amsmath` package.

## 4 Conclusion

As clarified by the publication of the interdisciplinary chemistry/mathematics books described above, the XyMTEX system coupled with the L<sup>A</sup>T<sub>E</sub>X system has been proven to be a reliable tool for publishing books of high printing quality which contain structural formulas along with mathematical equations.

## References

- [1] S. Fujita, “XyMTEX: Reliable Tool for Drawing Chemical Structural Formulas,” Shonan Institute of Chemoinformatics and Mathematical Chemistry, Kanagawa (2013), <http://xymtex.com/fujitas3/xymtex/indexe.html>.
- [2] S. Fujita, *J. Chem. Inf. Comput. Sci.*, **26**, 205–212 (1986).
- [3] S. Fujita, *Theor. Chim. Acta*, **76**, 247–268 (1989).
- [4] S. Fujita, *J. Am. Chem. Soc.*, **112**, 3390–3397 (1990).
- [5] S. Fujita, *J. Org. Chem.*, **69**, 3158–3165 (2004).
- [6] S. Fujita, *Tetrahedron*, **60**, 11629–11638 (2004).
- [7] S. Fujita, *Theor. Chem. Acc.*, **113**, 73–79 (2005).
- [8] S. Fujita, *J. Math. Chem.*, **42**, 481–534 (2007).
- [9] S. Fujita, “Symmetry and Combinatorial Enumeration in Chemistry,” Springer-Verlag, Berlin-Heidelberg (1991).
- [10] S. Fujita, “Computer-Oriented Representation of Organic Reactions,” Yoshioka-Shoten, Kyoto (2001).
- [11] S. Fujita, “XyMTEX—Typesetting Chemical Structural Formulas,” Addison-Wesley Japan, Tokyo (1997).
- [12] S. Fujita, *J. Comput. Chem. Jpn.*, **4**, 69–78 (2005).
- [13] S. Fujita, “Organic Chemistry of Photography,” Springer-Verlag, Berlin-Heidelberg (2004).
- [14] S. Fujita, “Diagrammatical Approach to Molecular Symmetry and Enumeration of Stereoisomers,” University of Kragujevac, Faculty of Science, Kragujevac (2007).
- [15] S. Fujita, “Combinatorial Enumeration of Graphs, Three-Dimensional Structures, and Chemical Compounds,” University of Kragujevac, Faculty of Science, Kragujevac (2013).

◊ Shinsaku Fujita  
Shonan Institute of Chemoinformatics  
and Mathematical Chemistry  
<http://xymtex.com>

## TANSU—A workflow for cabinet layout

Pavneet Arora

### Abstract

A workflow using ConTeXt and Asymptote to design cabinet layouts and evaluate the impact of the design to costing is discussed. This work builds on the YAWN workflow which suggested the use of TeX as the “View” in a Model-View-Controller framework.

### 1 Introduction

At the TUG 2012 conference and in *TUGboat* 33:2 (2012), the YAWN workflow [1] was presented. It emphasized considering TeX as the “View” in a Model-View-Controller framework.

YAWN, however, focused on the steps leading up to the final document production. It concerned itself more with the *Model* and the *Controller*, the former utilizing YAML [2] and the latter utilizing Ruby, than with the layout of the resulting document, which was a simple text document akin to what might have come off a line printer.

This article extends the work of YAWN to take on the problem of cabinet layout, with a particular emphasis on the presentation of the final document. It does so while continuing to use the elements of YAWN as a design pattern.

It is natural for a client to seek alternative design layouts for storage units in areas such as kitchens, offices, and libraries even before awarding a project to a specific vendor. The layout is invaluable in visualising how the space might be used, and what options can fit in the allotted space. An adjunct to the layout is, of course, the question of cost. How much will a change in the layout cost?

The challenge for the prospective vendor, on the other hand, is to offer up sufficient detail to the client without over-committing resources to this exercise, since this is a fixed expense which may or may not be recouped later on. Shortening the time required to generate these layouts and associated costs affords a vendor two advantages: first, an opportunity to explore layout options alongside a customer, and second, a way to monitor the implications of the layout changes to the production cost, thus ensuring the viability of their own quotation.

Even when a layout is decided upon, it is useful to evaluate the offerings from different vendors against the design to compare costs. Again, as advocated in YAWN, decoupling the configuration from the catalogue of components from different manufacturers allows *rapid estimating*.

### 2 TANSU

In homage of the location of TUG 2013, Tokyo, the cabinet layout workflow is named TANSU—*tansu* (タンス) being the Japanese word for the traditional storage unit.

TANSU—the acronym—is derived from the following:

- TeX and
- Asymptote driven
- Nomenclature for
- Storage
- Unit layout.

### 3 Specification

The specification of the layout is a straightforward YAML file. Here is an example:

```
:projectID: 1923IMPHOT
:projectAddress: |
    Imperial Hotel Apartments
    Frank Lloyd Wright Edition (1923)
    Tokyo, Japan
:clientName: Okura Kihachiro
:cabinetSpec:
    :manufacturer: Fabritec
    :series: EuroStyle
    :walls:
        -
            :wall:
                :name: East
                :baseCabinets:
                    -
                        HD30844D
                    -
                        B2D24
                    -
                        S24
                    -
                        BSD30
                    -
                        HD1584-R
                :wallCabinets:
                    -
                        HD30844D
                    -
                        W1230
                    -
                        W2430
                    -
                        W1230-R
                    -
                        W3015HZ
                    -
                        HD1584-R
```

The project in question is the fictitious Imperial Hotel Apartments. Baron Kihachiro, one of the investors in the Frank Lloyd Wright-designed Tokyo Imperial Hotel of 1923, has decided to develop a set of apartments along the same lines. However, there is no budget for the custom cabinets that Wright would no doubt insist upon, so we have been approached to come up with design options and their associated costs.

Each wall is given as an array of *base* cabinets (cabinets affixed to the ground) and *wall* cabinets

(cabinets mounted on a wall). The hybrid, *tall* cabinets which extend from floor to top, e.g., HD30844D and B1584-R, need to be listed in both arrays.

The notation assigned to each cabinet is taken here from the catalogue of the specified manufacturer, Fabritec, but it is generic enough to be applied to the offerings of other manufacturers. So, for instance, B2D24 indicates a 2-drawer cabinet that is nominally 24 inches wide, a common cabinet option.

In building catalogues for different manufacturers or even different cabinet series from the same manufacturer, the same notation, once decided upon, can be applied across all the catalogues.

Often, a single cabinet model number is used without an indication of door swing, which is switchable during the field installation. However, for the purpose of visualisation a suffix in the specification allows the layout to indicate intent as in W1230-R for a *right-hinged* door; the default door swing is taken to be *left-hinged*, as in the cabinet W1230 where no suffix is given.

#### 4 Catalogue

A catalogue is similarly constructed using YAML, a sample cabinet from which is given here:

```
:cabinets:
  :subcategory: Cabinets
  :items:
    -
      :model: BD24
      :width: 24"
      :height: 30 1/4"
      :depth: 23 5/8"
      :doors:
        -
          -
            :swing: :drawer
            :width: 24"
            :height: 4"
        -
          -
            :swing: :left
            :width: 12"
            :height: 26 1/4"
        -
          -
            :swing: :right
            :width: 12"
            :height: 26 1/4"
      :desc: 1-drawer 24"W base cabinet
      :price: 362.27
```

One thing that is immediately apparent is a need to deal effectively with *customary units* of length, i.e., feet-inches-fractional inches. Fortunately, there exists a succinct — and one I would consider beautiful and even poetic because of it — regular expression

that does just that (with some slight modification for use in Ruby) [4] returning the matched parts:

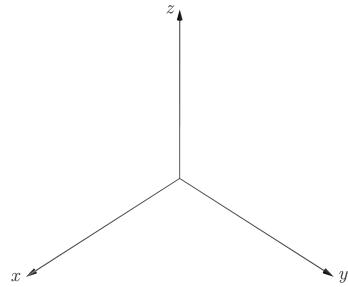
```
re=%r{(?:(?:(<Feet>\d+)[ ]*(?:'|ft)){0,1}[ ]
*(<Inches>\d*(?![\/\w])){0,1}(:[ ,\-\]){:0,1}
(?:<Fraction>(<FracNum>\d*)\/(?:<FracDen>\d*))|{:0,1}{(<Decimal>\.d*){:0,1}{(:\x22| in))}|{(:<Feet>\d+)[ ]*(?:'|ft)[ ]*){:1}}
```

The constituent parts of each cabinet are specified in the `:doors:` hash: they consist of rows of drawers and doors represented as a two-dimensional array. In TeX terms, they may be thought of as cells in a table, some of which span columns (only). So in this case, the first row from the top is a single drawer 24-inches wide by 4-inches high. The next row consists of two doors hinged to the outside frame swinging out from the middle.

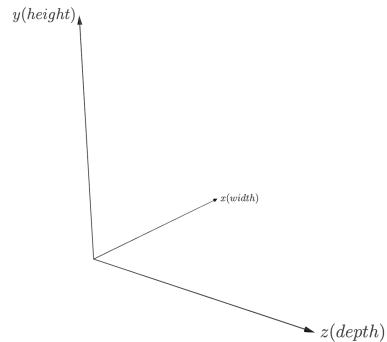
#### 5 Output

The *Model* consists of the aforementioned *specification* and *catalogue*, while the controller is implemented in Ruby to analyse the two.

TANSU builds up a wall layout by retrieving the catalogue entry for each cabinet in the specification, then mapping the cabinet dimensions in the coordinate space of Asymptote [3]. The native coordinate system for Asymptote is the traditional mathematical one (the difference in apparent label size is due to 3D perspective):



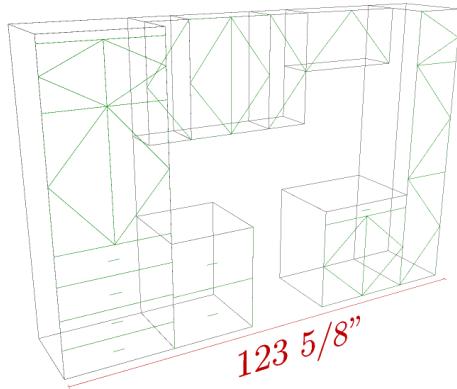
However, for cabinet layout the natural coordinate system is the following:



Once the cabinet box is drawn, doors are overlaid and their swings indicated.

What then is the output? Well, a distinct advantage of using Asymptote is that not only can it handle three-dimensional drawing with aplomb and produce EPS output from the drawing instructions, but it also has a built-in OpenGL renderer allowing one to perform transformations interactively, such as rotations and changes to the viewport. This makes it simple to explore the generated layout from different angles. Here is the rendered layout for the above specification:

Client Name ID:	Okura Kihachiro
Project ID:	1923IMPHOT
Project Address:	Imperial Hotel Apartments Frank Lloyd Wright Edition (1923) Tokyo, Japan



The associated costs for this layout are typeset using ConTEXt's *Natural Tables* mechanism [5] and shown in the report as follows:

#### Base Cabinets

Model No.	Description	Price
HD30844D	30" x 84" tall cabinet with 4 drawers	\$1,192.89
B2D24	2-pot drawer 24"W base cabinet	\$362.98
S24	24"W base cabinet opening	\$250.64
BSD30	24"W sink cabinet with drawer face	\$344.77
HD1584	15" x 84" tall cabinet	\$501.75
	Sub-total	\$2,653.03

#### Wall Cabinets

Model No.	Description	Price
W1230	12" wall cabinet	\$136.13
W2430	24" wall cabinet with glass door	\$254.80
W1230	12" wall cabinet	\$136.13
W3015HZ	24" x 15" bridge cabinet, top hinge	\$176.93
	Sub-total	\$703.99

\$3,357.02

TANSU demonstrates that the ideas described in YAWN, namely of considering TeX as the *View* in an M-V-C framework and representing the *Model* in YAML, offers an extensible methodology with wide-ranging applications. In a completely different domain from that used to initially demonstrate YAWN the same workflow has been used with success.

## References

- [1] Pavneet Arora. YAWN—A TeX-enabled workflow for project estimation. <http://tug.org/TUGboat/tb33-2/tb104arora.pdf>, 2012.
- [2] Oren Ben-Kiki, Clark Evans, and Ingry. YAML Ain't Markup Language (YAML) version 1.2. <http://www.yaml.org/spec>, October 2009.
- [3] John Bowman et al. Asymptote: The vector graphics language. <http://asymptote.sourceforge.net>.
- [4] Normand Frechette. Feet-inch to Decimal. [http://regexlib.com/REDetails.aspx?regexp\\_id=2127](http://regexlib.com/REDetails.aspx?regexp_id=2127).
- [5] Hans Hagen et al. ConTEXt natural tables. <http://wiki.contextgarden.net/TABLE>.

◊ Pavneet Arora  
pavneet\_arora (at)  
bespokespaces dot com  
<http://blog.bansisworld.org>

---

## Bibulous — A drop-in BIBTEX replacement based on style templates

Nathan Hagen

### Abstract

BIBTEX has long been an essential tool for TeX and LATEX users, but the long list of re-implementation efforts attests to unfulfilled needs. The Bibulous project attempts to fulfill many of these needs with a unique approach based on style templates, providing a flexibility that is unmatched by existing tools. We illustrate its capability with examples of custom bibliography styles, multilingual bibliographies, glossaries, and more.

### 1 Introduction

For beginning LATEX/BIBTEX users, the difficulty of building and customizing bibliography style files can come as a shock. BIBTEX’s style files are written in an old-style stack-based language that requires significant effort to learn, edit, and write. It is natural, however, to feel that since bibliographies are highly structured one *should* be able to specify them simply and not, as one commonly finds with BIBTEX’s bibliography styles, in files with over a thousand lines of difficult code. The Bibulous project shows that with style templates one can indeed specify complete bibliography styles in a matter of only a few lines of text, using a format understandable even for novices. Style templates provide a flexibility unmatched by alternative approaches to creating bibliographies, and have important advantages over even the more modern approach taken by Biblatex [1] and Biber [2, 3].

In addition to style templates, Bibulous also implements many of the modern enhancements to BIBTEX, such as the ability to work smoothly with languages other than English, better support for non-standard bibliography structures and heterogeneous databases, and increased formatting options, among others. Moreover, one can use the same framework to generate any of a bibliography, glossary, nomenclature, list of symbols, and/or more, by specifying a different style template for each case.

### 2 Template example

A short example illustrates how templates work. For a simple bibliography consisting of only journal articles and books, a complete style file may consist of just a few lines, as shown in Figure 1. Angle brackets indicate a template variable, which can be (a) a variable (*e.g.* `<year>`) that maps to a field (*e.g.* `year = {·}`) in the database entry; (b) a “special” variable that is generated by the program, *e.g.* `<aу>` is gen-

erated by formatting the database entry’s `author = {·}` field into a list of names; or (c) a variable that is defined by the user within the template’s variable definitions. Each variable represents a string to be inserted into the template at that point, so that for a database entry with `year = {1988}`, a template containing `(<year>)` is replaced with `(1988)`. The same procedure follows for each variable found within the template definition.

Any variable that is not wrapped within square brackets is considered a *required* variable: if this field is missing from the bibliography database entry, then a warning message is used in its place, with the default warning message being simply `???`.

The `[·|·]` bracket notation behaves like an `if ... elseif...` statement: before performing any substitutions of variables into the template, the contents of the first block — located inside square brackets `[·]`, or between the open square bracket and the first vertical bar `[·|` — are checked to see whether all variables within it are defined within the entry. If any variables inside the block are undefined (missing from the database entry), then the second block (between the vertical bar and the close bracket `|·]`, is evaluated. And so on. This provides a flexible way of modifying the inputs to the formatted reference depending on which fields exist within a given database entry. Thus, while some users may prefer to define a given field as `institution`, and others may prefer `organization`, a structure such as

`[<institution>|<organization>]`

provides a simple means of accommodating both.

Finally, a structure that ends with an empty block (`|` rather than `]`) means that even though the individual blocks are optional, at least one of the blocks is required to be defined within the database entry. Thus, for example, the required variable `<note>` is equivalent to `[<note>|]`.

To see how Bibulous uses a template to generate `.bb1` file entries, we can walk through the definitions given in Fig. 1. First, from the `.aux` file, Bibulous obtains a list of citation keys that map to database entries. Using the `sortkey` template, it generates a key for each reference, sorts the keys into the desired order, and then walks through each corresponding database entry, using an appropriate template to format each reference in turn. In the case of the example shown, the sorting key is simply the numbering in which the entry was cited (`<citemum>`). When Bibulous locates an entry in the database, it reads the `entrytype` (this particular style file assumes that only `entrytypes article and book` exist in the citation list) to locate which template definition to use. If

```

TEMPLATES:
article = <au>, \enquote{<title>}, \textit{<journal>} \textbf{<volume>}: ...
    [<startpage>--<endpage>|<startpage>|<eid>|] (<year>). [ <note>]
book = [<au>|<ed>|], \textit{<title>} (<publisher>, <year>)...
    [, pp. ^<startpage>--<endpage>]. [ <note>]
SPECIAL-TEMPLATES:
sortkey = <citemum>
citelabel = <citemum>
```

**Figure 1:** A style template file example. An ellipsis (...) at the end of a line indicates a line continuation. All entries in the “Special Templates” section of the file are optional, and are used to replace default settings.

the current database entry is an `article` entrytype, Bibulous will find the list of variables in the `article` template definition and begin replacing them one by one with their corresponding fields. The `<au>` variable represents a formatted string of author names (formatted from the `author` field according to default options, in this case), followed by a comma and a space. If no `author` field is found in the bibliography entry, then it inserts “???” there to indicate a missing required field. Next it locates the `title` field and inserts it, with a comma, inside `\enquote{}` formatting instructions. Next follows an italicized journal name, and a boldface volume number. All of these so far are designated as required fields. Next, if the `pages` field is found in the entry’s database, Bibulous will parse the start and end page numbers (assuming that the `pages` field uses dash-delimited page identifiers) and insert them here using an en-dash separator. If the `pages` field contains only one page, then the `endpage` variable will be undefined, and Bibulous will attempt to use the next block, which uses only the `startpage`. If the `pages` field is missing entirely from the database entry, then Bibulous checks to see if the `eid` (electronic identifier) field is defined, and if so uses it instead. However, if the `pages` and `eid` fields are both undefined, then the code inserts “??”. Finally, the template instructs putting the `year` inside parentheses, and if a `note` field is defined in the entry, then it is added onto the end (following the period). If `note` is not defined, then Bibulous adds nothing.

The `book` template in the example follows a similar procedure, but has different required and optional fields. The list of names at the beginning of the reference, for example, can be either a list of authors or a list of editors (with preference given to authors, if defined in the entry). The `book` template also requires a `publisher` field to be inserted inside the parentheses with the publication year.

The `sortkey` “special template” allows users flexibility in defining how they want their citations

sorted. The default is simply to sort by numerical order (`sortkey = <citemum>`), but one can also choose to sort by author name, then title, then year, as in

```
sortkey = <authorlist.0.last>...
    <authorlist.0.first>...
    [<title>|<booktitle>]<year>
```

or, if a user wants the entry to define a special sort for any given entry, then they can use any fields within the entry that they wish. For example, to have an author sorted under a different name than is used in the reference, one can add a `sortname` key to the database entry and redefine the `sortkey` template as

```
sortkey = [<sortname>|<authorlist.0.last>...
    <authorlist.0.first>|[]<title>|...
    <booktitle>]<year>
```

And likewise for any other field such as `sorttitle` or `sortyear` for modifying citation sorting.

The final item that Bibulous needs to specify for the reference is its item label—the label that goes at the front of the reference, and is often the same as the `sortkey`. This label is generated using the `citelabel` template. For scientific journal articles the most common choice for this is the citation number (e.g. `citelabel = <citemum>`). For other areas of study, the reference labels may, for example, take the form of the first author’s last name, then first name, then year, as in the following example:

```
citelabel = \textbf{<authorlist.0.last>,
    <authorlist.0.first> (<year>).}
```

This can produce a reference list like the following:

**Schmader, Toni (2002).** Gender identification moderates stereotype threat effects on women’s math performance. *Journal of Experimental Social Psychology*, **38**, 194–201.

**Steele, Claude (1997).** A threat in the air: How stereotypes shape intellectual identity and performance. *American Psychologist*, **52**, 613–629.

```

TEMPLATES:
article = <au>. \enquote{[\href{<url>}<title>]|\href{<doi>}<title>|<title>|],} ...
    \textit{<journal>} <volume>(<number>): [<startpage>--<endpage>|...
        <startpage>|<eid>|] (<year>).

inproceedings = <au>, \enquote{[\href{<url>}<title>]|\href{<doi>}<title>|...
    <title>|],} in \textit{<booktitle>}[, <ed.if_singular(editorlist, edmsg1, ...
    edmsg2)] [, in <series>][, vol.^<volume>][, pp.^<startpage>--<endpage>|, ...
    <eid>] (<year>).

SPECIAL-TEMPLATES:
authorlist = <author.to_namelist()
editorlist = <editor.to_namelist()

authorname.n = [<authorlist.n.first.initial().]<authorlist.n.middle.initial().]...
    [<authorlist.n.prefix>]<authorlist.n.last>[, <authorlist.n.suffix>]
au = <authorname.0>, ..., { and }<authorname.9>
editorname.n = [<editorlist.n.first.initial().]<editorlist.n.middle.initial().]...
    [<editorlist.n.prefix>]<editorlist.n.last>[, <editorlist.n.suffix>]
ed = <editorname.0>, ..., { and }<editorname.3>

OPTIONS:
etal_message = , \textit{et~al.}
edmsg1 = , ed.
edmsg2 = , eds

```

**Figure 2:** Two example entrytype templates for an IEEE format bibliography. For the hyperlinked title, the template allows use of either a `url` or a `doi` field in the database entry.

Figure 2 shows how Bibulous can be used to develop custom bibliography styles, using an example that incorporates hyperlinks into the document title of each reference. In going from the style used in Fig. 1 to the new `article` style in Fig. 2, we have to convert from using italics for titles to using quotation marks around a hyperlinked text (if a `url` or `doi` field is provided in the entry), from using boldface to standard typeface for the volume, and adding an issue number in parentheses. The `inproceedings` style uses the same format for hyperlinked titles, and adds the ability to format fields such as `booktitle` and `series`, and provides a separate list of editors. For BIBTEX, modifying an existing style file to achieve these changes would be a daunting task for a casual user. Biblatex has succeeded in making this much easier, but with templates it can be easier still.

The other major change in going from Fig. 1 to Fig. 2 is in the special template definitions, where we can find several new constructions. The first is a “dot” operator placed after a variable inside the angle brackets. This can be used to apply an index—as in the case of a list or a dictionary type of variable—or an operator, as in `.to_namelist()` or `.initial()`. An index can be *explicit*, as in `<authorlist.0>`, which indicates taking the zeroth element of the list-type variable `authorlist`, or *implicit*, as in `editorname.n`. For an implicit index, the construction `<editorname.0>` used in conjunc-

tion with the variable definition for `editorname.n` indicates that all instances of the implicit index `n` in the template should be replaced with the explicit index 0. Finally, an ellipsis occurring in the *middle* of a string indicates an implicit loop. Thus, the definition

`<editorname.0>, ..., { and }<editorname.3>`

is equivalent to `<editorname.0>` when there is only one editor in the database entry, or equivalent to

`<editorname.0> and <editorname.1>`

when there are only two. For three editors, the implicit loop expands the template to

`<editorname.0>, <editorname.1>, and  
<editorname.2>`

and for four,

`<editorname.0>, <editorname.1>,  
<editorname.2>, and <editorname.3>`

Since the template does not specify the format for more than four editor names, the code builds an *et al.* construction when there are more than the specified number of names, so that the result becomes

`<editorname.0>, <editorname.1>,  
<editorname.2>, <editorname.3>,  
\textit{et~al.}`

where the form of the string “, et al.” is specified by an `etal_message` keyword option.

The implicit index and implicit loop features are only necessary when one needs to customize the

moviedb.bib	moviedb.bst
<pre>@movie{key,   title = {...},   director = {...},   year = {...},   rating = {...} }</pre>	<pre>TEMPLATES: movie = \nstars{&lt;rating&gt;} \color{blue}{...   &lt;title&gt;}\color{black}, &lt;director&gt; (\year). SPECIAL-TEMPLATES: editor = &lt;director&gt; editorlist = &lt;editor.to_namelist() sortkey = &lt;-year&gt;&lt;title&gt;&lt;editorlist.0.last&gt; citelabel = None OPTIONS: bibitemsep = Opt</pre>
<pre>%% In preamble: \usepackage{expl3} \ExplSyntaxOn \cs_new_eq:NN \Repeat \prg_replicate:nn \ExplSyntaxOff \newcommand{\nstars}[1]{\parbox{3em}{% \Repeat{#1}{\$\star\$}}}</pre>	<pre>1. ***** <a href="#">The Inheritance</a>, Per Fly (2003). 2. **** <a href="#">The Celebration</a>, Thomas Vinterberg (1998). 3. * <a href="#">The Kingdom</a>, Lars von Trier (1994).</pre>

**Figure 3:** An example illustrating the use of Bibulous’ style templates for a movie database. The citation sorting order is defined to be by year in descending order (the minus sign within the variable name inverts the direction of the sort), followed by title and then director’s last name. Note that the `editor = <director>` special template is necessary in order to map the `director` field to the `editor` field.

namelist formats with more flexibility than that allowed by Bibulous’ standard keyword arguments. Thus, for most styles these structures are not necessary, and one can use the default namelist format, modified through keyword arguments.

The `.if_singular(arg1, arg2, arg3)` operator is unique in that it accepts arguments. This allows it to insert `arg2` if `arg1` is singular, or `arg3` if plural. Here, the operator is given the argument `<editorlist>`, which has one entry if the `editor` field in the database entry contains only one name. In this case it returns the argument `edmsg1`, defined in the options list as “`, ed.`”. If more than one editor name is found in the database entry, then the operator returns the argument `edmsg2`, defined in the options list as “`, eds`”.

### 3 Unorthodox bibliography styles

Templates allow easy customization of how BIBTEX-format database entries are displayed. Fig. 3 shows an example .bib database structure, and a corresponding template file, for the display of a movie database. This simple example defines a LATEX command `\nstars{·}` for printing  $N$  stars as a display of the `rating` variable, shows the use of color commands, and works with new database fields. Thus, any LATEX markup the user wishes to incorporate into a bibliography, or into any database formatter, can be implemented in Bibulous’ style templates.

Another important feature of style templates is that they are language agnostic. For example, if we replace

```
TEMPLATES:
article = <au>, \enquote{...
book = [<au>|<ed>...
```

from the style template with

```
学術論文 = <au>, \enquote{...
本 = [<au>|<ed>...
```

(学術論文 and 本 are the Japanese words for “journal article” and “book”), then we can use the Japanese forms for every entry in the bibliography database, so that entries in the .bib file would have the form

```
@学術論文{entrykey, ...
@本{entrykey, ...}
```

Since this shows only the entry types and not the database fields in a non-English language, this still shows an English-language-centric implementation, but here is a fully Japanese style template:

```
本 = <著者名>, <題名> (<年>).
```

with a corresponding example database file entry

```
@本{漱石1906,
  題名 = {草枕},
  著者名 = {夏目 漱石},
  年 = 1906,}
```

The important feature here is that Bibulous does not need to know anything about the languages used in the database, the template, or the .tex file. It only needs to be able to convert the symbols to Unicode so that it can match any symbol from one file to that in another. And since Bibulous’ internals are written entirely in UTF-8 compliant code, multilingualism and localization come easily. Bibulous is even flexible enough to allow the use of

```

TEMPLATES:
collection = [<for_japanese_audience><author_ja>, <title_ja>[, <volume>卷] ...
    <publisher_ja> <year>). | <for_english_audience><author_en>, <title_ja> ...
    (<title_romaji>) {\makeopenbracket}<title_en>{\makeclosebracket}...
    [, vol.^<volume>] (<publisher>, <address>, <year>). | <for_english_audience>...
    <author>, <title>[, vol.^<volume>] (<publisher>, <address>, <year>)|]

OPTIONS:
document_language = Japanese ## whether the document is for a Japanese audience

VARIABLES:
for_japanese_audience = '{}' if (options['document_language'] == 'Japanese') else None
for_english_audience = '{}' if (options['document_language'] == 'English') else None

```

**Figure 4:** Template file for the multi-language bibliography example.

mathematics markup within keys and labels, even though L<sup>A</sup>T<sub>E</sub>X itself currently cannot handle this.

Another example of Bibulous' versatility in dealing with multiple languages is that of an “audience switch”. The following example is adapted from a query made online [4]. For a single database file that can be used inside both English-language *and* Japanese-language publications, an English-language format should allow Japanese entries in the bibliography with author names in both *kanji* and romanized characters. Titles should also be given in this dual form, but sometimes with an English translation provided. English-language citations should appear as normal. In a Japanese-language publication, however, the Japanese entries should have the author, title, etc. in Japanese form without roman letters or English. Here is an example database entry:

```

@collection{Tsuruta2006,
    address_ja = {東京},
    address_romaji = {Tōkyō}
    author_ja = {鶴田 匠夫},
    author_en = {Tadao Tsuruta}
    publisher_ja = {新技術コミュニケーションズ},
    publisher_en = {Shin'gijutsu
        Communications},
    title_ja = {光の鉛筆},
    title_romaji = {Hikari no empitsu},
    title_en = {Pencil of light},
    volume = 7,
    year = 2006}

```

The corresponding style template file is shown in Fig. 4. The template is structured into two large optional blocks, and the selection of one or the other is decided by the `document_language` option variable defined by the user within the file. The formatted result will look something like the following for a Japanese audience

1. 夏目 漱石, 草枕 (春陽堂 1906).
2. 柳田 聖山, 禪學叢書, 10巻 (中文出版社 1974–1977).
3. 鶴田 匠夫, 光の鉛筆, 7巻 (新技術コミュニケーションズ 2006).

and the following for an English-language audience

1. Soseki Natsume, 草枕 (Kusamakura) [Pillow of Grass] (Shun'yōdō Publishing, Tōkyō 1906).
2. Seizan Yanagida, 禪學叢書 (Zengaku sōsho) [Collected Materials for the Study of Zen], vol. 10 (Chinese Book Publ. Co., Kyōtō 1974–1977).
3. Tadao Tsuruta, 光の鉛筆 (Hikari no empitsu) [Pencil of Light], vol. 7 (Shin'gijutsu Communications, Tōkyō 2006).

Thus, the template shown in Fig. 4 is not only flexible enough to handle custom database types and heterogeneous fields, but also different audience languages. Note that the example uses fields in English rather than in Japanese (*e.g.* `author_ja` rather than 著者名) in both the database and the template in order to illustrate the overall structure clearly to readers unfamiliar with Japanese.

#### 4 Implementing a glossary

The utility of style templates extends beyond bibliographies. Glossaries, lists of symbols (*i.e.* nomenclatures), and lists of acronyms are readily generated with Bibulous. Figure 5 shows an example `.tex` file, together with corresponding `.bib` and `.bst` files, and output `.bbl` file. Together, these create the formatted glossary etc. shown in Fig. 6.

In the example `.bib` file, readers may note the separation of the `symbol` entrytypes into independent `name` and `description` fields, instead of the form that the acronym entrytypes use. This is necessary to get around the limitation that L<sup>A</sup>T<sub>E</sub>X cannot use mathematical markup within citation keys.

```

gloss.tex
\documentclass{article}

\newcommand{\citetname}[1]{#1%
  \protect\nocite{#1}%

\makeatletter
\renewcommand{\biblabel}[1]{#1}
\renewenvironment{thebibliography}[1]
{\section*{\refname}%
  \list{}{\setlength{\labelwidth{1.5cm}}%
    \leftmargin\labelwidth
    \advance\leftmargin\labelsep
    \let\makelabel\descriptionlabel}%
\endlist}
\makeatother

\renewcommand{\refname}{Glossary, List of %
Symbols, and Nomenclature}

\begin{document}

While \citetname{Tilt} aberration changes only
the magnification, \citetname{SA} induces
image blur. Wavefront aberration is a
function of radial distance \cite{sym:rho}
and azimuthal angle \cite{sym:phi}. Of the
two aberrations, only \citetname{SA} has an
effect on the \citetname{PSF} and
\citetname{MTF}.

\bibliographystyle{gloss}
\bibliography{gloss}
\end{document}

```

```

gloss.bib
@symbol{sym:phi,
  name = "$\phi$",
  description = {Azimuthal angle.}}

@symbol{sym:rho,
  name = "$\rho$",
  description = {Radial distance from the
    optical axis.}}

@gloss{SA,
  name = {Spherical Aberration},
  description = {The departure from an ideal
    spherical wavefront that increases
    quadratically with radial distance.}}

@gloss{Tilt,
  name = {Tilt aberration},
  description = {A linear departure from an
    ideal wavefront --- equivalent to a
    magnification error.}}

@acronym{MTF = "Modulation Transfer
function"}
@acronym{PSF = "Point Spread Function"}

```

```

gloss.bst
TEMPLATES:
symbol = <description>.
gloss = <description>.
acronym = <description>
SPECIAL-TEMPLATES:
sortkey = <citekey>
citelabel = <name>
OPTIONS:
bibitemsep = 0pt
replace_newlines = True
case_sensitive_field_names = True

```

```

gloss.bbl
\begin{thebibliography}{6}
\setlength{\itemsep}{0pt}

\bibitem[MTF]{MTF}
Modulation Transfer function
\bibitem[PSF]{PSF}
Point Spread Function
\bibitem[Spherical Aberration]{SA}
The departure from an ideal spherical
wavefront that increases quadratically
with radial distance.
\bibitem[$\phi$]{sym:phi}
Azimuthal angle.
\bibitem[$\rho$]{sym:rho}
Radial distance from the optical axis.
\bibitem[Tilt aberration]{Tilt}
A linear departure from an ideal wavefront
--- equivalent to a magnification error.

\end{thebibliography}

```

**Figure 5:** The main **tex** file, style template (**bst**) file, bibliography database (**bib**) file, and resulting output (**bbl**) file for the glossary example.

Note that, because it is only a back-end engine, Bibulous by itself cannot split the glossary, list of acronyms, and list of symbols into three separate sections of the document. Doing so requires work by L<sup>A</sup>T<sub>E</sub>X itself, and thus requires changes to the *front* end. The example shown here provides

only a minimal amount of front-end work—defining the `\citetname{}` command, as well as redefining the `\biblabel` command and the `\thebibliography` environment—to make the bibliography structure appear as a list of definitions.

## Glossary, List of Symbols, and Nomenclature

<b>MTF</b>	Modulation Transfer function
<b>PSF</b>	Point Spread Function
<b>Spherical Aberration</b>	The departure from an ideal spherical wavefront that increases quadratically with radial distance.
$\phi$	Azimuthal angle.
$\rho$	Radial distance from the optical axis.
<b>Tilt aberration</b>	A linear departure from an ideal wavefront—equivalent to a magnification error.

**Figure 6:** The formatted glossary, list of symbols, and nomenclature (list of acronyms) for the inputs shown in Fig. 5. Note that the symbols  $\phi$  and  $\rho$  are sorted by citation key, which are `sym:phi` and `sym:rho` for the case shown here.

**Table 1:** Timing comparison for BIBTEX, Bibulous, and Biber, showing the amount of time required to generate a `.bb1` file for every entry in the given database.

Tool	Database size (# citations)		
	100	820	12 000
BIBTEX	0.112 sec	0.117 sec	1.01 sec
Bibulous	0.190 sec	1.667 sec	41 sec
Biber	1.115 sec	9.051 sec	17.5 min

## 5 Comparison with existing bibliography engines

Bibulous not only provides easy customization for users, but does so with minimal compromises in speed and without requiring a compiler for installing. It is written in Python and has no external dependencies (it requires only Python’s standard library), making it highly portable and easy to install, since Python is widely available on all modern platforms. And although it is slower than BIBTEX, it is considerably faster than Biber.

To demonstrate processing speeds, each of the three back-end processors were run on three bibliography databases inside Bibulous’ regression testing suite: one containing 12 000 entries, another with 820 entries, and a third with 100 entries. Using a standard desktop computer, running all three processors on these databases produces the timing results shown in Table 1. While the comparison is not entirely fair, since Biber is doing more work (especially by building hash codes for every author in every entry), the results show that Biber’s expanded capabilities have come at a significant sacrifice in speed.

Biblatex and Biber are quickly replacing BIBTEX as the standard tools of choice for LATEX bibliogra-

phy processors, as they provide much-needed functionality that BIBTEX does not: support for larger databases and non-English languages, internationalization, localization, multilingualism, and more. In order to specify a bibliography style, Biblatex uses a combination of keyword options that are set within the main `.tex` file to specify bibliography formats. A weakness of this approach is that a user must know a large set of keywords and how to combine these to generate a custom style. Learning how to format references can require reading and understanding a substantial volume of Biblatex’s documentation. With style templates, however, even first-time users can see how to customize the layout without needing to consult documentation, so that entirely new styles can be written up in a matter of minutes.

Beyond its use of templates, Bibulous has other important differences from BIBTEX, including:

1. Bibulous converts LATEX-markup accented characters to Unicode prior to performing sort functions. That is, while BIBTEX will sort `t\^ete`, `t{\^\e}te`, and `t{\^\{e\}}te` all as `tete` but `t\^ete` separately as `t\^ete`, Bibulous will sort all of these cases identically as `t\^ete`. In addition, Bibulous automatically detects the user’s default locale and sorts according to standard string collation algorithms.
2. Bibulous adds new options for formatting names inside `.bib` file entries, so that individual names are also allowed *three* or *four* comma delimiters. This gives users more freedom to control name layout.
3. Bibulous performs citation extraction by default—the relevant database entries from the main `.bib` files are extracted and placed together into a single (usually small) database. This speeds up program execution when operating with a large main database but a relatively

small number of citations.

4. Bibulous provides an `authorextract` command that creates a `.bib` file containing only those in which the given author's name is referenced. This can be useful for writing a CV.
5. Bibulous' `sentence_case` operator is almost identical to BIBTEX's `change.case$` command, but works with Unicode.
6. Bibulous provides a Python API to its own internal data structures, allowing users to extend or customize the program as they desire, by placing function definitions within style files.

## 6 Comments

Rather than providing a detailed explanation of usage, the aim here has been to introduce Bibulous to LATEX users and to showcase some of its strengths. These include:

1. using the simplicity and power of style templates to specify and customize bibliographies;
2. implementing a fully Unicode-compliant program without serious memory limitations.

The examples provided in the sections above show how these traits can be used to provide a powerful backend bibliography processor.

However, a range of functionality remains that is important for users but is not readily accessible through Bibulous alone. Tasks such as advanced citation labelling, the creation of indexes, etc. require integrating Bibulous with a LATEX package front-end. Bibulous' `authorextract` functionality, for example, makes it easy to create a publications list (such as for a CV), but without front-end integration it has no means of separating reference entries into independent lists. That is, one would often like a `cypublications.bst` style template that will sort references into sections differentiated by entry type (*e.g.* articles, book, inproceedings), with entries in each section sorted in reverse chronological order.

Without integration with a front-end package, Bibulous can only provide a single list as output for LATEX.

Similarly, advanced citation labelling, such as using citation commands like `natbib`'s `\citep{·}` and `\citet{·}`, require that additional information be written to the `.aux` file to inform the bibliography processor about the specific requirements of that specific reference's citation label. This kind of integration is a goal for version 2.0 of Bibulous.

Users interested in learning more details about how to use Bibulous can find much more in the user manual, available at the project website and in the source code repository. The source code for the entire repository (the core `bibulous.py` file plus testing software and documentation) can be found at <https://github.com/nzhagen/bibulous>.

## References

- [1] P. Lehman, A. Boruvka, P. Kime, and J. Wright, “The `biblatex` package: Programmable bibliographies and citations”. Available from CTAN: <http://ctan.org/pkg/biblatex>.
- [2] P. Kime and F. Charette, “`biber`: A backend bibliography processor for `biblatex`”. Available from CTAN: <http://ctan.org/pkg/biber>.
- [3] P. L. Kime, “Biber—the next generation backend processor for BIBLATEX”. *TUGboat* 33: 13–15 (2012).
- [4] <http://tex.stackexchange.com/questions/28010/how-to-create-multilingual-english-japanese-bibliographies-with-biblatex-bib>.

◇ Nathan Hagen  
Tucson AZ, USA  
`nhagen` (at) optics dot arizona dot edu  
<https://github.com/nzhagen/bibulous>

## The multibibliography package

Michael Cohen, Yannis Haralambous and  
Boris Veytsman

### Abstract

Conventional standards for bibliography styles entail a forced choice between index and name-year citations and corresponding references. We reject this false dichotomy, and describe a multibibliography, comprising alphabetic, sequenced, and also chronological orderings of references. An extended inline citation format is presented which integrates such heterogeneous styles, and is useful even without separate bibliographies. Richly hyperlinked for electronic browsing, the citations are articulated to select particular bibliographies, and the bibliographies are cross-referenced through their labels, linking among them.

### 1 Introduction

One of the aims of the list of references in an academic paper or book is to show the reader the current state of the field. A good bibliography creates a narrative, showing the context of the manuscript in the general picture of scientific inquiry—those proverbial “shoulders of giants” on which it stands.

There are two main ways to organize such a narrative: either around the ideas or around the authors. In the first case the order of citations follows the order of their mention in the main text. Thus the logic of the text is reflected in the bibliography list. In the second case the order of citations follows the authorship: we want alphabetic order by authors (with chronological subordering of works by the same authors). Accordingly, the inline citations in the first cases are usually numerical, whereas in the second case they are either numerical or, when possible, based on the authors’ names and publication years (perhaps abbreviated or contracted). This is the main difference between “numerical” and “named” bibliography styles [Daly, 2011: 1]. Both these styles have their own advantages and disadvantages. It is possible to imagine a third option: ordering the citations primarily accordingly to publication year, thus showing the chronology of progress in the field.

One may ask, why not use the advantages of both the currently employed styles, generating not one, but multiple lists of references? In the old days, when bibliographies were created and sorted manually, such a task was prohibitively expensive. This is no longer true.

Encouraged by the programmability of bibliographic styles and the flexibility of compiled for-

matting, we propose an extension of academic and scientific bibliographic styles. Conventional inline bibliographic citations, indicating full references in a separated bibliography, are either ordinal numbers generated according to first appearance in a document or a tag composed of respective authors’ names and publication year. To simultaneously deploy these heretofore mutually exclusive styles, we introduce a “multibibliography,” combining both “numerical” and “named” styles. We also add a chronological list, integrating all the information for the inline citations. This idea was conceived by the first author and implemented partly by the second author and the third author.

Rather than choosing between citations as **index numbers**, corresponding to alphabetically sorted authors’ names, as in BIBTEX’s “plain” style, and in order of first appearance in the document, as in the “unsrt” style, or **author names and publication year** (or some abbreviation thereof), as in the “alpha” style, we use both, mixing the two styles, as in “(Suzuki, 2013: 57)”, or, in case of associated page numbers, “(Suzuki, 2013: 57, p. 45–67)”.

This is admittedly unorthodox, unusual and unique, but satisfies our desire to have an easily understood cross-reference (without ambiguity in the case of name collision) and an ordinal reference (the last entry in the sequential bibliography also serving as a cardinal reference count), and also our preference to be able to see an inline reminder of the respective authors. As a bonus highlighting such usefulness, a “timeline” bibliography is also generated in chronological order.

### 2 Implementation and invocation

Such a multibibliography currently comprises three separate orderings. A Perl script compiles the multibibliography source. Specifically, running “`perl multibibliography.pl <fn>`” instead of `bibtex` (after the 1st-pass “`latex <fn>`” and before the usual 2nd and 3rd passes), generates separate `.bb1` files:

“**apalike**” style, sorted alphabetically, by first author’s family name,

“**unsrt**” style, in order of first appearance in the document, and with the label adjusted to lead with the sequence number, and also

“**chronological**” style, sorted according to date of publication, as in a timeline.

The functionality of the multibibliography package<sup>1</sup> is different from both the `multibib` package,<sup>2</sup>

<sup>1</sup> [www.ctan.org/pkg/multibibliography](http://www.ctan.org/pkg/multibibliography)

<sup>2</sup> [www.ctan.org/pkg/multibib](http://www.ctan.org/pkg/multibib)

which facilitates having separate bibliographies for each chapter in a monograph, and the `multibbl` package,<sup>3</sup> which facilitates separating referenced sources by their language [Mori, 2009: 2].

In `multibibliography.sty`, which should be loaded at the top of any invoking document, the standard “`thebibliography`” command is redefined and new commands called “`bibliographysequence`” and “`bibliographytimeline`” are defined, all of which respectively redefine the `bibitem` command accordingly to generate references in the appropriate format and order. The `chronological bst` file in the package, made with the `makebst` [Daly, 2007: 3] and `docstrip` utilities and using the `merlin.mbs` generic bibliography [Daly, 2011: 1], augments the built-in `apalike` and `unsrt` styles.

At the end of the document, the multibibliography can be rendered thusly:

```
\renewcommand{\bibname}{References sorted by
  name}
\markboth{References sorted by name}{%
  References sorted by name}
\bibliographystyle{apalike}
\addcontentsline{toc}{chapter}{References
  sorted by name}
\bibliography{.bib files}

\clearpage
\renewcommand{\bibname}{References sorted by
  first appearance}
\markboth{References sorted by first
  appearance}{References sorted by first
  appearance}
\addcontentsline{toc}{chapter}{References
  sorted by appearance}
\bibliographysequence{.bib files}

\clearpage
\renewcommand{\bibname}{References sorted
  chronologically}
\markboth{References sorted chronologically}{%
  References sorted chronologically}
\addcontentsline{toc}{chapter}{References
  sorted chronologically}
\bibliographytimeline{.bib files}
```

(For shorter document styles, such as this `article`, `\bibname` should be changed above to `\refname`, and adjustments to the Table of Contents as well as the `\clearpages` may be elided.)

This multibibliography system is copotentiated by the hypertextual `hyperref`<sup>4</sup> package. When using

<sup>3</sup> [www.ctan.org/pkg/multibbl](http://www.ctan.org/pkg/multibbl)

<sup>4</sup> [www.ctan.org/pkg/hyperref](http://www.ctan.org/pkg/hyperref)

them together with an appropriate viewer or browser (such as `xdvi`, `acrobat`, or Adobe Reader), clicking an inline citation jumps to the respective entry in one of the reference lists. As illustrated in Fig. 1, the `multibibliography` inline `hyperref` hotspot is articulated to allow clicking on

**the name**, which jumps to the corresponding entry in the alphabetical bibliography;  
**the index number**, which jumps to the respective entry in the sequential bibliography; or  
**the date**, which jumps to the matching entry in the chronological bibliography.

Similarly, cross-references among the respective subbibliographies are also hyperlinked, although from the labels, and not the `bibitem` bodies of the citations. The “[`backref=page`]” `hyperref` extension<sup>5</sup> is compatible, generating the familiar and useful back-references in all the subbibliographies: lists of clickable page number links associated with each entry in the bibliography pointing back to the respective citations (except those generated by `nocites`). The generation of these back-references, indicated by the hollow arrowheads in Fig. 1, represents “closing the loop” on the fully crossed relation set.

In the future, the date should be articulated to add the month to the `sort.label` in the presort FUNCTION in the `chronological bst` file, since it isn’t one of the built-in keys of the `makebst` package [Markey, 2009: 6], as the `merlin` system didn’t anticipate such fine-grained sortings. However, as [Markey, 2009: 6, p. 13] observes, using the month is somewhat problematic, since it is indicated by a character string, but is really an ordinal. If built-in macros (“`jan`”, “`feb`”, etc.) are used, they can be easily mapped to months and used for sorting, but if, as is often the case, the field is reinterpreted to mean date (bimonthly publications indicated by something like `month = "March & April"`, quarterly dates as `month = "Autumn"`, etc.), this scheme will not easily generalize.

We have not yet experimented with combining this package with other bibliographic packages [Patashnik, 1998: 4] such as `natbib`<sup>6</sup> or `chapterbib`<sup>7</sup> [Kopka and Daly, 2003: 5, p. 217–221].

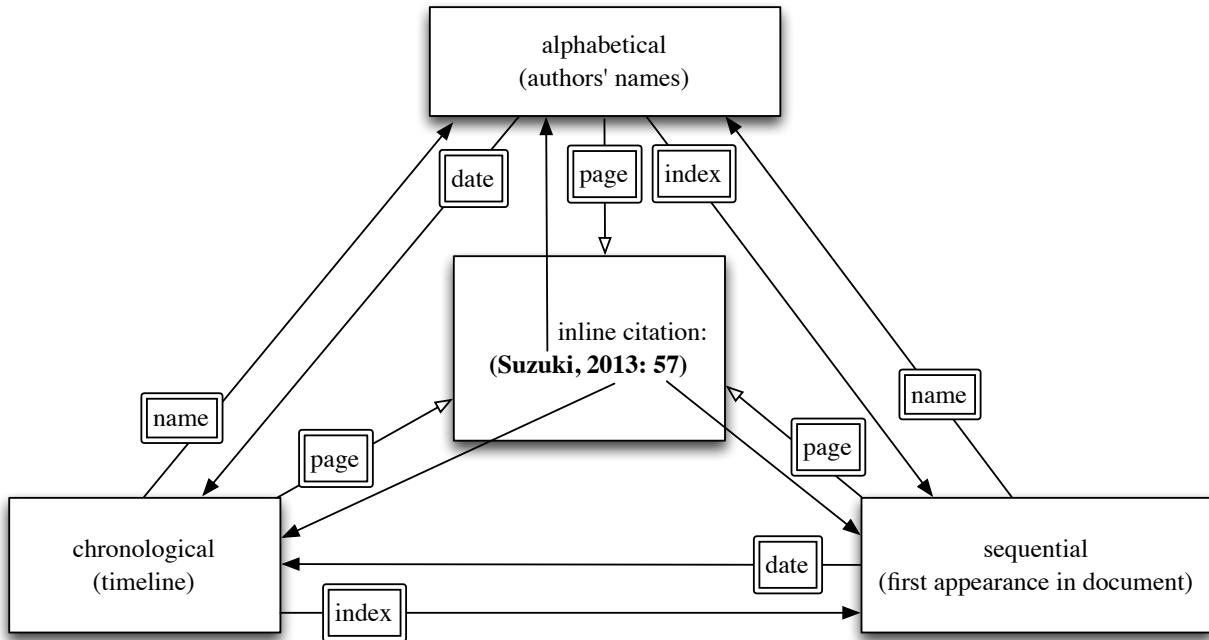
### 3 Implications

The extended inline citation style was designed for the multibibliography, but can be deployed and is useful even without it. The bibliographic dilation is perhaps more appropriate or at least more appealing for electronic dissemination, as traditional

<sup>5</sup> [www.tug.org/applications/hyperref/manual.html](http://www.tug.org/applications/hyperref/manual.html)

<sup>6</sup> [www.ctan.org/pkg/natbib](http://www.ctan.org/pkg/natbib)

<sup>7</sup> [www.ctan.org/pkg/chapterbib](http://www.ctan.org/pkg/chapterbib)



**Figure 1:** Hyperreferential links across document and among multibibliographies: Each inline citation, exemplified by the block in the center, is linked to references in sub-bibliographies, which are cross-linked to each other and can also be linked back to the inline callouts. Hollow arrowheads represent links provided by `hyperref`'s `backref`; solid arrowheads represent links provided by the `multibibliography` package.

print-based publishers might resent the cost of extra pages. The fully crossed hyperreferential links are a convenient way of establishing the context of references, seamlessly expressing citations' appearances in the document and in time.

Bibliographic subsections might be sorted by other ad-hoc keys. Maybe the three “slices” through the bibliographic database that we have organized suffice for most ordinary publishing, but even more styles of bibliographic lists could be crafted, corresponding to special purposes, sorted by attributes such as number of authors, number of pages, conference or journal, location, etc. For simple examples, a subbibliography near the end of an art book could be sorted by name of artist considered by each monograph, or music books could have bibliographies sorted by name of performing group or family or given name of artist. Of course all monographs about each artist would be grouped together in that subbibliography, and each citation would include page numbers for each call-out within the text.

These kinds of lists could be (and often are) explicitly compiled by authors, but only painstakingly. However, such concordances would lack the automatic back references to the call-outs. A mod-

ern, flexible scheme enables various presentations of bibliographic information, so that each references subsection acts as a kind of special index, but with granularity not at the topic level, but at the document level.

Such extended subbibliographies both represent and also re-present references, showing them in fresh and useful settings. There are two related activities encouraged by such recontextualization:

- looking up a particular entry (including page call-outs), and
- exploiting “locality of reference,” so that other related sources are likely to be nearby.

The philosophy is to leverage the power of hyperreferential idioms to augment reading by considering a document as a special kind of database that is indexed in appropriate dimensions, especially including the name–value pairs in its associated bibliographic information (such as that captured by `BIBTeX` files) plus derived information available after compilation (such as sequence number and appearance location).

It is our hope that old-fashioned conventions, established in the context of technological restrictions that have now been overcome, may be relaxed.

We anticipate that future schemes will allow dynamic reordering, as if the bibliography were a spread-sheet-like database, which of course it is. We find this multidimensional presentation useful, are adopting it at the first author's university as a recommended style for masters theses and doctoral dissertations, and hereby encourage other institutions to emulate this innovation, especially for extended works such as monographs and books.

### References sorted by name

- [Daly, 2007: 3] Daly, P. W. (2007). Customizing bibliographic style files. <http://mirror.ctan.org/macros/latex/contrib/custom-bib/makebst.pdf>. 341
- [Daly, 2011: 1] Daly, P. W. (2011). A Master Bibliographic Style File for numerical, author-year, multilingual applications. <http://mirror.ctan.org/macros/latex/contrib/custom-bib/merlin.pdf>, v. 4.33. 340, 341
- [Kopka and Daly, 2003: 5] Kopka, H. and Daly, P. W. (2003). *Guide to L<sup>A</sup>T<sub>E</sub>X*. Addison-Wesley Professional, 4th edition. 341
- [Markey, 2009: 6] Markey, N. (2009). Tame the BeaST: The B to X of BIBTeX. [http://mirror.ctan.org/bibtex/tamethebeast/ttb\\_en.pdf](http://mirror.ctan.org/bibtex/tamethebeast/ttb_en.pdf), v. 1.4. 341
- [Mori, 2009: 2] Mori, L. F. (2009). Managing bibliographies with L<sup>A</sup>T<sub>E</sub>X. *TUGboat*, 30(1):36–48. <http://tug.org/TUGboat/tb30-1/tb94mori.pdf>. 340
- [Patashnik, 1998: 4] Patashnik, O. (1998). BIBTEXing. <http://mirror.ctan.org/biblio/bibtex/contrib/doc/btxdoc.pdf>. 341
- [6: Markey, 2009] Nicolas Markey. Tame the BeaST: The B to X of BIBTeX, October 2009. [http://mirror.ctan.org/bibtex/tamethebeast/ttb\\_en.pdf](http://mirror.ctan.org/bibtex/tamethebeast/ttb_en.pdf), v. 1.4. 341

### References sorted by year

- [Patashnik, 1998: 4] Patashnik, O. BIBTEXing. 1998. <http://mirror.ctan.org/biblio/bibtex/contrib/doc/btxdoc.pdf>. 341
- [Kopka and Daly, 2003: 5] Kopka, H. and Daly, P. W. *Guide to L<sup>A</sup>T<sub>E</sub>X*. Addison-Wesley Professional, 4th edition, 2003. ISBN 0-321-17385-6. 341
- [Daly, 2007: 3] Daly, P. W. Customizing bibliographic style files. 2007. <http://mirror.ctan.org/macros/latex/contrib/custom-bib/makebst.pdf>. 341
- [Markey, 2009: 6] Markey, N. Tame the BeaST: The B to X of BIBTeX. 2009. [http://mirror.ctan.org/bibtex/tamethebeast/ttb\\_en.pdf](http://mirror.ctan.org/bibtex/tamethebeast/ttb_en.pdf), v. 1.4. 341
- [Mori, 2009: 2] Mori, L. F. Managing bibliographies with L<sup>A</sup>T<sub>E</sub>X. *TUGboat*, 30(1):36–48. <http://tug.org/TUGboat/tb30-1/tb94mori.pdf>. 340
- [Daly, 2011: 1] Daly, P. W. A Master Bibliographic Style File for numerical, author-year, multilingual applications. 2011. <http://mirror.ctan.org/macros/latex/contrib/custom-bib/merlin.pdf>, v. 4.33. 340, 341

### References sorted by appearance

- [1: Daly, 2011] Patrick W. Daly. A Master Bibliographic Style File for numerical, author-year, multilingual applications, October 2011. <http://mirror.ctan.org/macros/latex/contrib/custom-bib/merlin.pdf>, v. 4.33. 340, 341
- [2: Mori, 2009] Lapo F. Mori. Managing bibliographies with L<sup>A</sup>T<sub>E</sub>X. *TUGboat*, 30(1):36–48. <http://tug.org/TUGboat/tb30-1/tb94mori.pdf>. 340
- [3: Daly, 2007] Patrick W. Daly. Customizing bibliographic style files, 2007. <http://mirror.ctan.org/macros/latex/contrib/custom-bib/makebst.pdf>. 341
- [4: Patashnik, 1998] Oren Patashnik. BIBTEXing, 1998. <http://mirror.ctan.org/biblio/bibtex/contrib/doc/btxdoc.pdf>. 341
- [5: Kopka and Daly, 2003] Helmut Kopka and Patrick W. Daly. *Guide to L<sup>A</sup>T<sub>E</sub>X*. Addison-Wesley Professional, 4th edition, 2003. 341

- ◊ Michael Cohen  
Spatial Media Group, Computer Arts Lab.  
University of Aizu  
Aizu-Wakamatsu, Fukushima 965-8580  
Japan  
mcohen (at) u-aizu dot ac dot jp  
[www.u-aizu.ac.jp/~mcohen](http://www.u-aizu.ac.jp/~mcohen)
- ◊ Yannis Haralambous  
Département Informatique Télécom Bretagne  
Technopôle de Brest Iroise, CS 83818  
29238 Brest Cedex 3 France  
yannis.haralambous (at) telecom-bretagne  
dot eu  
[international.telecom-bretagne.eu/  
welcome/studies/msc/professors/  
haralambous.php](http://international.telecom-bretagne.eu/welcome/studies/msc/professors/haralambous.php)
- ◊ Boris Veytsman  
Systems Biology School and Computational  
Materials Science Center  
MS 6A2  
George Mason University  
Fairfax, VA 22030 USA  
borisv (at) lk dot net  
[borisv.lk.net](http://borisv.lk.net)

## **L<sup>A</sup>T<sub>E</sub>X and graphics: Basics and packages**

Aleksandra Hankus and Zofia Walczak

### **1 Introduction**

There are the number of distinct ways of producing graphics, each with advantages and disadvantages in terms of flexibility, device independence and ability to include arbitrary T<sub>E</sub>X text. In this paper we will discuss two ways of placing graphics inside a L<sup>A</sup>T<sub>E</sub>X document. The first is about graphics imported to the T<sub>E</sub>X file from an external graphic program, and the second about creating graphics inside a T<sub>E</sub>X document. We will discuss documents with graphics which are intended to be printed, and presentations made with the `beamer` class.

### **2 Importing graphics into L<sup>A</sup>T<sub>E</sub>X**

When we want to include graphics in a document we have to take into account that L<sup>A</sup>T<sub>E</sub>X cannot manage pictures directly. L<sup>A</sup>T<sub>E</sub>X just creates a box with the desired size for the image we want to include and embeds a reference to the picture, without any other processing. This means we have to take care of the format and size of the images to be included. This is not such a hard task because L<sup>A</sup>T<sub>E</sub>X supports the most common picture formats around.

#### **2.1 The `graphicx` package**

Since L<sup>A</sup>T<sub>E</sub>X can't manage pictures directly, we load the `graphicx` package for help by placing the following in the preamble of our document:

```
\usepackage{graphicx}
```

The image formats we can use depend on the driver that `graphicx` is using, and since the driver is automatically chosen according to the compiler (T<sub>E</sub>X variant) being used, in practice the allowed image formats depend on the compiler.

The only format you can include while compiling with `latex` is Encapsulated PostScript (EPS). An EPS file declares the size of the image, which makes it easy for L<sup>A</sup>T<sub>E</sub>X to arrange the text and the graphics in the best way. EPS is (typically) a vector format, meaning that it can have very high quality if it is created properly, namely with programs that are able to manage vector graphics.

If we are compiling with `pdflatex` to produce a PDF, we have a wider choice. We can insert graphics in JPG, PNG, PDF. EPS format can also be used if it is converted to PDF; in current distributions, that happens automatically with the help of `epstopdf`.

The same L<sup>A</sup>T<sub>E</sub>X source can be compiled in both `latex` and `pdflatex` without any change, as long as we avoid using particular packages. We can use both

compilers for documents with pictures as well, if we remember to provide the pictures in proper format (both EPS and one of JPG, PNG or PDF).

#### **2.2 Including graphics**

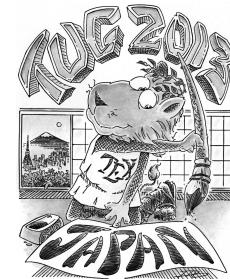
After we have loaded the `graphicx` package in the preamble, we can include images using the command `\includegraphics`, whose syntax is the following:

```
\includegraphics[arg1,arg2,...,argn]{imgname}
```

The arguments in square brackets are optional, whereas arguments in curly braces are compulsory.

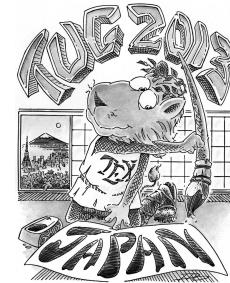
#### **2.3 Examples**

For scaling images we can use the optional argument `scale=<number>`:

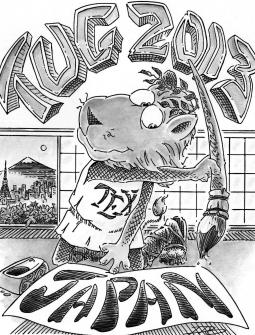


```
\includegraphics[scale=.16]{name}
```

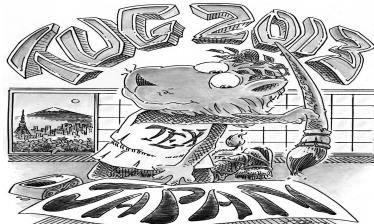
We can give image dimensions with either or both of the optional arguments `width=<number>` and `height=<number>`. When we specify only one or the other, the second will be chosen proportionally. When we specify both, the image will be resized without preserving proportions.



```
\includegraphics[width=3cm]{name}
```

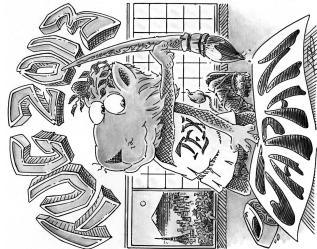


```
\includegraphics[height=4.5cm]{name}
```



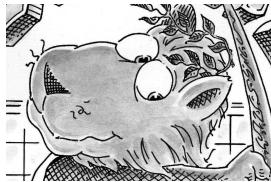
```
\includegraphics[width=5cm,height=3cm]{name}
```

To rotate the image, the option `angle=<number>` is used.



```
\includegraphics[scale=0.18,angle=90]{name}
```

And finally, here is an example of how to crop an image to focus on one particular area of interest. For this purpose we use the `trim` argument; in order, its parameters are `<left> <bottom> <right> <top>`.



```
\includegraphics[trim=.5cm 1.1cm .3cm .5cm,clip, width=3.5cm]{name}
```

We can specify which image file is to be preferred by `pdflatex` through this preamble command:

```
\DeclareGraphicsExtensions{.pdf,.png,.jpg}
```

This specifies the files to include in the document (in order of preference), if there exist files with the same name but with different extensions.

## 2.4 The figure environment

There are many situations where we want to add to the image a caption and possibly a cross-reference. We can do that with the `figure` environment, but we have to take into account that this is a so-called “floating” environment. The minimum required code is the following:

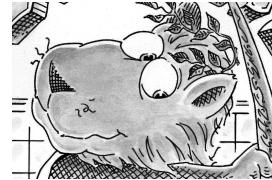
```
\begin{figure}[pos]
\includegraphics{image name.png}
\end{figure}
```

where the optional argument `[pos]` stands for the allowed positions of the figure on the page. Such a float placement specifier can consist of the following characters in any order: `htb!p`. For example, speci-

fying `[bt]` means that our picture can be placed on the bottom or top area of the page.

The above code is relatively trivial, and doesn’t offer much functionality. The next sample shows an extended use of the `figure` environment which is almost universally useful, offering a caption, label and centering the image, placed at either the current position (“here”) or the top of the page.

```
\begin{figure}[ht]
\centering
\includegraphics{name}
\caption{Caption}
\label{fig:1}
\end{figure}
```



**Figure 1:** Just the lion

## 3 Supporting the creation of graphics directly in L<sup>A</sup>T<sub>E</sub>X

There are many packages to do pictures in (L<sup>A</sup>)T<sub>E</sub>X itself rather than importing graphics created externally, starting with simple use of the L<sup>A</sup>T<sub>E</sub>X `picture` environment.

### 3.1 The picture environment

The `picture` environment is used to draw pictures composed of text, straight lines, arrows and circles. The objects in the picture are positioning by specifying their coordinates. The first picture environment was created by Leslie Lampert.

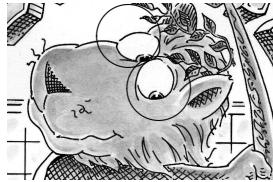
The basic syntax for the environment is:

```
\begin{picture}(width,height)(xoffset,yoffset)
  picture commands
\end{picture}
```

Thus, the `picture` environment has one mandatory argument, which specifies the size of the picture. The environment produces a rectangular box with width and height determined by the values of these two arguments. Coordinates are specified in the usual way with respect to an origin, which is normally at the lower-left corner of the picture. The optional positioning argument following the size argument can change the origin. If we decide to modify our picture by shifting everything, we can just add the appropriate optional argument.

Everything that appears in a picture is drawn by the `\put` command.

Using the `picture` environment we can “decorate” our previous image, for example to add to our image “glasses”.



**Figure 2:** Adding glasses

```
\setlength{\unitlength}{1cm}
\begin{picture}(4,2.5)
\includegraphics[width=3.5cm]{image name.jpg}
\put(-1.9,1.9){\circle{.8}}
\put(-1.5,1.2){\circle{.8}}
\end{picture}
```

### 3.2 XY-pic package

`XY-pic` is a special package for drawing diagrams. It works smoothly with most formats, including L<sup>A</sup>T<sub>E</sub>X, *AMSTeX*, *AMS-TeX*, and plain TeX. To use it, add the following line to the preamble of your document:

```
\usepackage[all]{xy}
```

where “all” means you want to load a large standard set of functions from `XY-pic`, suitable for developing complex diagrams. Below we show an example.

$$\begin{array}{ccc}
 G/T & \xrightarrow{\quad} & G/T \\
 \downarrow & & \downarrow \\
 E & \xrightarrow{\hat{f}} & BT \\
 \downarrow \pi & & \downarrow p \\
 S^{2k} & \xrightarrow{f} & BG
 \end{array}
 \quad \left\{
 \begin{array}{l}
 G/T \ar[r] = \ar[d] \\
 \& G/T \ar[d] \\
 E \ar[r]^{\hat{f}} & \ar[d] \\
 \& \ar[d]^{\pi} \\
 S^{2k} \ar[r]^f & BG
 \end{array}
 \right.$$

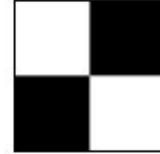
### 3.3 PSTricks—the pstricks package

Here, the basic package to use is `pstricks`, to be loaded with the usual command `\usepackage` in the document preamble. PSTricks commands are usually placed in a `pspicture` environment, whose mandatory argument gives the coordinates of the upper-right corner (the lower-left is the origin by default).

```
\begin{pspicture}(x1,y1)
  pstricks commands
\end{pspicture}
```

Here is a basic example. The `\psframe` command draws an unfilled rectangle, and its starred version makes it filled.

```
\begin{pspicture}(6,4)
\psframe(1,1)(3,3)
\psframe*(1,1)(2,2)
\psframe*(2,2)(3,3)
\end{pspicture}
```



### 3.4 PSTricks—the psfrag package

The `psfrag` package allows L<sup>A</sup>T<sub>E</sub>X users to replace text strings in EPS files created by external programs with L<sup>A</sup>T<sub>E</sub>X text or equations. To use `psfrag`, create an EPS file and then perform the following steps

- In the document, use the `\psfrag` command to specify the PostScript text in the EPS to be replaced, and the replacement L<sup>A</sup>T<sub>E</sub>X string. This makes the specified substitution occur in any subsequent `\includegraphics` command issued in the same environment.
- Use the `\includegraphics` command as usual.

The `\psfrag` command has the following syntax:

```
\psfrag{PStext}[posn][PSposn][scale][rot]
          [text]
```

**Remark:** PSfrag cannot be used with pdfL<sup>A</sup>T<sub>E</sub>X. If such substitution is needed, one option is to use the L<sup>A</sup>T<sub>E</sub>X-to-DVI-to-PostScript-to-PDF route that was used before pdfL<sup>A</sup>T<sub>E</sub>X. PSfrag also doesn't work with `beamer`. An alternative there is to use the TikZ package (with the EPS figure converted to PDF).

### 3.5 The amscd package

The `amscd` package provides a CD environment that emulates the commutative diagram capabilities of *AMSTeX* version 2.x. This means that only simple rectangular diagrams are supported, with no diagonal arrows or more exotic features.

```

 A_{PL}(Y) —————→ A_{PL}(X) —————→ A_{PL}(F)
      m_Y ↑           m ↑           m̄ ↑
      (ΛV_Y, d) —————→ (ΛV_Y ⊗ ΛV, d) —————→ (ΛV, d̄)
 
 $$ \backslash CD
 A_{\{PL\}}(Y) @>>> A_{\{PL\}}(X) \\
 @>>> A_{\{PL\}}(F) \\
 @A{m\_Y}AA @A{m}AA \\
 @A{\bar{m}}AA @A{\bar{m}}AA \\
 (@\Lambda V_Y, d) @>>> (@\Lambda V_Y \otimes @\Lambda V, d) \\
 @>>> (@\Lambda V, d̄) \\
 @. @. @. \\
 \end{CD}
 
```

**Remark:** The `amscd` package does not work with the `beamer` class.

### 3.6 MusiXTEX

MusiXTEX is a set of macros and fonts which enables music typesetting within the TeX system.

It contains symbols for staves, notes, chords, beams, slurs and ornaments, ready to be arranged to form a sheet of music.

But it must be told how to position those symbols on the page. This can be done manually, if you elect to proceed by entering MusiXTEX commands manually into an input file.

However most users will find it far less taxing to let such decisions be made largely by the preprocessor PMX, which also uses a much simpler input language than MusiXTEX. Here is an example of the output.



**Remark:** MusiXTEX needs L<sup>A</sup>T<sub>E</sub>X, which is automatically invoked when needed; but in general, L<sup>A</sup>T<sub>E</sub>X and MusiXTEX cannot be combined. For typesetting a large musical score it is better to use another alternative.

### 3.7 Graphics with PGF/TikZ

One possible solution for drawing graphics directly with TeX commands is PGF/TikZ. TikZ can produce portable graphics in both PDF and PostScript formats using either plain (pdf)TeX, (pdf)L<sup>A</sup>T<sub>E</sub>X or ConTeXt. It comes with very good documentation, and there is an extensive collection of examples at <http://www.texample.net/tikz>.

Using TikZ in a L<sup>A</sup>T<sub>E</sub>X document requires loading the `tikz` package

```
\usepackage{tikz}
```

somewhere in the preamble, as usual. This automatically loads the `pgf` package. To load further libraries use

```
\usetikzlibrary{list of libraries}
```

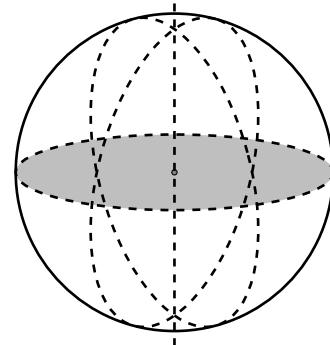
Some useful existing libraries are: `arrows`, `automata`, `backgrounds`, `calendar`, `chains`, `matrix`, `mindmap`, `patterns`, `petri`, `shadows`, `shapes.geometric`, and there are plenty more.

Drawing commands are usually enclosed in a `tikzpicture` environment:

```
\begin{tikzpicture}[options]
  tikz commands
\end{tikzpicture}
```

or alternatively we can use the `\tikz` command:

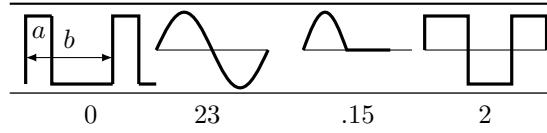
```
\tikz[options]{tikz commands}
```



If we specify the bounding box (it's an optional argument to the environment) with the `baseline` option as we show here:

```
\begin{tikzpicture}
  [x=0.0714\textwidth,y=0.5cm,
   baseline=(current bounding box.east)]
\path[use as bounding box](0,-1)rectangle(2,1);
\draw (0,0)--(2,0);
```

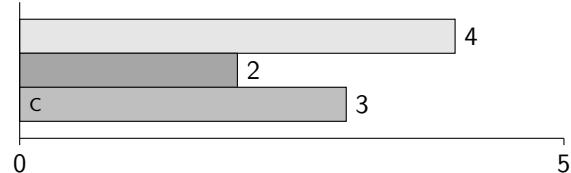
we can draw a table with different pictures in every cell in the row, all aligned together.



## 4 Some packages based on PGF/TikZ

### 4.1 The bchart package

`bchart` is a L<sup>A</sup>T<sub>E</sub>X package for drawing simple bar charts with horizontal bars on a numerical *x*-axis. It is based on the TikZ drawing package.

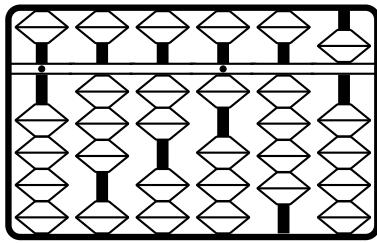


```
\begin{bchart}[max=5,scale=.9]
\bbar[color=gray!20]{4}
\bbar[color=gray!70]{2}
\bbar[text=\scriptsize{C},color=gray!50]{3}
\end{bchart}
```

**Remark:** The `bchar` package can be used with both `latex` and `pdflatex`, and it also works with the `beamer` class.

## 4.2 The pgf-soroban package

The soroban is an abacus developed in Japan; the `pgf-soroban` package lets us typeset representations of soroban values. We load the package in the usual way, with `\usepackage{pgf-soroban}` in the preamble. There is no need to load any corresponding graphics package, as all required packages are loaded by the soroban package. The package sets a base unit as 1 mm, as well as other lengths. If we want to change the size, the units can be changed with, e.g., `\ladj{0.25}`. The soroban picture below represents the number 321.45.



```
\ladj{0.5}
\begin{tikzpicture}
\tige{1}{0}{1} \tige{2}{3}{0}
\tige{3}{2}{0} \tige{4}{1}{1}
\tige{5}{4}{0} \tige{6}{5}{0}
\cadre{6}
\end{tikzpicture}
```

There is also a soroban package for PSTricks, named `pst-soroban`.

## References

- [1] D.P. Carlisle, Packages in the ‘graphics’ bundle, 2005, [ctan.org/pkg/grfguide](http://ctan.org/pkg/grfguide).
- [2] A. Delmotte, `pgf-soroban`—Create images of the soroban using TikZ/PGF, [ctan.org/pkg/pgf-soroban](http://ctan.org/pkg/pgf-soroban).
- [3] M. Goossens, F. Mittelbach, S. Rahtz, D. Roegel, H. Voß, *L<sup>A</sup>T<sub>E</sub>X Graphics Companion*, second edition, Addison-Wesley, 2007.
- [4] H. Kopka and P.W. Daly, *A Guide to L<sup>A</sup>T<sub>E</sub>X 2 <sub>$\epsilon$</sub>* , fourth edition, Addison-Wesley, 2003.
- [5] T. Kuhn, `bchart`: Simple bar charts in L<sup>A</sup>T<sub>E</sub>X, version 0.1.2, [ctan.org/pkg/bchart](http://ctan.org/pkg/bchart).
- [6] L. Lamport, *L<sup>A</sup>T<sub>E</sub>X: A Document Preparation System*, Addison-Wesley, second edition, 1994.
- [7] L. Lamport, *L<sup>A</sup>T<sub>E</sub>X: A Document Preparation System*, Wydawnictwa Naukowo-Techniczne, Warszawa, 2004 (in Polish).
- [8] E. Rafajłowicz and W. Myszka, *L<sup>A</sup>T<sub>E</sub>X, Zaawansowane Narzędzia*, Akad. Ofic. Wydawn. PLJ, Warszawa, 1996 (in Polish).
- [9] D. Taupin, `MusiXTEX`. Using T<sub>E</sub>X to write polyphonic or instrumental music, Version 1.15, [ctan.org/pkg/musixtex](http://ctan.org/pkg/musixtex).
- [10] Z. Walczak, *L<sup>A</sup>T<sub>E</sub>X for the Impatient*, Wydawn. Uniwersytetu Łódzkiego, 2012 (in Polish).

◊ Aleksandra Hankus  
Institute of Mathematics,  
University of Silesia, Poland  
`aleksandra.hankus (at) us dot edu dot pl`

◊ Zofia Walczak  
Faculty of Mathematics and  
Computer Science,  
University of Łódź, Poland  
`zofia.walczak (at) math dot uni dot lodz dot pl`

## Plots in L<sup>A</sup>T<sub>E</sub>X: Gnuplot, Octave, make

Boris Veytsman and Leyla Akhmadeeva

### Abstract

Making scientific and engineering documents with complex plots may be difficult and time-consuming. This is especially true if data updates require re-building of plots and documents. In this report a workflow based on an integration of (L)T<sub>E</sub>X, Gnuplot and Octave using Makefiles in a Unix environment is proposed and discussed in detail.

### 1 Introduction

Some time ago one of us (BV) worked on a report about aircraft navigation accuracy. This report included about forty charts of predicted errors as depending on the aircraft altitude, location of surveillance radios, etc. Then a coworker came to the office to tell me that some parameters of the model had changed and requested replotting all the charts. “How long would it take to do it?”, he asked with some trepidation, since the deadline was close. “Well”, was the answer, “I have a rather slow computer here. Probably about two to three minutes.” Having said this, the author changed several lines in one of the configuration files and typed `make`. In two minutes the machine happily produced an updated report.

This example illustrates a certain point about computers. They can take over the mind-numbing drudgery (like replotting dozens of charts) so we can do interesting things (like analyzing the message behind these charts). Unfortunately, for many people computers are just glorified typewriters/calculators/drawing devices, requiring constant hand holding and manual interventions. These users try to perform all the boring minute steps themselves, redoing them when anything changes. However, humans are not especially good at boring and repetitious work. They make mistakes. This often leads to embarrassing results (see, for example, the discussion of spreadsheet errors in Reinhart and Rogoff’s paper by Herndon, Ash, and Pollin, 2013). It is much more rewarding to teach a computer to do such work for you.

In this paper we show how to teach your computer to make high quality plots for your papers and reports, and to remake them as needed. This involves a combination of T<sub>E</sub>X, Gnuplot or Octave, and Makefiles. The system is highly customizable, and rather easy to use. Of course, T<sub>E</sub>X is the heart of the system, and it was developed with T<sub>E</sub>X in mind on each step. Thus we hope it might be of interest to the *TUGboat* readership.

Some points should be made before we discuss

this system. First, it was developed “in house”, and thus reflects certain tastes and idiosyncrasies. Second, we started to work on it long time ago—before such tools as `latexmk` or Asymptote were available. Thus it uses only classic tools and has a certain “retro” computing spirit. Third, it was developed for a Unix-like environment.<sup>1</sup> One can get it working on Windows using any free implementation of `make`, but that is beyond the scope of this paper.

### 2 Gnuplot graphics

There are many choices for a plotting program suitable for a T<sub>E</sub>X user: PSTRicks, PGF/TikZ, METAFONT, METAPOST, and Asymptote come to mind, as well as a plethora of non-free solutions. However for complex graphics, especially three-dimensional ones, Gnuplot is, in our opinion, among the best choices. It has the right combination of sound defaults (axis labeling, tick marks location, etc.) and the option of changing any default if needed. A full discussion of the rich possibilities of this program is also beyond our scope here. We recommend the extensive built-in help (try `help terminal epslate`, for example) and the book by Janert (2009).

How can we include the graphics produced by Gnuplot into a T<sub>E</sub>X document? The simplest solution is to make Gnuplot output an EPS or PDF file and use `\includegraphics` to put this plot into the proper place. However, this idea has a number of flaws. First, the text on the graphics will be done in Helvetica and Symbol fonts, which may well clash with your body font. Second, you may want to use T<sub>E</sub>X for annotations inside the graphics.

TikZ provides a method for smooth integration of Gnuplot-produced plots in the `\tikzpicture` environment. However, it tends to replot all graphics whenever you change the T<sub>E</sub>X file, which might be time consuming.

Gnuplot has a number of T<sub>E</sub>X-compatible output formats (“terminals” in Gnuplot terminology). Probably the best choice for complex graphics is `epslatex` (or `pstex` for plain T<sub>E</sub>X). These terminals produce a `.tex` file that has all the labels, while the graphics are saved in a PostScript file, which is automatically called by the `.tex` file. An example of the usage of this terminal is shown in Figure 1. Let us discuss it in detail.

The first line sets the output format:

```
set terminal epslate monochrome
```

The option `monochrome` is chosen here because the printer charges *TUGboat* extra for color pages. In most cases the `color` option is preferable. Note: even

<sup>1</sup> Including GNU/Linux and Mac OS X.

```

set terminal epslatex monochrome
set output "function-fig.tex"
set pm3d          # Colored surface
unset surface      # We do not want to plot the mesh lines
set isosamples 100, 100 # Smooth surface
set ztics 0.2       # Increment for z tick marks
set cbtics 0.2       # Increment for colored box
set format '$%g$'
set xtics offset 0,-.3
set ytics offset 1,0
set ztics offset -1,0
set cbtics offset 1,0
set xrange [-1.5:1.5]
set yrange [-1.5:1.5]
set label 1 \
'${f(\\mathbf{x})}=\\exp\\left(-\\lvert\\mathbf{x}\\rvert^2\\right)$' \
at -1.5,-1,1
set label 2 \
'$\\displaystyle\\max_{\\mathbf{x} \\in \\mathbb{R}^2} f(\\mathbf{x})$' \
at 1,1,1.3
set arrow 1 from 1,1,1.3 to 0,0,1 front
splot exp(-x**2-y**2) title ""
set output

```

**Figure 1:** A Gnuplot script `function.gp` with `epslatex` output

with the `monochrome` option, the package `color` or `xcolor` must be called by your main `TeX` file.

The next line sets the name of the output `.tex` file; we chose `function-fig.tex`. (We explain this naming convention below.)

The lines `set pm3d` and `unset surface` explain how to plot the three-dimensional graphics: using color (well, shades of gray for our monochrome display) to show the height and not plotting the surface mesh lines.

The line `set isosamples` sets the number of points where the function is calculated. We use  $100 \times 100 = 10000$  points for a smooth plot.

The lines `set ztics` and `set cbtics` set the increment for the ticks on the  $z$  axis and the color box in the legend. We do not use similar commands for `xtics` and `ytics` since the defaults are good enough.

The line `set format '$%g$'` refers to the format of the tick marks. It makes the numbers to be typeset in the math mode. The default is `%g`, which should be familiar to those knowing C formatting commands. Thus normally Gnuplot typesets tick marks as text, so minus signs become dashes.

The next lines slightly move the tick numbers for the  $x$ ,  $y$ , and  $z$  axes and the color box. (Gnuplot is not completely aware of the font metrics for `TeX` fonts, so its position calculations are sometimes not good enough for the demanding eyes of *TUGboat* editors.)

The lines `set xrange` and `set yrange` set the  $x$

and  $y$  domains for the plotting: from  $-1.5$  to  $1.5$ . We do not use a similar `set zrange` command since the default (based on the maximal and minimal values of the function plotted) looks good.

The next lines are `label` commands. They have three arguments: label number (1 and 2 in our case), label text (enclosed within single quotes) and label position (`at` statements). The text in the labels is `TeX` code interpreted in the context of your main document. Thus you can put arbitrarily complex annotations on your graphics. Gnuplot provides some mechanisms for fine-tuning the label reference point; in most cases, however, you do not need to change the default.

We use an arrow from the label to the top of the plot. It is set by the `set arrow` command. Its arguments include arrow number, arrow start and arrow end. The last keyword, `front`, means that the arrow is plotted on the front layer of the picture, i.e., is not to be obscured by the plot itself.

The actual plot is done by the penultimate line:

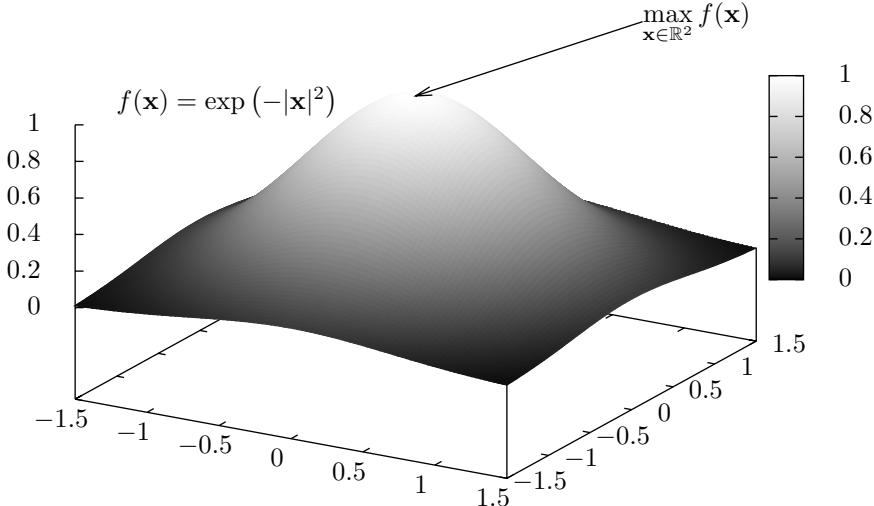
`splot exp(-x**2-y**2) title ""`

It means: do a surface plot of the function  $\exp(-x^2 - y^2)$ , with an empty title (by default, Gnuplot typesets the formula as the title).

The last line, `set output`, writes the results to the output files and closes them. See Figure 2.

## 2.1 Plots from data points

In the above example we plotted a mathematical



**Figure 2:** The plot generated by the script in Figure 1

expression. Gnuplot can also plot data from a file, obtained from an experiment or other calculations. As an example we show a script in Figure 3. Here we plot the stopping distances of cars moving with different speeds as measured in 1920s. The data is from the R distribution (R Core Team, 2013). We put them in a space-separated file `cars.dat`, which looks like this:

```
# "speed" "dist"
4 2
4 10
7 4
...
```

The first line shows the column names. It starts with the comment symbol `#` to tell Gnuplot not to try to plot it.

Most of the commands in Figure 3 are similar to those in Figure 1. Let us look at the ones that are different.

The line `set logscale xy` tells Gnuplot to create a log-log plot. The lines `set xlabel` and `set ylabel` set the labels for  $x$  and  $y$  axes correspondingly. The command `set label 1` contains a TeX expression that includes a mathematical formula and rotation to typeset the formula along the line it describes. All rotation and typesetting is done by TeX rather than by Gnuplot.

Since we make a two-dimensional plot rather than a three-dimensional one, we use the `plot` command rather than `splot`:

```
plot "cars.dat" with points pt 4 title "", \
exp(-0.73+1.6*log(x)) title ""
```

This command has two arguments separated by a comma and corresponding to two objects we want to plot: a data file, plotted with points of

```
set terminal epslatex monochrome
set output "cars-fig.tex"
set logscale xy
set xrange [1:100]
set yrange [1:500]
set xlabel 'Speed, mph'
set ylabel 'Stopping distance, feet'
set format '$%g$'
set label 1 \
    '\rotatebox{41}{$\ln y=-0.73+1.6\ln x$}' \
    at 1.8, 4
plot "cars.dat" with points pt 4 title "", \
    exp(-0.73+1.6*log(x)) title ""
set output
```

**Figure 3:** Another Gnuplot script, `cars.gp`

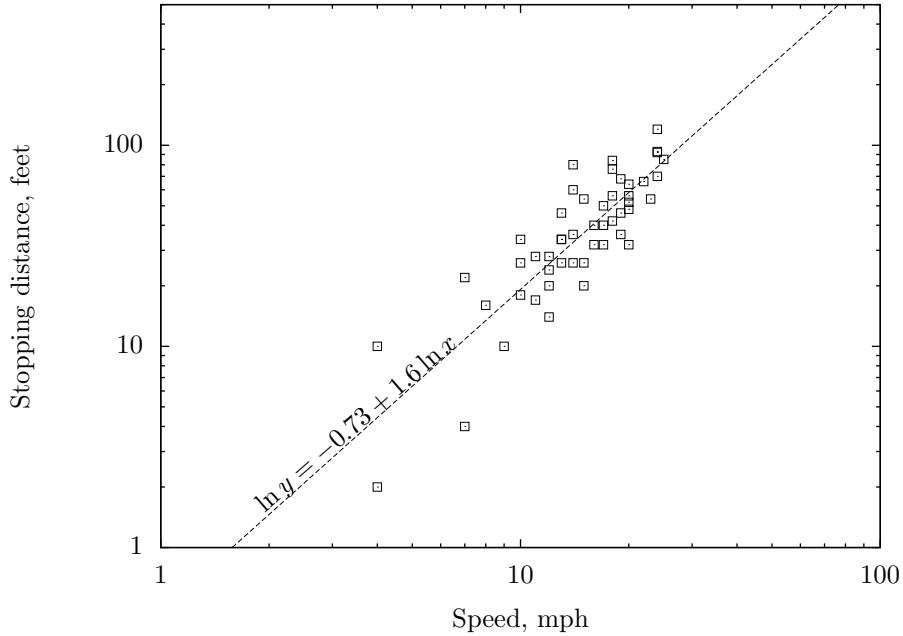
type 4 (these happen to be unfilled squares), and a mathematical expression corresponding to a straight line on the log-log scale. The result is shown in Figure 4.

A good way to debug and tune the graphics is to comment out the first two lines of the script and run it through Gnuplot, thus seeing the results online, changing the script until you get a satisfactory result.

### 3 Octave graphics

Gnuplot's built-in features cover most mathematical needs. However, sometimes they are not enough. What can we do then? As discussed in the previous section, we can calculate the data points in an external program and feed the result to Gnuplot as a (space separated) text file. Another possibility is to use software that can talk to Gnuplot directly.

A good choice for this is Octave. Octave is a high level language and program for numerical calculations. It is similar to and mostly compatible with



**Figure 4:** The plot generated by the `cars.gp` script in Figure 3

the commercial program MATLAB. As often happens with free software, some of Octave's capabilities surpass those of its commercial sibling. In particular, the technique of generating  $\text{\TeX}$ -compatible graphics described in this article *does not* work in MATLAB. The latter can produce plots in the EPS format (Octave can do this too), but text annotations on these plots are done in its own fonts, which may clash with the body text.

The current version of Octave uses Gnuplot for graphics, so most of the features discussed in the previous section are available in Octave. However, the syntax is slightly different. The most important difference is the order of commands. In Gnuplot we first “set” the annotations: legend, labels, etc., and then plot the data. In Octave we plot the data first, and only then add annotations to the existing figure.

In Figures 5 and 6 we show a plot used in one of our reports. In this report we discussed the behavior of a certain system. It depended on a dimensionless parameter  $\rho$ . The report showed that this parameter ought to satisfy the following condition:

$$\text{ber}_1 \rho = \text{bei}_1 \rho$$

where  $\text{ber}_1$  and  $\text{bei}_1$  are so-called Kelvin functions, related to the Bessel function of complex argument (Olver, Lozier, Boisvert, and Clark, 2010, § 10.61). Thus we wanted a plot of the expression

$$\delta(\rho) = \text{ber}_1 \rho - \text{bei}_1 \rho$$

and the point  $\rho_0$  for which  $\delta(\rho_0) = 0$ .

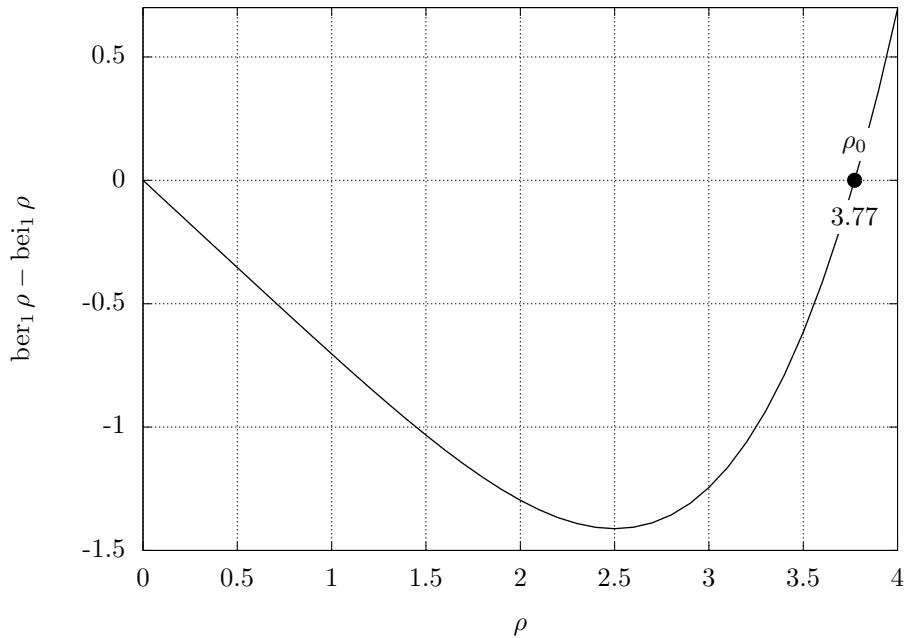
Unfortunately, Gnuplot does not know anything about Kelvin functions. An attempt to calculate them using Bessel functions of complex argument leads (at least in version 4.4) to the following truthful, but not especially helpful message: *can only do bessel functions of reals*. To tell the truth, Octave also knows nothing about Kelvin functions. However, unlike Gnuplot, it is not afraid of Bessel functions of complex argument.

In Figure 5 we show an Octave script `kelvin.m` that generates the required plot. Let us discuss the script in detail.

The first three lines define functions  $\text{ber}_1$ ,  $\text{bei}_1$  and  $\delta$  using Octave's built-in `besselj` command. The next line, `rho0 = fsolve(delta, 4)` calculates the root of equation  $\delta(x) = 0$  starting from the point  $x = 4$ , and assigns the result to the variable `rho0`. By the way, this root is  $\rho_0 = 3.7727$ .

The reason for the next command is the way Octave executes plot commands. In Gnuplot we first set up the “terminal”, and only then draw the plot. Thus, at the time of plotting our computer already knows we want to save the result to a file and does not make on-screen plots. In Octave we first draw the plot on the screen and only then “save” the result. This slows down the execution. The command `figure('visible', 'off')` switches off this on-screen drawing.

The next few lines perform the actual plotting. Octave has two ways to make a plot of a function. One is the `fplot` command: `fplot(delta, [0, 4])`



**Figure 6:** The plot generated by the `kelvin.m` script in Figure 5

```

ber1 = @(x) -real(besselj(1,x*exp(pi*1i/4)));
bei1 = @(x) imag(besselj(1,x*exp(1i*pi/4)));
delta = @(x) ber1(x)-bei1(x);
rho0 = fsolve(delta,4);
figure('visible','off');
x = 0:0.1:4;
plot(x, delta(x), 'linewidth', 2);
hold on;
plot([rho0], [0], 'o', 'linewidth', 10);
text(rho0, 0.15, \
    '\colorbox{white}{$\rho_0$}', \
    'horizontalalignment', 'center');
text(rho0, -0.15, \
    sprintf("\colorbox{white}{$%.2f$}", rho0), \
    'horizontalalignment', 'center');
title("");
legend("off");
grid();
xlabel('$\rho$');
ylabel('$\ber_1\rho - \bei_1\rho$');
print -depslatex -mono "-S800,600" \
    "kelvin-fig.tex"

```

**Figure 5:** An Octave script, `kelvin.m`

would make the graph shown in Figure 6. Unfortunately, in the current version of Octave the command `fplot` has rather limited options for fine control of the plot; in particular, it does not allow to change the default line width. The default lines look weak in our monochrome version. So here we use the generic `plot` command instead of `fplot`. We gen-

erate an array of abscissas with the command `x = 0:0.1:4`, and then plot `delta(x)` versus `x`. The command `plot(x, delta(x), 'linewidth', 2)` generates the plot with a line width of 2, meaning twice the default width.

Normally a `plot` or `fplot` command erases previous graphics and starts afresh. The line `hold on` preserves the graphic and combines it with the next plot. This next plot consists of only one point, a large dot at the zero of the function  $\delta(x)$ : `plot([rho0], [0], 'o', 'linewidth', 10)`; The parameter `'o'` sets the shape of the marker, while setting the line width to 10 makes it large.

The next two commands, using `text(...)`, are similar to Gnuplot `label` commands. We set up the coordinates of the text and the text itself. Additional parameters help to tune the output.

The first command places the “text” string `\colorbox{white}{$\rho_0$}` at the point  $(\rho_0, 0.15)$ , i.e. above the root of our equation. The text is centered on the reference point (the other options for `horizontalalignment` are `left` and `right`). This “text” is a L<sup>A</sup>T<sub>E</sub>X command that creates the symbol  $\rho_0$  inside a rectangle with white background (`colorbox`). We use this rectangle because otherwise the plot overlaps the symbol.

The second command is more complicated. Here the “text” is dynamically constructed by Octave itself. The function `sprintf` is similar to the function

of the same name in C, awk, perl, and many other languages. Its first argument is the “format”, and the remaining arguments are interpreted according to this format. In our case we output a string which includes  $\rho_0$  typeset with two decimal figures per the specification `%.2f` in the format. The function returns the string<sup>2</sup> `\colorbox{white}{\$3.77\$}`, which is typeset centered at the point  $(\rho_0, -0.15)$ , i.e. under the root of the equation. We again use `\colorbox` to create the white background for the label.

The next commands are analogous to those in Gnuplot: we switch off the main title and legend, switch on the grid and set up the axes labels. Since `\ber` and `\bei` are *not* standard L<sup>A</sup>T<sub>E</sub>X operators, our `.tex` file has the following definitions based on the macro `\DeclareMathOperator` provided by the `amsmath` package:

```
\DeclareMathOperator{\ber}{ber}
\DeclareMathOperator{\bei}{bei}
```

The last line,

```
print -depslatex -mono "-S800,600" \
  "kelvin-fig.tex"
```

saves the graphs into the file `kelvin-fig.m`. It uses `epslatex` format (similar to Gnuplot’s `epslatex` terminal), and mono-chrome rendering. The magic string `"-S800,600"` sets the size of the figure in points (the reason why it must be in quotes is better known to the authors of Octave).

Like Gnuplot, Octave can create complex three-dimensional graphics, which we leave as an exercise to the reader.

Unfortunately, we failed to find the analog of the Gnuplot `set format` command, so tick marks on Figure 6 are typeset in text mode: compare ‘-1’ (wrong!) and ‘-1’ (right). This problem can be corrected by a simple `sed` script acting on the file `kelvin-fig.m`, which we leave as another exercise to the reader.

#### 4 Insertion of graphics in the `.tex` file

The methods described in the previous sections produce two files for each graphics: a `.tex` file with the textual material, and a PostScript file (either `.eps` or `.ps`) with the graphics material. For example, in the directory with this paper are the following files:

```
cars-fig.eps      cars-fig.tex
function-fig.eps  function-fig.tex
kelvin-fig.eps    kelvin-fig.tex
```

To use these graphics in the text, we “read in” the `.tex` file using the command `\input`, for example, `\input{function-fig}`. The associated PostScript

---

<sup>2</sup> Exercise: why does the format use double backslash?

file is automatically inserted by the `.tex` file with the corresponding `\includegraphics` command.

This works fine with the traditional `latex` with `dvips` route. What happens if you use `pdflatex`? Fortunately, recent distributions are smart enough to automatically convert `.eps` files to `.pdf` (using `epstopdf`), so after a run of `pdflatex` you can find in the working directory the files

```
cars-fig-eps-converted-to.pdf
function-fig-eps-converted-to.pdf
kelvin-fig-eps-converted-to.pdf
```

This conversion is done transparently to the user.<sup>3</sup> Of course, the PostScript files must have a correct bounding box for correct results.

#### 5 Putting everything together: Makefiles

The process described in the previous sections may seem rather complex. We run Gnuplot and/or Octave, `latex`, `dvips`, `ps2pdf`, `pdflatex`, ... If we change some of the files, we need to rerun the necessary portions of the process. A human should not do this manually (and probably cannot do it without introducing errors).

The famous utility `make` can do this for you.

Let us recall the basics. The utility reads a *Makefile* which sets up rules and dependencies. Rules tell it how to “make” a certain file: you run `latex` on a `.tex` file to generate a `.dvi` file, you run `dvips` on a `.dvi` file to generate a `.ps` file, etc. Dependencies record that a file A *depends* on the file B: whenever B is changed, A must be regenerated. You can find more information, for example, in the classic book by Mecklenburg (2004).

In this section we set up a typical Makefile for T<sub>E</sub>X and Gnuplot or Octave.

Let us start with Gnuplot. We will use the extension `.gp` for our Gnuplot scripts and the following naming convention: a file `file.gp` generates the files `file-fig.tex` and `file-fig.eps`. The part `-fig` is used to set up the `clean` task: to clean the directory we delete all generated files.

So, we can define the following simple rule: each `file-fig.tex` file depends on the `file.gp`, and `gnuplot` is used to generate it:

```
%-fig.tex: %.gp
  gnuplot $<
```

Here, according to Makefile syntax, `%` is a “wildcard”, and `$<` means the “prerequisite” (the `.gp` file).

Here is a similar rule for Octave-generated files:

```
%-fig.tex: %.m
  octave $<
```

---

<sup>3</sup> Unfortunately, Gnuplot’s `pstex` terminal for plain T<sub>E</sub>X uses PostScript specials instead of `\includegraphics`. This makes the technique described here inapplicable.

So now we have two rules for generation of *<file>-fig.tex* files: either from Gnuplot or from Octave. Happily, `make` is smart enough to choose the right one: if it finds an appropriate file ending in *.gp*, it uses the first rule, and if it finds an appropriate file ending in *.m*, it uses the second.<sup>4</sup>

Let us now discuss the generation of PDF files from *.tex* sources. In this article we discuss the `pdflatex` route; the rules for the “traditional” `latex → dvips` route are left as another exercise.

The basic idea is relatively simple: run `pdflatex` until the labels converge. The code below has an additional quirk of running `bibtex` several times because citations may use `crossref` fields:

```
% .pdf: %.tex
    pdflatex $*
    - bibtex $*
    pdflatex $*
    - while ( grep -q \
'`^LaTeX Warning: Label(s) may have changed' \
$*.log ) do (bibtex $*; pdflatex $*;) done
```

Of course, this is not the full story. We need to tell `make` that whenever a plot is changed, all PDF files must be regenerated. This can be done by adding to the Makefile lines like

```
document.pdf: plot-fig.tex
```

for each `\input{plot-fig}` line.

This line tells `make` to regenerate the main PDF (using the rule above) when `plot-fig.tex` changes; the companion `plot-fig.eps` could be added as another dependency, but since the two `plot-fig.*` files are always created simultaneously, it’s not necessary.

What about the conversion `.eps → .pdf`? Will that be done as well? The answer is yes. `TeX` uses a simple but sufficient algorithm for this conversion: whenever `.eps` file is newer than the generated `.pdf` file, the latter is regenerated. Thus, after you change `plot.gp`, one line in the Makefile triggers a number of events:

1. The program `make` finds the new `plot.gp` and calls Gnuplot to regenerate `plot-fig.tex`.
2. As a side effect the file `plot-fig.eps` is recreated by Gnuplot.
3. `TeX` finds the new `plot-fig.eps` and generates a new `plot-fig-eps-converted-to.pdf`.
4. The new version of the main PDF file is created.

If you have many plots, you might find it cumbersome to add a dependency for each. Fortunately, Makefiles can include subfiles, allowing us to automatically generate the dependencies. Each line will

---

<sup>4</sup> If both Gnuplot and Octave files are present, `make` chooses the rule that appears first in the Makefile.

```
#!/usr/bin/env perl
# Extract information from input statements
# in TeX files. Usage:
# makefigdepend FILE FILE FILE ... > depend

foreach my $file (@ARGV) {
    open (FILE, $file) || die "open($file): $!";
    $file =~ s/\.tex$/.pdf/;
    while (<FILE>) {
        while (/\\input(?:\\[[^\\]]+\\])*\\{\\{[^\\}]+\\}\\}/g) {
            print "$file: $1.tex\n";
        }
    }
    close FILE;
}
exit 0;
```

Figure 7: A Perl script for generation of dependencies

have the form `A: B`, showing that file `A` depends on file `B`. The way to do this is the following:

1. Add to the Makefile the line `-include depend`, which instructs `make` to read the file `depend` if it exists. The dash at the beginning tells `make` not to worry if this file is not found (e.g. at the start of the run).
2. Add to the Makefile the rules to generate the file `depend` from the sources.

For the second task we need to write a program to generate the dependency file. A simple Perl script `makefigdepend.pl` to do this is shown in Figure 7. (The choice of Perl makes our solution less “retro” than it could be: `sed` and `awk` could do the job.) Then the rule

```
depend: ${TEXFILES}
    perl makefigdepend.pl \
    ${TEXFILES} >depend
```

generates the required file. For example, the file `depend` for this article is the following.<sup>5</sup>

```
gnuplotmk.pdf: function-fig.tex
gnuplotmk.pdf: cars-fig.tex
gnuplotmk.pdf: kelvin-fig.tex
gnuplotmk.pdf: function-fig.tex
```

When we plot a data file, we want `make` to redo the plot not only when the `.gp` file is new, but also if the original data change. To this end, we add a line to the Makefile:

```
cars-fig.tex: cars.dat
```

An astute reader can see that we wrote this dependency manually. It is of course possible to write a

---

<sup>5</sup> An exercise: why is an identical line about `function-fig.tex` repeated in the generated file?

```

TEXFILES = gnuplotmk.tex
PDFS = ${TEXFILES:%.tex=%.pdf}

all: ${PDFS}

%.pdf: %.tex
    pdflatex $*
    - bibtex $*
    pdflatex $*
    - while ( grep -q \
' ^LaTeX Warning: Label(s) may have changed' \
        *.log ) \
    do (bibtex $*; pdflatex $*;) done

%-fig.tex: %.gp
    gnuplot $<
%-fig.tex: %.m
    octave $<

cars-fig.tex: cars.dat

clean:
    $(RM) *.aux *.bb1 *.dvi *.log \
    *.out *.toc *.blg *.lof *.lot \
    *.eps *-fig* depend
distclean: clean
    $(RM) ${PDFS}

depend: ${TEXFILES}
    perl makefigdepend.pl \
    ${TEXFILES} > depend
-include depend

```

**Figure 8:** Makefile for this paper

script to parse the Gnuplot files and put such lines in the file `depend`. Again, this is left as an exercise.

If the data files themselves are generated by another program, as would be typical, we can tell `make` to run this program if necessary by adding the dependencies of data files upon the necessary input parameters. This leads to incredibly smart behavior: as soon as any of the configuration or data files change, `make` regenerates all pieces that could be influenced by this change.

The last task is *cleaning*. A common convention is to provide two targets: `clean` removes all generated files except the principal (PDF) output, while `distclean` or `veryclean` deletes everything but the original sources:

```

clean:
    $(RM) *.aux *.bb1 *.dvi *.log \
    *.out *.toc *.blg *.lof *.lot \
    *.eps *-pics.* *-fig* depend
distclean: clean
    $(RM) ${PDFS}

```

In Figure 8 we show the Makefile for this paper.

## Acknowledgements

We are grateful to Marcel Richter who urged us to put together the notes in a readable form, and to Michael Kotelyanskii who made many useful comments about the manuscript.

## References

- Herndon, Thomas, M. Ash, and R. Pollin. “Does High Public Debt Consistently Stifle Economic Growth? A Critique of Reinhart and Rogoff”. Working Paper 322, University of Massachusetts, Amherst. Political Economy Research Institute, 2013. [http://www.peri.umass.edu/fileadmin/pdf/working\\_papers/working\\_papers\\_301-350/WP322.pdf](http://www.peri.umass.edu/fileadmin/pdf/working_papers/working_papers_301-350/WP322.pdf).
- Janert, Philipp K. *Gnuplot in Action. Understanding Data with Graphs*. Manning Publications Co., 2009.
- Mecklenburg, Robert. *Managing Projects with GNU Make*. O'Reilly Media Inc., Sebastopol, CA, third edition, 2004. Available at <http://oreilly.com/catalog/make3/book/index.csp>.
- Olver, F. W. J., D. W. Lozier, R. F. Boisvert, and C. W. Clark, editors. *NIST Handbook of Mathematical Functions*. Cambridge University Press, New York, NY, 2010.
- R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.

- ◊ Boris Veytsman  
School of Systems Biology &  
Computational Materials  
Science Center, MS 6A2  
George Mason University  
Fairfax, VA 22030  
`borisv (at) 1k dot net`
- ◊ Leyla Akhmadeeva  
Bashkir State Medical University  
3 Lenina Str., Ufa, 450000, Russia  
`la (at) ufanuro dot org`  
<http://www.ufanuro.org>

### Entry-level MetaPost 3: Color

Mari Voipio

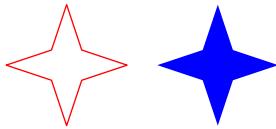
This installment of the entry-level MetaPost series discusses color; previous topics include the basic grid (*TUGboat* 34:1) and image transformations (34:2). For basic information on running MetaPost, either standalone or within a ConTeXt document, see <http://tug.org/metapost/runningmp.html>.

As we have seen in the earlier tutorials, MetaPost recognizes the colors *Red*, *Green* and *Blue* (plus black and white). We have also used color for both outlines and fills. Here's an example star:

```
numeric u; u := 2mm; % defining the unit
path star; % defining a star shape
star := (3u,3u) -- (0u,4u) -- (3u,5u)
    -- (4u,8u) -- (5u,5u) -- (8u,4u)
    -- (5u,3u) -- (4u,0u) -- cycle;

% outlined red star:
draw star withcolor red;

% filled blue star, a bit to the right:
fill star shifted (10u,0u) withcolor blue;
```



These three basic colors can be modified by “removing” some color, thus letting more black in, or by mixing in another color, e.g. white. The amount of color is given by adding a number between 0 and 1 to the color name. I think about these as percentages; I read *.25red* as “take 25% red and the rest black”.

On the other hand, we can add in a color other than black. For instance, *.25[red,white]* gives a color that is 75% red and 25% white—a quarter “of the way” between red and white, thus closer to red.

Here are some examples:

```
numeric u; u := 5mm;
% all black:
fill unitsquare scaled 1u withcolor 0red;

fill unitsquare scaled 1u shifted (1u,0)
    withcolor .25red;
fill unitsquare scaled 1u shifted (2u,0)
    withcolor .5red;
fill unitsquare scaled 1u shifted (3u,0)
    withcolor .75red;
fill unitsquare scaled 1u shifted (4u,0)
    withcolor red; % basic red
fill unitsquare scaled 1u shifted (5u,0)
    withcolor .25[red,white];
fill unitsquare scaled 1u shifted (6u,0)
    withcolor .5[red,white];
```

```
fill unitsquare scaled 1u shifted (7u,0)
    withcolor .75[red,white];
```

```
% all white (not visible):
fill unitsquare scaled 1u shifted (8u,0)
    withcolor 1[red,white];
```



(Only eight squares are evident, although nine were specified, because a white square on white paper, or a white background, is “invisible”.)

Once we have found a pleasing shade, we can name it in the same way as a path:

```
% defining a color:
color lightpink; lightpink := .8[red,white];

% a light pink star:
fill star withcolor lightpink;
```



However, this *is* a star and those are conventionally drawn in yellow. Not red, not green, not blue—yellow. How to achieve that?

The answer lies in RGB values, RGB being the color scheme used by MetaPost. Each RGB color is expressed with three values in the range 0–255, the numbers telling what proportions of red, green and blue to mix to achieve a certain color. E.g., the RGB code for the color “gold” is 255–215–0. To make MetaPost understand the code, the color proportions are given as fractions:

```
% fill star with gold (255-215-0):
fill star withcolor (255/255, 215/255, 0/255);
```



RGB charts are widely available online, so we don't have to guess the color codes (although that can be quite fun). I use “The Other RGB Chart” (<http://www.tayloredmktg.com/rgb>) because it is organized by color and includes color names that I like to use in my code.

```
numeric u; u := 10mm;
% color "dark orchid", 153-50-204
color darkorchid;
darkorchid := (153/255, 50/255, 204/255);
% fill a square:
fill unitsquare scaled 1u withcolor darkorchid;
```



The RGB color scheme is primarily for screens, so what you see is not necessarily exactly what prints. Screens are different, too, so what you see might not be the same as what I see. However, considering that MetaPost is mostly used to draw schematic graphics, the RGB color scheme is adequate for the job.

### Bonus: Shading

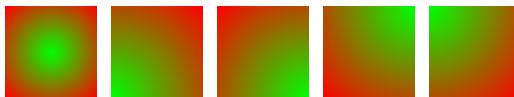
If you use ConTeXt, you can easily use shading from one color to another. The feature utilizes both Metafun and some advanced PDF trickery and thus is only available in ConTeXt. However, there is a cheat that gives MetaPost users limited access to a similar feature, see below.

**Shading in ConTeXt.** Shading can be either *circular* or *linear*. Circular shading can be either centered (option number 0, the default) or off-centered to one of the corners (options 1–4). Linear shading can be either diagonal (options 1–4) or vertical (options 5 and 7) or horizontal (options 6 and 8). Linear option 0 defaults to option 5, left-to-right shading.

For shading we have to specify what is shaded, how it is shaded (the option number) and the colors to shade from and to. Here are examples of all the circular shading options:

```
numeric u; u := 12mm;
path sq; sq := unitsquare scaled 1u;

% Circular shading (Metafun):
circular_shade (sq, 0, green, red);
circular_shade (sq shifted ((1u+2mm),0), 1,
                 green, red);
circular_shade (sq shifted ((2u+2*2mm),0), 2,
                 green, red);
circular_shade (sq shifted ((3u+3*2mm),0), 3,
                 green, red);
circular_shade (sq shifted ((4u+4*2mm),0), 4,
                 green, red);
```



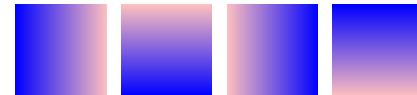
Diagonal linear shading (1–4) examples:

```
% Diagonal linear shading:
linear_shade (sq, 1, blue, .75[red,white]);
linear_shade (sq shifted ((1u+2mm),0), 2,
              blue, .75[red,white]);
linear_shade (sq shifted ((2u+2*2mm),0), 3,
              blue, .75[red,white]);
linear_shade (sq shifted ((3u+3*2mm),0), 4,
              blue, .75[red,white]);
```



Vertical/horizontal linear shading (5–8) examples:

```
% Vertical/horizontal linear shading:
linear_shade (sq, 5, blue, .75[red,white]);
linear_shade (sq shifted ((1u+2mm),0), 6,
              blue, .75[red,white]);
linear_shade (sq shifted ((2u+2*2mm),0), 7,
              blue, .75[red,white]);
linear_shade (sq shifted ((3u+3*2mm),0), 8,
              blue, .75[red,white]);
```



One of my first MetaPost exercises was a graphic that imitates a line drawn with pen and ink; I needed such a line to create a document that looks like a 17<sup>th</sup> century manuscript. As I use ConTeXt (which incorporates Metafun), I didn't realize at the time that I was using a very special feature—I was happy enough to have finally found something that did the job (although getting the line *where* I wanted it wasn't that easy...).

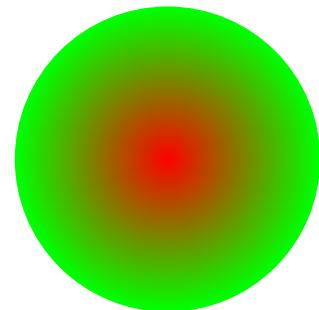
```
path cline; % defining the line
cline := (2mm,0) -- (70mm,0) -- (68mm,1mm)
-- (0,2mm) -- cycle;

% filling the line with shading:
linear_shade(cline, 0,
              black, 0.7[black,white]);
```

---

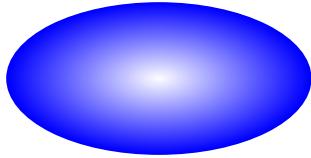
**Shading in plain MetaPost.** In plain MetaPost we can achieve a shaded circle by drawing multiple concentric rings, changing color at each step. For this we use a loop. Here is a complete example:

```
outputtemplate := "%j-%c.mps";
beginfig (1);
for i = 1 step 1 until 100 :
  a := i/100;
  draw fullcircle scaled (i*.4mm)
    withcolor a[red,green]
    withpen pencircle scaled .4mm;
endfor;
endfig;
end.
```



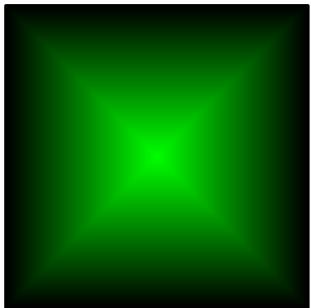
With Metafun, we can add `xscaled` or `yscaled` to create a shaded ellipse:

```
for i = 1 step 1 until 100 :
  a := i/100;
  draw fullcircle scaled (i*.2mm)
    withcolor a[white,blue]
    withpen pencircle scaled .2mm;
endfor;
currentpicture := currentpicture xscaled 2;
```



Also with Metafun, a shaded square/rectangle is possible:

```
for i = 1 step 1 until 100 :
  a := i/100;
  draw fullsquare scaled (i*.4mm)
    withcolor a[green,black]
    withpen pencircle scaled .4mm;
endfor;
```



This approach only works with PostScript output (`.mps/.eps/.ps`), not with SVG output.

### Filling and drawing at the same time

We can also both draw and fill a shape. However, we need to fill first and then draw. It is a bit counterintuitive at first; I think of this as coloring the shape, then inking the outline on top.

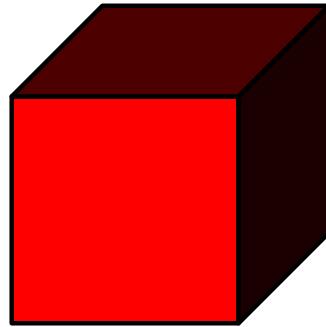
When I was a little girl, my father entertained me by teaching me to draw 3D shapes, including a cube, so we'll make that our example.

```
% draw 3d cube
numeric u; u := 6mm;
path cfront; cfront := (0,0) -- (5u,0)
  -- (5u,5u) -- (0,5u) -- cycle;
path ctop; ctop := (0,5u) -- (5u,5u)
  -- (7u,7u) -- (2u,7u) -- cycle;
path csid; csid := (5u,0) -- (7u,2u)
  -- (7u,7u) -- (5u,5u) -- cycle;
```

```
fill cfront withcolor red;
draw cfront withpen pencircle scaled .1u;
% (black is the default color)

fill ctop withcolor .3red;
draw ctop withpen pencircle scaled .1u;

fill csid withcolor .1red;
draw csid withpen pencircle scaled .1u;
```



### Credits

My thanks to Hans Hagen for the plain MetaPost shading trick and for patiently answering my questions about color in MetaPost.

### References

Running MetaPost and Metafun:

<http://tug.org/metapost/runningmp.html>

MetaPost manual:

<http://tug.org/docs/metapost/mpman.pdf>

MetaFun manual:

<http://www.pragma-ade.com/general/manuals/metafun-p.pdf>

The Other RGB Color Chart:

<http://www.tayloredmktg.com/rgb/>

◊ Mari Voipio  
 mari dot voipio (at) lucet dot fi  
<http://www.lucet.fi>

[Editor's note: This was not a submitted presentation at the TUG'13 meeting; it is included in this issue as a bonus article.]

## TUG 2013 abstracts

Editor's note: The slides and other samples for many of the talks are posted at <http://tug.org/tug2013/program.html>.

— \* —

### Jin-Hwan CHO

*A case study on TeX's superior power: Giving different colors to building blocks of Korean syllables*  
 In 2007 Dave Walden, the instigator and primary interviewer of TUG's Interview Corner, tossed a tricky question at me: "One of the concerns of many people in the TeX world is that TeX is relatively unknown in the larger worlds of typesetting and word processing, compared with commercial programs such as Adobe's InDesign and Microsoft Word. How do you see the future of TeX when it comes to Asian languages?" Since then, it has been my mission to find a wonderful answer, that is, a TeX product which other programs cannot reproduce.

Unicode contains 11,172 modern Korean syllables, all of which are composed by only 24 building blocks. In this talk, I will show an interesting TeX example containing a large number of Korean syllables each of which is grouped by building blocks of different colors. Nobody, of course, would try to reproduce this example with other commercial programs.

### Hans HAGEN

*How we try to make working with TeX comfortable*  
 Just as book and music production is under pressure, so is the way we produce documents. We're accustomed to instant rendering in browsers and even if WYSIWYG is not that important when most of the time is spent on writing instead of messing with the look and feel, there is the comfort factor to keep in mind. The last few years I have spent quite some time on a comfortable edit-proofing cycle: from advanced syntax highlighting to fast rendering. Do such things matter and is it worth the effort?

### Hans HAGEN

*How we move(d) on with math*

Given the amount of time I spend on LuaTeX and ConTeXt I occasionally ask myself if it really makes sense to do this. The answer to that question is determined by several factors. Probably the most important factor is the user base: what are their demands, how do they like to code, what control do they want, and therefore, where can these tools be of help? Another factor is relevance: can this combination do certain things better than other tools? One area that has always drawn users is math typesetting.

So, how up to date is TeX in that respect? Can we still claim victory there? Did we evolve well? Can we survive?

### Shizuya HAKUTA

*LISP on TeX: A LISP interpreter written using TeX macros*

Although TeX macros are useful, writing macros can be difficult for novice users. To make TeX easier to use, there is some research combining TeX and another programming language. Approaches have included calling an external interpreter and embedding an additional language in a member of the TeX engine family. We have taken yet another approach, possible because TeX is a Turing machine: implementing a language processor with TeX macros. The result, called 'LISP on TeX', allows us to embed LISP scripts in a LATEX document. The interpreter is written entirely with TeX macros and it is available through CTAN (<http://ctan.org/pkg/lisp-on-tex>). In this talk, we would like to illustrate how to use it and contrast it with LuaTeX, PerlTeX, and related approaches.

### Yoshifumi MAEDA & Masataka KANEKO

*Making math textbooks and materials with TeX+KETpic+hyperlinks*

Because of its precision and simplicity, the graphics capability originally present in TeX should have great potential in mathematics education. However, it seems to be burdensome for typical TeX users to fully utilize such capability. Although including graphical images generated by using computer algebra systems (CAS) is a typical alternate approach, the resulting documents tend to become inefficient for practical use in a classroom.

The CAS macro package named KETpic is one of the most hopeful candidates for realizing convenient and efficient use of TeX graphics. For instance, it enables us to edit high-quality math textbooks and materials containing: 2D-graphics which are precise in shape and length, and 3D-graphics which are readily understandable.

In this talk, we emphasize that the programmability of KETpic (associated with CAS) and TeX could make the use of TeX more flexible. For example, many documents with graphics can be readily generated by using both for-loop programming and "meta commands" of KETpic, and those documents can be readily linked also by using the `hyperref` package (connected to the "hypertextlink" function).

Such unified use of TeX graphics and TeX programming through KETpic might be applicable to many other situations in math classrooms, and should enhance the possibility of TeX use in education.

**Yasuhide MINODA***TeX in educational institutes*

Tokyo Educational Institute (Tetsuryokukai) is a preparatory school specializing in the entrance exam for Tokyo University. We use TeX for our texts, workbooks, other handouts, and even for internal documents and memorandums.

We used other software in the past, but we switched to TeX and converted our original documents (over 100,000 pages) into TeX files over the last few years.

In Tetsuryokukai, we now have over 200 teachers, with various levels of computer skill, so in order to introduce TeX we:

- developed related software (automatic installer, TeX2img and so on),
- prepared various style files,
- educate and motivate teachers.

In this presentation, I would like to present what we have been doing in our company, in the hope that it can be an interesting and helpful example of introducing TeX throughout an institution, especially in the field of education.

**Frank MITTELBACH***The stony road to complex page layout*

We discuss the challenges encountered in attempting to automate complex page layout. What are the real life use cases? How can they be approached? What remains unresolved after more than three decades of TeX programming efforts and why? (Slides and video at <http://www.latex-project.org/papers/>.)

**Frank MITTELBACH***LATEX3: Using the layers*

In this talk we will briefly present the architecture of LATEX3 with its four conceptual layers: document interface layer; typesetting element layer; document design layer; programming layer.

We will then look in some detail at `xparse`—a LATEX $2_{\varepsilon}$ -like user interface, as an example of the LATEX3 document interface layer, that can already be used to provide extended functionality for existing LATEX $2_{\varepsilon}$  documents and packages.

We conclude with a brief tour of `expl3`, the foundation layer for LATEX3 that provides the basis for all higher-level modules of LATEX3 but can also be usefully deployed to develop packages for LATEX $2_{\varepsilon}$ .

The `expl3` language is by now in a stable state and gets more and more traction outside the LATEX3 development work, which can be seen, for example, by its use in a growing number of answers on the question and answer portal <http://tex.stackexchange.com> and in the appearance of LATEX $2_{\varepsilon}$  packages that

use it for programming. (Text and diagram based on a previous talk by Joseph WRIGHT.)

**Ross MOORE***Making mathematical content accessible using Tagged PDF and LATEX*

'Tagged PDF' (more specifically PDF/UA) is the method developed by Adobe to allow the Web Content Accessibility Guidelines (WCAG 1.0, WCAG 2.0) to be satisfied within PDF documents. In this talk I will show the latest developments on using an extended version of pdftEX to allow Tagged PDF documents to be produced, satisfying both PDF/A (Archivability) and PDF/UA (Universal Accessibility).

I'll show examples which include quite complicated mathematical expressions, fully tagged with MathML, which can be 'Read Aloud' in Adobe's Acrobat and free Reader software. These will include 'real-world' documents containing such features as top-matter, nested list environments, logos, watermarks and other pagination artifacts, tabular material within mathematics, and some support of colour and text-styling. A special math-indexing feature has been developed, which allows the result of processing by external programs to be identified and reused in successive LATEX runs. This indexing feature leads to significant time savings when developing a full document over many processing runs.

The full paper is available at <http://ceur-ws.org/Vol-1010/paper-01.pdf>.

**Keiichiro SHIKANO***Indexing makes your book perfect*

Most of you already know how to make books using LATEX. And some of you might know how to make back-of-the-book indexes with LATEX. However, are you ready to worry about how the index of your book should be? Or, if you have already gone through a trouble of writing or editing books, have you actually taken advantage of indexing in your work?

The index, which would be inserted at the back of your book, is not just a reference list of words

appearing in your book. Picking out keywords or chunks of text from your manuscript, then arranging them in another way—usually in alphabetical order, often complements your book. In other words, you can exploit indexing to make your book better!

Through this tutorial, you will find what is required for good indexes, how indexing helps you and your readers, and some techniques for making back-of-the-book indexes with L<sup>A</sup>T<sub>E</sub>X. On top of that, in non-alphabetical languages, you cannot simply use `makeindex` or `xindy`, mainly because these languages don't have any concept of alphabetical order. So, I will also go over practical cases of making back-of-the-book indexes in non-alphabetical languages.

### **Yumi TAKATA**

*Japanese typeface design — similarities and differences from Western typeface design*

What is Japanese typeface design about? As a Japanese type designer for nearly 30 years, I will explain what it is to design a Japanese typeface, and what it does and does not have in common with designing a Western typeface.

First, we will take a quick look at the history of Japanese characters, in particular how the shapes of the characters have evolved through time.

Then I will illustrate the process of Japanese typeface design in detail. Japanese typeface designers face the challenge of dealing with more than 9,000 characters and multiple constituent scripts. Some examples will be given of the various techniques we use to create readable and visually appealing typefaces, including adjusting for optical illusion.

Another big challenge we face is the vertical and horizontal writing modes. I will show how we fine-tune the glyph design of each character, one by one, for both vertical and horizontal writing modes.

Finally, the complications related to Japanese coded character sets will be briefly explained.

I hope my presentation gives you a grasp of the Japanese typeface design and leads to further discussion.

### **Yusuke TERADA**

*Development of TeXShop — the past and the future*

TeXShop is a widely-used open source T<sub>E</sub>X editor and previewer for Mac OS X. TeXShop is developed by Richard Koch, emeritus professor of mathematics at the University of Oregon, and many other worldwide contributors, including me. Now it is localized for as many as 10 languages. While it has already sufficient functions for editing T<sub>E</sub>X documents, TeXShop is still being updated. In this presentation, I will give an outline of the design concept of TeXShop and some new features that have been added recently, es-

pacially for editing Japanese documents. In addition, I will show a vision of TeXShop for the future.

### **Didier VERNA**

*TiCL: The prototype*

Last year, I presented some ideas about using one of the oldest programming language (Lisp), in order to modernize one of the oldest typesetting systems (T<sub>E</sub>X). That talk was mostly focused on justifying the technical fitness of Lisp for this task. This time, I would like to take the opposite view and demonstrate a prototype from the user's perspective. This will involve showing what a TiCL document could look like, the implications in terms of typesetting vs. programmatic features, and also in terms of extensibility (relating this to package authoring).

### **Alan WETMORE**

*Wind roses for T<sub>E</sub>X documents*

In recent years a great many systems for including plots and graphics in T<sub>E</sub>X documents have been developed. Many varieties of scientific plots are directly supported by these packages. One style of plot which has not been available is a wind rose: describing the probability of wind speed and direction with a stylized polar plot. This report will describe a set of macros for TikZ for preparing wind rose plots.

### **Masafumi YABE**

*Japanese text layout — basic issues*

This tutorial presents basic issues concerning page formats and typesetting methods applied to the main text of a Japanese book with reference to the typographic characteristics of the Japanese writing system. The issues to be discussed are threefold.

The first section focuses on the text direction, vertical or horizontal writing mode, which depends on an editorial decision and affects, in many ways, the page layout as well as the printed forms of a Japanese text.

The second section concerns typesetting methods applied to basic Japanese text as a sequence of characters without spaces between words, and illustrates relevant typographic building blocks in line with composition rules with an emphasis on the functional importance of punctuation marks and their surrounding spaces for line and paragraph adjustments.

The last section addresses several issues about methods for mixed composition of Japanese and Western texts, presenting major technical problems relating to differentiation and harmonization of typographically heterogeneous elements in sequential texts: Western text in the context of main horizontal or vertical Japanese text as well as Japanese text in the context of main Western text.

---

**ConTeXt Group: Proceedings,  
6th meeting (2012)**

The ConTeXt Group publishes proceedings of the annual ConTeXt meetings.  
<http://group.contextgarden.net>.

Dayplan; pp. 5–6  
 Schedule of talks.

MARI VOIPIO, CraftTeX; p. 7  
 [Expanded version published in *TUGboat* 33:3.]

TACO HOEKWATER, MetaPost: PNG Output;  
 pp. 8–9  
 [Published in *TUGboat* 34:2.]

PATRICK GUNDLACH, Database publishing with  
 LuaTeX and the speedata Publisher; p. 10

Database publishing is the repetitive (semi-)automatic transformation from a data source to some kind of output (HTML, PDF, EPUB, ...). A common demand in high volume output is to optimize page usage. Our software (called ‘speedata Publisher’) is written in Lua and makes heavy use of the LuaTeX engine. We use TeX to break paragraphs into lines, arrange the programmatically created boxes and glue for layout of complex tables and to write clean PDF. The publisher is released under the AGPL.

WILLI EGGER, Minutes of the 2nd ConTeXt  
 Group membership meeting; pp. 11–12

TACO HOEKWATER, MetaPost path resolution  
 isolated; pp. 13–18

A new interface in MPLib version 1.800 allows one to resolve path choices programmatically, without the need to go through the MetaPost input language.

TACO HOEKWATER, Parsing PDF content streams  
 with LuaTeX; pp. 19–23

The new `pdfparser` library in LuaTeX allows parsing of external PDF content streams directly from within a LuaTeX document. This paper explains its origin and usage.

LIGU SCARSO, MFLua: Instrumentation of  
 METAFONT with Lua; pp. 24–35  
 [Published in *TUGboat* 32:2.]

WILLI EGGER, Conference portfolio; pp. 36–40

In accordance with the conference’s theme, a workshop for making a portfolio binder was held. The portfolio was made so it could carry the papers for the conference, such as preprints of the proceedings, additional papers and the carpenter’s pencil given to each participant. The construction is made from a single sheet of cardboard with folded flaps along

three sides, so that it completely envelops the content. The portfolio is held closed by a black elastic band.

HANS HAGEN, Simple spreadsheets; pp. 41–51

Occasionally a question pops up on the ConTeXt mailing list such that answering it becomes a nice distraction from a boring task at hand. The spreadsheet module is the result of such a diversion. As with other support code in ConTeXt, this is not a replacement for ‘the real thing’ but just a nice feature for simple cases. Of course some useful extensions might appear in the future.

HANS HAGEN and IDRIS SAMAWI HAMID,  
 Oriental TeX: Optimizing paragraphs; pp. 52–81

One of the objectives of the Oriental TeX project has always been to play with paragraph optimization. The original assumption was that we needed an advanced non-standard paragraph builder to handle Arabic correctly, but in the end we found that a more straightforward approach is to use a sophisticated OpenType font in combination with a paragraph postprocessor that uses the advanced font capabilities. This solution is somewhat easier to imagine than a complex paragraph builder but still involves quite some juggling.

JEAN-MICHEL HUFFLEN, MIBIBTeX and its new  
 extensions; pp. 82–91

These last years, MIBIBTeX’s kernel functions have been reused and extended in order to put new programs about bibliographies into action. Examples are the `hal` program, allowing an open archive site to be populated, the `mlbiblate` program, building bibliographies suitable for the `biblate` package, the `mlbibcontext` program, doing the same task for ConTeXt documents. We show how all these programs are organised, and explain how some operations can be refined or extended. From an efficiency point of view, the programs `mlbiblate` and `mlbibcontext` are written using Scheme only, so they are more efficient than analogous programs that would interpret a `.bst` bibliography style of BIBTeX.

JEAN-MICHEL HUFFLEN, Demonstration of the  
`mlbibcontext` program; pp. 92–93

This short statement aims to sketch the broad outlines of the presentation performed at the 6th ConTeXt meeting.

Abstracts; pp. 94–95

Participant list of the 6th ConTeXt meeting;  
 pp. 96–97

[Received from Taco Hoekwater.]

**MAPS 44 (Spring 2013)**

MAPS is the publication of NTG, the Dutch language TeX user group (<http://www.ntg.nl>).

TACO HOEKWATER, Redactieel [From the editor]; pp. 1–2

HANS HAGEN, Does TeX have a future; pp. 3–7  
[Published in *TUGboat* 34:2.]

KEES VAN DER LAAN, CD and DVD labels; pp. 8–12

Making CD and DVD labels by PostScript, to be printed on prefabricated glued paper, assisted by Photoshop for the conversion of an illustration into EPSF, is explained.

KOEN WYBO, Review of *L<sup>A</sup>T<sub>E</sub>X and Friends* by Marc van Dongen; pp. 13–14

Saying that L<sup>A</sup>T<sub>E</sub>X is not easy to learn is a truism. With a good book like *L<sup>A</sup>T<sub>E</sub>X and Friends*, you will more than adequately be put on the road.

C.M. FORTUIN, Kegelsneden benaderen [Conic approximation]; pp. 15–26

Conic sections can systematically be approximated by vertices of a (part of a) circumscribed polygon. An algorithm is developed for the determination of the support points of a third degree iterative “Bézier” approach of a conic. Initially, three points are needed. The algorithm is independent of the position of the conic section.

KEES VAN DER LAAN, Pythagoras Trees in PostScript; pp. 27–48

Pythagoras Trees are drawn elegantly in PostScript, varied by randomness, colour and the use of curves. Lindenmayer production rules for systematic PS program development are enriched by PS concepts.

KEES VAN DER LAAN, Classical Math Fractals in PostScript; pp. 49–78

Classical mathematical fractals in BASIC are explained and converted into lean-and-mean EPSF **defs**, of which the .eps pictures are delivered in .pdf format and cropped to the prescribed **BoundingBox** when processed by Acrobat Pro, to be included easily in pdf(L<sup>A</sup>)TeX, Word, ... documents. The EPSF fractals are transcriptions of the Turtle Graphics BASIC codes or programmed anew, recursively, based on the production rules of oriented objects. The Lindenmayer production rules are enriched by PostScript concepts. Experience gained in converting a TeX script into WYSIWYG Word is communicated.

HANS VAN DER MEER, Exam Papers Revisited; pp. 79–90

Described is a module for the consistent production and maintenance of student examinations. It can typeset questions with long or short answers, yes/no questions and multiple choice. The questions are formulated as XML documents and access ConTeXt through a special interface with HTML-like syntax.

HANS VAN DER MEER, A bit of HTML and a bit of ConTeXt; pp. 91–96

Described is a module for the typesetting of a subset of HTML operators. These can be used to build data sets in XML with HTML as formatting elements and have them typeset in ConTeXt. Other features are the inclusion of predefined content and provision for language localized words and expressions.

HANS VAN DER MEER, Yet Another Table; pp. 97–105

Described is a module for the typesetting of tables. The module resembles the L<sup>A</sup>T<sub>E</sub>X **tabular** environment but is in fact based on a much older package, the origins of which are lost to the author.

SIETSE BROUWER, Making the ConTeXt wiki easier to improve; pp. 106–108

An effort is underway to encourage both reading and editing of the ConTeXt wiki. This article names nine concrete improvements that are part of this effort, and makes a case for each of them. These nine items are the following. To impose structure and to ease navigation: predictable article names; navboxes; and a simple Main Page. To coordinate efforts: a “How this wiki works” page; a village pump; and templates for flagging problems. To make things easy for our editors: templates for common things; template documentation; sandboxes and testcases for templates.

TACO HOEKWATER, MetaPost: Numerical engines; pp. 109–113

After years of talks about future plans for MetaPost 2.0, finally real progress is being made. This paper introduces a pre-release of MetaPost 2 that can optionally use IEEE floating point for its internal calculations instead of the traditional 32-bit integers.

HANS HAGEN, Simple Spreadsheets; pp. 114–122  
A ConTeXt spreadsheet module, based on Lua.

MICHAEL GURAVAGE, 5<sup>th</sup> International ConTeXt Meeting; pp. 123–126  
Conference report.

[Received from Wybo Dekker.]

---

### **Die TeXnische Komödie 3–4/2013**

*Die TeXnische Komödie* is the journal of DANTE e.V., the German-language TeX user group (<http://www.dante.de>). [Non-technical items are omitted.]

#### **Die TeXnische Komödie 3/2013**

MARKUS KOHM, Was ist eigentlich: die Besonderheit des @-Zeichens in Befehlsnamen? [What's special about @ in command names?]; pp. 16–23

Nowadays many TeX users find their information about (La)TeX not only in books but somewhere on the Internet. Often the responder uses terms and things for which the knowledge about their meaning is assumed. "What's special" shall be a loose series of articles that deliver the meaning behind common terms. This installment covers the @ within macro names. Especially with questions that are not easily solved by using a special class or package, the TeX user is confronted with commands that have a @ in their names. This may raise the suspicion that they cannot be used like other commands.

ROGER JUD, Autovervollständigung mit TeXnicCenter [Auto-completion with TeXnicCenter]; pp. 24–27

This article shows how to use and customize TeXnicCenter's auto-completion.

MARCO DANIEL, Das Paket minted und der Apostroph [The `minted` package and the apostrophe]; pp. 28–29

Highlighting program listings with the help of Pygmentize becomes increasingly popular. A small disadvantage of this method is handling of apostrophes, which are unfortunately often used in various programming languages and are displayed with serifs in the output. This article shows how to fix this.

ROGER JUD, Eulersche Gerade mit `tkz-euclide` zeichnen [Drawing Euler's straight lines with `tkz-euclide`]; pp. 30–36

Using the Euler straight line construction, this article presents some functions of the `tkz-euclide` package.

HERBERT VOSS, Schleifenmakro [Loop macro]; p. 37

There are various packages that offer macros for loops. Since nowadays ε-TEX features are universally available, one may define his own macro working with recursion.

#### **Die TeXnische Komödie 4/2013**

MICHAEL PIEFEL, UML-Diagramme in LATEX-Dokumenten [UML diagrams in LATEX documents]; pp. 21–28

For software developers UML is a commonly used modelling language with a clear graphical notation. In particular, class diagrams can be used to document object-oriented analyses and designs. This article shows several approaches to embed and create such diagrams in LATEX documents.

AXEL KIELHORN, LATEX auf Mobilgeräten [LATEX on mobile devices]; pp. 29–33

Tablets are devices for consumers, not for producers. That's the common opinion. An article in *TUGboat* 33:2 (2012) tried to refute this by installing LATEX on an Android tablet, which required installing GNU/Linux. Clearly this solution was not optimal! This article shows what has happened since then.

HERBERT VOSS, QR-Codes im Rand ausgeben [Printing QR codes in the margin]; pp. 34–37

In books and lecture transcripts there are more and more references to external literature given as QR codes. Especially for electronic media, this can be useful. QR codes can be easily created with `pst-barcode`, while shortening a URL requires more effort.

ROGER JUD, Punktzahlen addieren und ausgeben [Adding points and printing them at the beginning of the document]; pp. 38–40

A teacher's life involves creating exams, exercise sheets, etc. Often one assigns points to individual exercises that then need to be added up. This article shows how this can be achieved without using dedicated document classes (such as `exam` by Philip Hirschhorn).

[Received from Herbert Voß.]



## The Treasure Chest

This is a list of selected new packages posted to CTAN (<http://ctan.org>) from July through December 2013, with descriptions based on the announcements and edited for extreme brevity.

Entries are listed alphabetically within CTAN directories. A few entries which the editors subjectively believe to be of especially wide interest or otherwise notable are starred; of course, this is not intended to slight the other contributions.

We hope this column and its companions will help to make CTAN a more accessible resource to the *TEX* community. Comments are welcome, as always.

◊ Karl Berry  
 tugboat (at) tug dot org  
<http://tug.org/ctan.html>

### biblio

#### **besjournals** in *biblio/bibtex/contrib*

For journals published by the British Ecological Society.

### fonts

#### **accanthis** in *fonts*

An old style serif font.

#### **alegreya** in *fonts*

A dynamic serif design intended for literature.

#### **anonymouspro** in *fonts*

A monospaced font in several styles.

#### \* **fbb** in *fonts*

Bembo-like font based on Cardo.

#### **fandol** in *fonts*

Four fonts for Chinese typesetting. (See article in this issue.)

#### **gillius** in *fonts*

A sans serif inspired by Gill Sans.

#### **merriweather** in *fonts*

Harmonized serif and sans serif design with a large x-height and open forms.

#### **mintspirit** in *fonts*

A distinctive sans serif.

#### **xcharter** in *fonts*

Adds oldstyle figures, small caps, and more to Charter.

### graphics

#### **harveyballs** in *graphics/pgf/contrib*

Draw Harvey Balls.

### biblio/bibtex/contrib/besjournals

**neuralnetwork** in *graphics/pgf/contrib*  
 Draw neural networks and other colorful graphs.  
**timing-diagrams** in *graphics/pgf/contrib*  
 Draw timing diagrams.

### language

**bxcjkjatype** in *language/japanese/bxcjkjatype*  
 Support for Japanese typesetting with pdflATEX and the *cjk* package.

**kotex-oblivoir** in *language/korean*  
 Document class for Korean based on *memoir*.

**kotex-plain** in *language/korean*  
 Typeset Hangul using plain *TEX*.

**kotex-utf** in *language/korean*

Typeset Hangul, with input in UTF-8.

**kotex-utils** in *language/korean*  
 Scripts and support files for index generation and Korean typesetting.

### macros/generic

**pdf-trans** in *macros/generic*  
 Transformations of *TEX* boxes.

### macros/latex/contrib

**askmaps** in *macros/latex/contrib*  
 American-style Karnaugh maps.

**brandeis-dissertation** in *macros/latex/contrib*  
 Brandeis dissertations.

**embedall** in *macros/latex/contrib*  
 Attach all source files to the generated PDF.

**graphviz** in *macros/latex/contrib*  
 Write graphviz inline in LATEX documents.

\* **grid-system** in *macros/latex/contrib*  
 Support the grid system as known from HTML and CSS.

**gtl** in *macros/latex/contrib*  
 Manipulate generalized token lists.

**guitarchordschemes** in *macros/latex/contrib*  
 Typeset guitar chord and scale tablatures.

**idxcmds** in *macros/latex/contrib*  
 Indexing with semantic commands.

**lexref** in *macros/latex/contrib*  
 For European law, especially Swiss and German.

**lt3graph** in *macros/latex/contrib*  
 A graph data structure for *expl3*.

**macroswap** in *macros/latex/contrib*  
 Swap the meaning of two macros by name.

**metrix** in *macros/latex/contrib*  
 Typeset metrics (prosodics) as symbols or above the syllables of a verse.

**minorrevision** in *macros/latex/contrib*  
 Quote and refer to a manuscript for minor revisions.

---

**noindentafter** in `macros/latex/contrib`  
Suppress indentation after environments, etc.

**pas-cours** in `macros/latex/contrib`  
Write math lessons using TikZ.

**pas-crosswords** in `macros/latex/contrib`  
Typeset crossword grids and definitions.

**pas-tableur** in `macros/latex/contrib`  
Make spreadsheets using TikZ.

**phonrule** in `macros/latex/contrib`  
Typeset linear phonological rules as in Chomsky's *The Sound Pattern of English*.

**reflectgraphics** in `macros/latex/contrib`  
Fancy reflections for L<sup>A</sup>T<sub>E</sub>X graphics.

**ribbonproofs** in `macros/latex/contrib`  
Draw "ribbon proofs", a diagrammatic representation of a mathematical proof that a computer program meets its specification.

**sslides** in `macros/latex/contrib`  
A slides class including header and footer.

**translations** in `macros/latex/contrib`  
Internationalization of L<sup>A</sup>T<sub>E</sub>X 2<sub>E</sub> packages.

**unravel** in `macros/latex/contrib`  
Step through L<sup>A</sup>T<sub>E</sub>X code for debugging.

**with-macro** in `macros/latex/contrib`  
Pass token lists as parameters.

**xcjk2uni** in `macros/latex/contrib`  
Convert CJK characters to Unicode in pdfL<sup>A</sup>T<sub>E</sub>X.

---

**macros/latex/contrib/beamer-contrib**

**themes/phnompenh** in `m/l/c/beamer-contrib`  
Simple Beamer theme.

---

**macros/latex/contrib/biblatex-contrib**

**biblatex-source-division** in `m/l/c/biblatex-contrib`  
Support divisions of sources in references.

---

**macros/luatex**

**lilyglyphs** in `macros/luatex/lateX`  
Include musical symbols from LilyPond in LuaL<sup>A</sup>T<sub>E</sub>X or X<sub>L</sub><sup>A</sup>T<sub>E</sub>X.

**simurgh** in `macros/luatex/lateX`  
Typeset Parsi in LuaL<sup>A</sup>T<sub>E</sub>X.

---

**support**

**datatooltk** in `support`  
Java application to accompany `datatool` package.

**ltximg** in `support`  
Isolate and convert all TikZ or PSTricks graphics.

**tlg2latex** in `support`  
Convert text from Thesaurus Linguae Graecae.

---

**In memoriam: Jean-Pierre Drucbert (1947–2009)**

Denis Bitouzé, on behalf of GUTenberg



**Jean-Pierre Drucbert**

GUTenberg, the French T<sub>E</sub>X user group, has learned with great sadness that Jean-Pierre F. Drucbert passed away on January 25, 2009. Jean-Pierre wrote several notable L<sup>A</sup>T<sub>E</sub>X packages, including `minitoc` and `xr`; a complete list can be found at <http://www.ctan.org/author/drucbert>.

We asked colleagues of Jean-Pierre about his wishes concerning his packages, but they don't know. Their license (LPPL) allows anybody of good will to undertake their development.

Born in Lille, France, in 1947, Jean-Pierre Drucbert earned a degree in 1970 in engineering at the École Nationale Supérieure de l'Aéronautique et de l'Espace (National School of Aeronautics and Space).

After majoring in computer science, he joined the Centre d'Études et de Recherches de Toulouse (now ONERA) as a systems engineer, mainly in the field of IT applied to scientific computing. Jean-Pierre set up and administered several generations of computers (including an CII-Iris 80 and Cray XMP), particularly for scheduled computational jobs. Moreover, he provided IT support to hundreds of users, mostly scientists, especially in the field of parallel computing.

Meanwhile, fascinated by the sciences of language and typography, Jean-Pierre showed an early interest in T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X. He compiled and installed them on several different systems, from Multics to GNU/Linux through various proprietary Unix systems. He offered to ONERA users a very complete L<sup>A</sup>T<sub>E</sub>X manual, including French translation of many packages' documentation. He contributed for more than twenty years to the L<sup>A</sup>T<sub>E</sub>X community by developing packages.

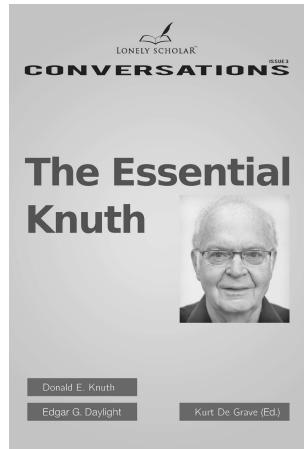
He will be missed.

---

**Book review: *The Essential Knuth***

David Walden

Donald E. Knuth and Edgar G. Daylight, *The Essential Knuth*. Lonely Scholar, 2013, 94 pp. Paperback, US\$15.00. ISBN 978-9491386039.



*The Essential Knuth* is primarily an interview of Don Knuth by Edgar Daylight. It is published as a booklet of about 90 pages divided into six chapters: 1. Childhood, 2. College, 3. ALGOL, 4. Structured Programming, 5. Computer Pioneers, 6. Historiography. There is also a preface by the interviewer, a 66-item bibliography of books and papers mentioned during the interview, and an index. The title page states that the booklet was edited by Kurt De Grave. In other words, it is longer and more richly documented and carefully published than many interviews.<sup>1</sup> The interview was done in November 2012 in Frankfurt, Germany.

Edgar Daylight is a researcher in the history of programming languages living in Belgium, with a goal of extensively interviewing high-profile and retired computer scientists.<sup>2</sup> Thus, his interview of Knuth deals with Knuth's work with computer languages and language processors, Knuth's interest in the methodology of structured programming, and Knuth's personal involvement with the computing pioneers of the 1950s and 1960s who were involved with computer language research, such as Ole-Johan Dahl, Edsger Dijkstra, C. A. R. (Tony) Hoare, and Peter Naur. There is almost no mention of TeX in the booklet.

I am happy I bought and read the booklet. It covers some things about Knuth that I had not read elsewhere and gives a slightly different slant on some

things of which I was already aware. Although the booklet doesn't go into Knuth's work with TeX, that was OK with me; plenty has already been written (by Knuth and others) about TeX in this our TeX community. Rather, because I came into computing in the 1950s and 1960s and knew of many of the computing pioneers from that era through their books and journal articles, I greatly enjoyed learning more about Knuth's interactions with the rest of those pioneers. Also, I am interested in the practice of researching and writing computing history, and the booklet contains some discussion, both implicit and explicit, of this. For those interested in more detail in this regard, I wrote another review for the *IEEE Annals of the History of Computing*; a preprint of that review is at <http://walden-family.com/ieee/daylight-knuth.pdf>.

I recommend the booklet. However, I suspect that many potential readers will wish it was available in ebook form.

\* \* \*

TeX is mentioned in the booklet in a couple of discussions. Regarding specifying programs, Knuth states that he could not have specified TeX without also being a user of TeX. Regarding structured programming, Knuth notes that TeX is not entirely structured, as it includes macros which don't have to obey nested structure. Regarding portability of programs, Knuth states,

[W]hen I wrote TeX, I was extremely careful about portability. I completely avoided floating point arithmetic in places where the computations could affect page layout. Instead I implemented my own arithmetic for internal computations, using integer operations only, and I checked boundary conditions so my programs would be machine-independent. I wanted to be sure that everyone who uses TeX would get the same results, regardless of the country they lived in and regardless of the operating system they were using, either now or fifty years from now.

Regarding this machine portability point, elsewhere in the booklet Knuth states that he is not good at forecasting, for example, how big computer memories would become. I wonder if he also didn't anticipate a standard specification for floating point computer arithmetic; or if he anticipated a specification but couldn't wait; or if he would in any case have developed his own portable arithmetic capability for TeX.

<sup>1</sup> Some of them are available at <http://tug.org/interviews/#knuth>.

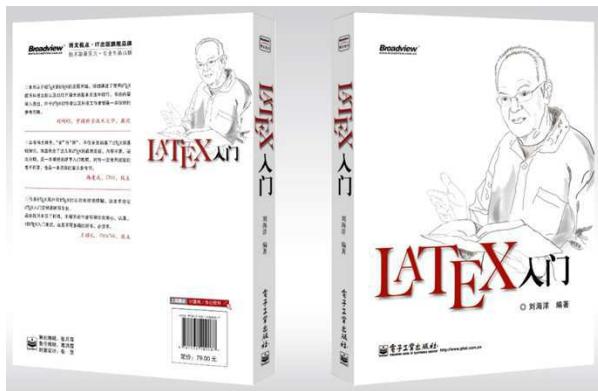
<sup>2</sup> Email of August 3, 2012, by Daylight to SIGCIS.org.

◊ David Walden  
<http://www.walden-family.com/texland>

## Book review: *Introduction to L<sup>A</sup>T<sub>E</sub>X*

Clerk Ma

Leo Liu, *Introduction to L<sup>A</sup>T<sub>E</sub>X*, Publishing House of Electronics Industry (China), July, 2013, 566 pp., Paperback, CNY ¥79, ISBN 978-7-121-20208-7.  
(刘海洋, *L<sup>A</sup>T<sub>E</sub>X 入门*, 电子工业出版社, 2013 年 7 月, 566 页, 简装, 79 元, ISBN 978-7-121-20208-7.)



It is not too difficult to typeset English with L<sup>A</sup>T<sub>E</sub>X. But when L<sup>A</sup>T<sub>E</sub>X meets the Chinese language, the task becomes tougher. When I used L<sup>A</sup>T<sub>E</sub>X to typeset my first equation in 2009, I knew only one option: the CJK package [1]. At that time I could not find any book to help me to learn L<sup>A</sup>T<sub>E</sub>X typesetting of Chinese in depth. (I ended up using the CJK package for a long time, then switched to xeCJK [2] in 2010 and LuaTeX-ja [3] in 2012.)

Recently I at last found a book sold in bookstores and devoted to L<sup>A</sup>T<sub>E</sub>X typesetting of the Chinese language. The book covers every aspect of L<sup>A</sup>T<sub>E</sub>X interesting to Chinese (and most) users: float objects, tables, mathematical formulae, beamer presentations. Moreover, at the end of the book there is a short course of programming in L<sup>A</sup>T<sub>E</sub>X and plain T<sub>E</sub>X for the benefit of those readers who want to write their own macros. With many detailed examples, I feel the book should be recommended both to L<sup>A</sup>T<sub>E</sub>X beginners and more experienced users wanting to typeset Chinese.

This book recommends using the X<sub>E</sub>T<sub>E</sub>X package xeCJK to typeset Chinese. In the world of multilingual documents, X<sub>E</sub>T<sub>E</sub>X has become a very popular and convenient T<sub>E</sub>X engine. The package works very well with both Chinese and (with zxjafbf [6]) Japanese. As far as I know, xeCJK is the best implementation of Chinese typography.

The author, Leo Liu, is a resident expert of the largest Chinese T<sub>E</sub>X forum CT<sub>E</sub>X [4]. He is the maintainer of xeCJK and zhmcjk [5].

◊ ◊ ◊

用 L<sup>A</sup>T<sub>E</sub>X 来排版英文并不复杂。但是, 当 L<sup>A</sup>T<sub>E</sub>X 遭遇中文的时候, 事情就开始变得棘手。在 2009 年的时候, 我用 L<sup>A</sup>T<sub>E</sub>X 排出了我第一个数学公式, 当时我只知道有 CJK 能处理相关的中文问题。在当时, 我找不到一本能引导我学习 L<sup>A</sup>T<sub>E</sub>X 的书, 尤其是处理中文方面。我用 CJK 包用了很长时间, 之后我换到了 xeCJK (2010 年) 和 LuaTeX-ja (2012 年)。最近, 我发现书店里面正在卖一本新 L<sup>A</sup>T<sub>E</sub>X 书。该书覆盖了 L<sup>A</sup>T<sub>E</sub>X 的方方面面: 浮动体处理, 表格, 数学公式, beamer。在该书末尾, 则给出了一个简短的关于 L<sup>A</sup>T<sub>E</sub>X 以及 plain T<sub>E</sub>X 的编程教程。这些内容可以满足那些想要实现自己的宏的读者们。书中附带了大量详尽的例子, 适合推荐给中国的初学者和在排版中文方面有问题的其他一些人们。该书主要使用 xeCJK 来排版中文。在 T<sub>E</sub>X 界, 使用 X<sub>E</sub>T<sub>E</sub>X 是最流行的也是最方便的。而 xeCJK 包可以完美处理中文和日文 (一个补丁版本叫做 zxjafbf)。据我所知, xeCJK 是中国排版传统的一个最好的实现。

该书作者, 刘海洋, 是中国最大论坛 CT<sub>E</sub>X 的 T<sub>E</sub>X 专家。他维护着 xeCJK 和 zhmcjk 两个包。

## References

- [1] Werner Lemberg, `cjk`—CJK language support, 2012. <http://ctan.org/pkg/cjk>.
- [2] Jiang Jiang, Qing Lee, Leo Liu and Wenchang Sun, `xecjk`—Support for CJK documents in X<sub>E</sub>T<sub>E</sub>X, 2013. <http://ctan.org/pkg/xecjk>.
- [3] Hironori Kitagawa, `luatexja`—Typeset Japanese with Lua(LA)T<sub>E</sub>X, 2013. <http://ctan.org/pkg/luatexja>.
- [4] CT<sub>E</sub>X: Chinese T<sub>E</sub>X. <http://www.ctex.org>.
- [5] Leo Liu, `zhmcjk`—Simplify configuration of CJK installations, 2012. <http://ctan.org/pkg/zhmcjk>.
- [6] Takayuki Yato, `zxjafbf`—Fallback CJK font support for xeCJK, 2012. <http://ctan.org/pkg/zxjafbf>.

◊ Clerk Ma  
clerkma@gmail.com

## TUG Institutional Members

American Mathematical Society,  
*Providence, Rhode Island*

Aware Software, Inc., *Midland Park, New Jersey*

Center for Computing Sciences, *Bowie, Maryland*

CSTUG, *Praha, Czech Republic*

Florida State University, School of Computational  
Science and Information Technology,  
*Tallahassee, Florida*

IBM Corporation, T J Watson Research Center,  
*Yorktown, New York*

Institute for Defense Analyses, Center for  
Communications Research, *Princeton, New Jersey*

Marquette University, Department of  
Mathematics, Statistics and Computer Science,  
*Milwaukee, Wisconsin*

Masaryk University, Faculty of Informatics,  
*Brno, Czech Republic*

MOSEK ApS, *Copenhagen, Denmark*

New York University, Academic Computing Facility,  
*New York, New York*

Springer-Verlag Heidelberg, *Heidelberg, Germany*

StackExchange, *New York City, New York*

Stanford University, Computer Science Department,  
*Stanford, California*

Stockholm University, Department of Mathematics,  
*Stockholm, Sweden*

University College, Cork, Computer Centre,  
*Cork, Ireland*

Université Laval, *Ste-Foy, Québec, Canada*

University of Ontario, Institute of Technology,  
*Oshawa, Ontario, Canada*

University of Oslo, Institute of Informatics,  
*Blindern, Oslo, Norway*

VTEX UAB,  
*Vilnius, Lithuania*

## TEX Consultants

The information here comes from the consultants themselves. We do not include information we know to be false, but we cannot check out any of the information; we are transmitting it to you as it was given to us and do not promise it is correct. Also, this is not an official endorsement of the people listed here. We provide this list to enable you to contact service providers and decide for yourself whether to hire one.

TUG also provides an online list of consultants at <http://tug.org/consultants.html>. If you'd like to be listed, please see that web page.

### **Aicart Martinez, Merce**

Tarragona 102 4<sup>o</sup> 2<sup>a</sup>  
08015 Barcelona, Spain  
+34 932267827  
Email: [m.aicart \(at\) ono.com](mailto:m.aicart(at)ono.com)  
Web: <http://www.edilatex.com>

We provide, at reasonable low cost, L<sup>A</sup>T<sub>E</sub>X or T<sub>E</sub>X page layout and typesetting services to authors or publishers world-wide. We have been in business since the beginning of 1990. For more information visit our web site.

### **Dangerous Curve**

PO Box 532281  
Los Angeles, CA 90053  
+1 213-617-8483  
Email: [typesetting \(at\) dangerouscurve.org](mailto:typesetting(at)dangerouscurve.org)  
Web: <http://dangerouscurve.org/tex.html>

We are your macro specialists for T<sub>E</sub>X or L<sup>A</sup>T<sub>E</sub>X fine typography specs beyond those of the average L<sup>A</sup>T<sub>E</sub>X macro package. If you use X<sub>H</sub>T<sub>E</sub>X, we are your microtypography specialists. We take special care to typeset mathematics well.

Not that picky? We also handle most of your typical T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X typesetting needs.

We have been typesetting in the commercial and academic worlds since 1979.

Our team includes Masters-level computer scientists, journeyman typographers, graphic designers, letterform/font designers, artists, and a co-author of a T<sub>E</sub>X book.

### **Latchman, David**

4113 Planz Road Apt. C  
Bakersfield, CA 93309-5935  
+1 518-951-8786  
Email: [technical.designs \(at\) gmail.com](mailto:technical.designs(at)gmail.com)  
Web: <http://www.elance.com/s/dlatchman>

Proficient and experienced L<sup>A</sup>T<sub>E</sub>X typesetter for books, monographs, journals and papers allowing your documents and books to look their possible best especially with regards to technical documents.

Graphics/data rendered either using TikZ or Gnuplot. Portfolio available on request.

**Peter, Steve**

295 N Bridge St.  
Somerville, NJ 08876  
+1 732 306-6309

Email: [speter \(at\) mac.com](mailto:speter(at)mac.com)

Specializing in foreign language, multilingual, linguistic, and technical typesetting using most flavors of *T<sub>E</sub>X*, I have typeset books for Pragmatic Programmers, Oxford University Press, Routledge, and Kluwer, among others, and have helped numerous authors turn rough manuscripts, some with dozens of languages, into beautiful camera-ready copy. In addition, I've helped publishers write, maintain, and streamline *T<sub>E</sub>X*-based publishing systems. I have an MA in Linguistics from Harvard University and live in the New York metro area.

**Shanmugam, R.**

No. 38/1 (New No. 65), Veerapandian Nagar, Ist St.  
Choolaimedu, Chennai-600094, Tamilnadu, India  
+91 9841061058

Email: [rshanmugam92 \(at\) gmail.com](mailto:rshanmugam92(at)gmail.com)

As a Consultant, I provide consultation, training, and full service support to individuals, authors, typesetters, publishers, organizations, institutions, etc. I support leading BPO/KPO/ITES/Publishing companies in implementing latest technologies with high level automation in the field of Typesetting/Prepress, ePublishing, XML2PAGE, WEBTechnology, DataConversion, Digitization, Cross-media publishing, etc., with highly competitive prices. I provide consultation in building business models & technology to develop your customer base and community, streamlining processes in getting ROI on our workflow, New business opportunities through improved workflow, Developing eMarketing/E-Business Strategy, etc. I have been in the field BPO/KPO/ITES, Typesetting, and ePublishing for nearly 20 years, and handled various projects. I am a software consultant with Master's Degree. I have sound knowledge in *T<sub>E</sub>X*, *I<sub>A</sub>T<sub>E</sub>X2<sub>E</sub>*, *XML<sub>T</sub>E<sub>X</sub>*, Quark, InDesign, XML, MathML, eBooks, ePub, Mobi, iBooks, DTD, XSLT, XSL-FO, Schema, ebooks, OeB, etc.

**Sievers, Martin**

Klaus-Kordel-Str. 8, 54296 Trier, Germany  
+49 651 4936567-0  
Email: [info \(at\) schoenerpublizieren.com](mailto:info(at)schoenerpublizieren.com)  
Web: <http://www.schoenerpublizieren.com>

As a mathematician with ten years of typesetting experience I offer *T<sub>E</sub>X* and *I<sub>A</sub>T<sub>E</sub>X* services and consulting for the whole academic sector (individuals, universities, publishers) and everybody looking for a high-quality output of his documents.

From setting up entire book projects to last-minute help, from creating individual templates, packages and citation styles (*BIB<sub>T</sub>E<sub>X</sub>*, *biblate<sub>X</sub>*) to typesetting your math, tables or graphics—just contact me with information on your project.

**Sofka, Michael**

8 Providence St.  
Albany, NY 12203  
+1 518 331-3457

Email: [michael.sofka \(at\) gmail.com](mailto:michael.sofka(at)gmail.com)

Skilled, personalized *T<sub>E</sub>X* and *I<sub>A</sub>T<sub>E</sub>X* consulting and programming services.

I offer over 25 years of experience in programming, macro writing, and typesetting books, articles, newsletters, and theses in *T<sub>E</sub>X* and *I<sub>A</sub>T<sub>E</sub>X*: Automated document conversion; Programming in Perl, C, C++ and other languages; Writing and customizing macro packages in *T<sub>E</sub>X* or *I<sub>A</sub>T<sub>E</sub>X*; Generating custom output in PDF, HTML and XML; Data format conversion; Databases.

If you have a specialized *T<sub>E</sub>X* or *I<sub>A</sub>T<sub>E</sub>X* need, or if you are looking for the solution to your typographic problems, contact me. I will be happy to discuss your project.

**Veytsman, Boris**

46871 Antioch Pl.  
Sterling, VA 20164  
+1 703 915-2406  
Email: [borisv \(at\) lk.net](mailto:borisv(at)lk.net)  
Web: <http://www.borisv.lk.net>

*T<sub>E</sub>X* and *I<sub>A</sub>T<sub>E</sub>X* consulting, training and seminars. Integration with databases, automated document preparation, custom *I<sub>A</sub>T<sub>E</sub>X* packages, conversions and much more. I have about seventeen years of experience in *T<sub>E</sub>X* and thirty years of experience in teaching & training. I have authored several packages on CTAN, published papers in *T<sub>E</sub>X* related journals, and conducted several workshops on *T<sub>E</sub>X* and related subjects.

**Young, Lee A.**

127 Kingfisher Lane  
Mills River, NC 28759  
+1 828 435-0525  
Email: [leeayoung \(at\) morrisbb.net](mailto:leeayoung(at)morrisbb.net)  
Web: <http://www.latexcopyeditor.net>  
<http://www.editingscience.net>

Copyediting your *.tex* manuscript for readability and mathematical style by a Harvard Ph.D. Your *.tex* file won't compile? Send it to me for repair. Experience: edited hundreds of ESL journal articles, economics and physics textbooks, scholarly monographs, *I<sub>A</sub>T<sub>E</sub>X* manuscripts for the Physical Review; career as professional, published physicist.

## Calendar

### 2014

Feb 28 – Mar 2	Typography Day 2014, Symbiosis Institute of Design, Pune, India. <a href="http://www.typoday.in">www.typoday.in</a>
Mar 10	TUGboat 35:1, submission deadline (regular issue)
Apr 10–14	TYPO San Francisco, “Rhythm”, Yerba Buena Center for the Arts, San Francisco, California. <a href="http://typotalks.com/sanfrancisco">typotalks.com/sanfrancisco</a>
Apr 11–14	DANTE Frühjahrstagung (25 <sup>th</sup> anniversary of DANTE e.V.) and 50 <sup>th</sup> meeting, Universität Heidelberg, Germany. <a href="http://www.dante.de/events/dante2014.html">www.dante.de/events/dante2014.html</a>
Apr 30 – May 4	EuroBachоТЕХ 2014: EuroTeX and 22 <sup>nd</sup> BachоТЕХ Conference, Bachotek, Poland. <a href="http://www.gust.org.pl/bachotex">www.gust.org.pl/bachotex</a>
Jun 9 – Aug 1	Rare Book School, University of Virginia, Charlottesville, Virginia. Many one-week courses on type, bookmaking, printing, and related topics. <a href="http://www.rarebookschool.org/schedule">www.rarebookschool.org/schedule</a>
Jun 23–26	Book history workshop, École de l'institut d'histoire du livre, Lyon, France. <a href="http://ihl.enssib.fr">ihl.enssib.fr</a>
Jul 2–4	International Society for the History and Theory of Intellectual Property (ISHTIP), 6 <sup>th</sup> Annual Workshop, “The Instability of Intellectual Property”. Uppsala, Sweden. <a href="http://www.ishtip.org/?p=596">www.ishtip.org/?p=596</a>
Jul 8–12	Digital Humanities 2014, Alliance of Digital Humanities Organizations, Lausanne, Switzerland. <a href="http://dh2014.org">dh2014.org</a> , <a href="http://adho.org/conference">adho.org/conference</a>

### TUG 2014 Portland, Oregon.

Jul	The 35 <sup>th</sup> annual meeting of the TЕХ Users Group. <a href="http://tug.org/tug2014">tug.org/tug2014</a>
-----	--

Aug 4–8	Balisage: The Markup Conference, Washington, DC. <a href="http://www.balisage.net">www.balisage.net</a>
Aug 10–14	SIGGRAPH 2014, Vancouver, British Columbia. <a href="http://s2014.siggraph.org">s2014.siggraph.org</a>
Sep	ACM Symposium on Document Engineering, Fort Collins, Colorado. <a href="http://www.documentengineering.org">www.documentengineering.org</a>
Sep 8–9	“Forms and formats: Experimenting with print, 1695–1815”, Centre for the Study of the Book, Bodleian Library, University of Oxford, UK. <a href="http://www.bodleian.ox.ac.uk/csb/community">www.bodleian.ox.ac.uk/csb/community</a>
Sep 8–13	8 <sup>th</sup> International ConTeXt Meeting, Bassenge, Belgium. <a href="http://meeting.contextgarden.net/2014">meeting.contextgarden.net/2014</a>
Sep 14–19	XML Summer School, St Edmund Hall, Oxford University, Oxford, UK. <a href="http://xmlsummerschool.com">xmlsummerschool.com</a>
Sep 17–21	Association Typographique Internationale (ATypI) annual conference, Theme: “Point Counter Point”, Barcelona, Spain. <a href="http://www.atypi.org">www.atypi.org</a>
Sep 17–21	SHARP 2014, “Religions of the Book”, Society for the History of Authorship, Reading & Publishing, Antwerp, Belgium, <a href="http://www.sharpweb.org">www.sharpweb.org</a>
Nov 8–9	The Twelfth International Conference on Books, Publishing, and Libraries, “Disruptive Technologies and the Evolution of Book Publishing and Library Development”, Simmons College, Boston, Massachusetts. <a href="http://booksandpublishing.com/the-conference">booksandpublishing.com/the-conference</a>

### 2015

Mar 19–21	“Publish or Perish? Scientific periodicals from 1665 to the present”. The Royal Society, London, UK. <a href="http://royalsociety.org/events/">royalsociety.org/events/</a>
-----------	---

*Status as of 15 December 2013*

For additional information on TUG-sponsored events listed here, contact the TUG office (+1 503 223-9994, fax: +1 815 301-3568. e-mail: [office@tug.org](mailto:office@tug.org)). For events sponsored by other organizations, please use the contact address provided.

A combined calendar for all user groups is online at [texcalendar.dante.de](http://texcalendar.dante.de).

Other calendars of typographic interest are linked from [tug.org/calendar.html](http://tug.org/calendar.html).

## Introductory

- 268 Clerk Ma and Jie Su / Project Fandol: GPL fonts for Chinese typesetting
  - summary of a new free Chinese font family
- 259 Didier Verna / The incredible tale of the author who didn't want to do the publisher's job, ...
  - writing a book chapter on spec with  $\text{\TeX}$
- 357 Mari Voipio / Entry-level MetaPost 3: Color
  - outlining, filling, choosing colors, linear and circular shading [not a presentation at the conference]

## Intermediate

- 329 Pavneet Arora / TANSU—A workflow for cabinet layout
  - ConTeXt, Asymptote, YAML, and three-dimensional design and costing
- 366 Karl Berry / The treasure chest
  - new CTAN packages, July–December 2013
- 340 Michael Cohen, Yannis Haralambous and Boris Veytsman / The multibibliography package
  - simultaneous bibliographies by appearance, author, year, etc.
- 344 Aleksandra Hankus and Zofia Walczak / L<sup>A</sup>T<sub>E</sub>X and graphics: Basics and packages
  - introduction to graphics usage with different engines, selected add-on packages
- 263 Jason Lewis / How to make a product catalogue that doesn't look like a dissertation
  - practical database generation and layout problems, with recommended packages
- 279 Clerk Ma / Braille fonts in Project Fandol
  - history, relevant standards, and new Chinese Braille fonts
- 302 Andrew Mertz and William Slough / A gentle introduction to Python $\text{\TeX}$ 
  - using Python in documents for computation, plotting, web access, and more
- 281 Ken Nakano and Hajime Kobayashi / Case study: Typesetting old documents of Japan
  - typesetting of Komonjo books published by Shiryo Hensan-jo with p $\text{\TeX}$
- 269 Matthew Skala / Tsukurimashou: A Japanese-language font meta-family
  - motivation and implementation of a METAFONT-based CJK family

## Intermediate Plus

- 332 Nathan Hagen / Bibulous—A drop-in BIB $\text{\TeX}$  replacement based on style templates
  - Unicode-based bibliography implementation in Python using explicit templates
- 293 Norbert Preining /  $\text{\TeX}$  Live Manager's hidden gems: User mode and multiple repository support
  - managing user trees with `tlmgr`, and multiple sources for fetching packages
- 297 Norbert Preining / Redistributing  $\text{\TeX}$  and friends
  - handling  $\text{\TeX}$  Live's configuration in a downstream distro
- 285 Takuji Tanaka / up $\text{\TeX}$ —Unicode version of p $\text{\TeX}$ with CJK extensions
  - comparison of multilingual and other support in up $\text{\TeX}$  with other engines
- 349 Boris Veytsman and Leyla Akhmadeeva / Plots in L<sup>A</sup>T<sub>E</sub>X: Gnuplot, Octave, make
  - work flow for handling regeneration of complex plots
- 313 Lu Wang and Wanmin Liu / Online publishing via pdf2html $\text{\TeX}$ 
  - handling  $\text{\TeX}$  Live's configuration in a downstream distro

## Advanced

- 325 Shinsaku Fujita / The X<sup>M</sup> $\text{\TeX}$  system for publishing interdisciplinary chemistry/mathematics books
  - basic usage and history of X<sup>M</sup> $\text{\TeX}$ , an advanced chemical typesetting system
- 289 John Plaice / Typesetting and layout in multiple directions—Proposed solution
  - separating writing style from box direction in full generality

## Contents of other $\text{\TeX}$ journals

- 363 ConTeXt Proceedings, 6th meeting (2012); MAPS 44 (2013); Die  $\text{\TeX}$ nische Komödie 3–4/2013

## Reports and notices

- 246 TUG 2013 conference information
- 250 TUG 2013 conference program
- 252 Norbert Preining / TUG 2013 in Tokyo
- 360 TUG 2013 abstracts (Cho, Hagen, Hakuta, Maeda & Kaneko, Minoda, Mittelbach, Moore, Shikano, Takata, Terada, Verna-tcl, Wetmore, Yabe)
- 367 Denis Bitouzé / In memoriam: Jean-Pierre Drucbert (1947–2009)
- 368 Dave Walden / Book review: *Essential Knuth*
  - an extensive interview focused on Knuth's computer science achievements
- 369 Clerk Ma / Book review: *Introduction to L<sup>A</sup>T<sub>E</sub>X*, Leo Liu
  - a book on introduction to typesetting Chinese and Japanese with L<sup>A</sup>T<sub>E</sub>X
- 370 Institutional members
- 370  $\text{\TeX}$  consulting and production services
- 372 Calendar