Typesetting with Kanji

Norbert Preining
JAIST, Japan
TEX Live Team, Debian Developer

TUG 2012, Boston, 2012-08-16

Layout of the talk

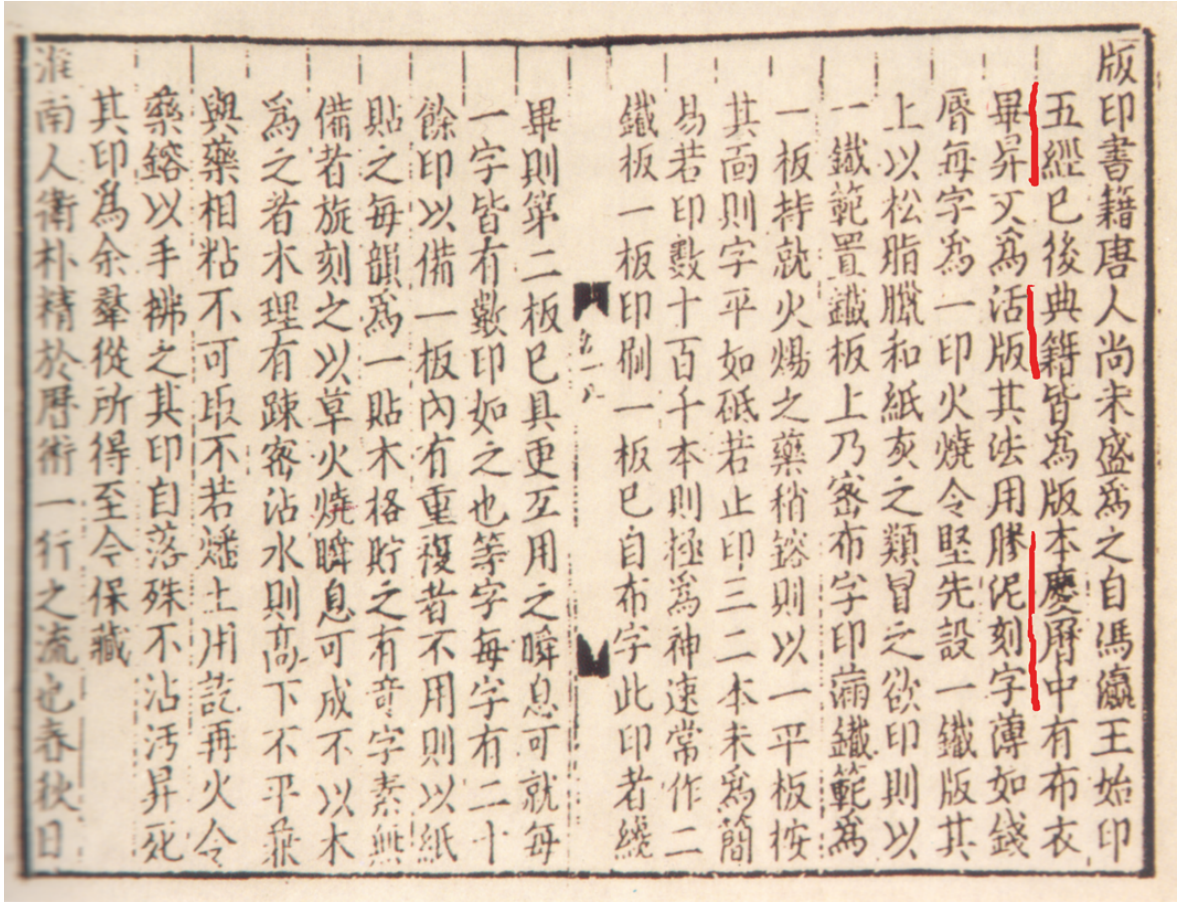
Specialities of Japanese Typography

- the different writing systems in Japan
- mixing different writing systems
- grid setting and spacing rules

Implementation in pTEX and friends

- History of the pTEX family
- Character classes, spacing
- pTEX and TEX Live
- Problems and future developments

The ideal of aligned typesetting



版印書籍唐人尚未盛爲之自馮瀛王始印
五經已後典籍皆爲版本慶曆中有布衣
畢昇又爲活版其法用膠泥刻字薄如錢
脣每字爲一印火燒令堅先設一鐵版其
上以松脂臘和紙灰之類冒之欲印則以
一鐵範置鐵板上乃密布字印滿鐵範爲
一板持就火煬之藥稍鎔則以一平板按
其面則字平如砥若止印三二本未爲簡
易若印數十百千本則極爲神速常作二
鐵板一板印刷一板已自布字此印者纔
畢則第二板已具更互用之瞬息可就每
一字皆有數印如之也等字每字有二十
餘印以備一板內有重複者不用則以紙
貼之每韻爲一貼木格貯之有奇字素無
備者旋刻之以草火燒瞬息可成不以木
爲之者木理有疎密沾水則高下不平兼
與藥相粘不可取不若燒土用訖再火令
藥鎔以手拂之其印自落殊不沾汚昇死
其印爲余輩從所得至今保藏
淮南人衛朴精於曆術一行之流也春秋日

A page of *Dream Pool Jotting* by Shen Kuo,
the earliest record of making movable types

… and reality

その上、今日の空模様も少からず、この平安朝の下人の Sentimentalisme に影響した。申（さる）の刻（こく）下（さが）りからふり出した雨は、いまだに上るけしきがない。そこで、下人は、何をおいても差当り明日（あす）の暮しをどうにかしようとして——云わばどうにもならない事を、どうにかしようとして、とりとめもない考えをたどりながら、さっきから朱雀大路にふる雨の音を、聞くともなく聞いていたのである。

Reasons for unalignedness

- different writing systems
- punctuation symbols
- special rules (orphans, widows, …)
- different typographic development (hanging punctuation, indentation)

Different writing systems in Japan

ideographic	中国日本鬱
hiragana	あかげじゅぎょ
katakana	アカゲシュギョ
romaji	Hello World

Example for horizontal type setting

その上、今日の空模様も少からず、この平安朝の下人の Sentimentalisme に影響した。申（さる）の刻（こく）下（さが）りからふり出した雨は、いまだに上るけしきがない。そこで、下人は、何をおいても差当り明日（あす）の暮しをどうにかしようとして——云わばどうにもならない事を、どうにかしようとして、とりとめもない考えをたどりながら、さっきから朱雀大路にふる雨の音を、聞くともなく聞いていたのである。

Example for vertical type setting

その上、今日の空模様も少からず、この平安朝の下人の Sentimentalisme に影響した。申（さる）の刻（こく）下（さが）りからふり出した雨は、いまだに上るけしきがない。そこで、下人は、何をおいても差当り明日（あす）の暮しをどうにかしようとして——云わばどうにもならない事を、どうにかしようとして、とりとめもない考えをたどりながら、さっきから朱雀大路にふる雨の音を、聞くともなく聞いていたのである。

Mixing roman letters into vertical text

Mixing roman letters into vertical text

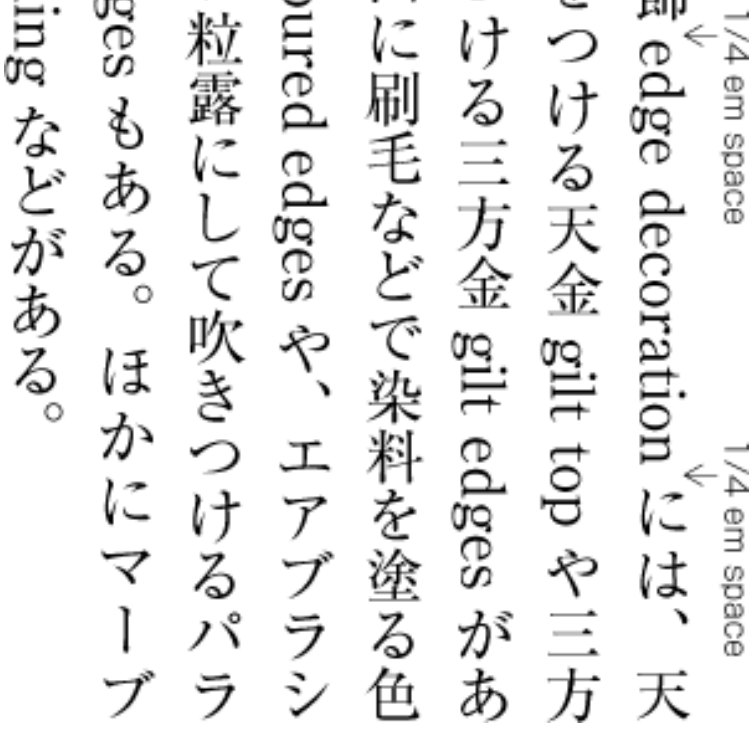
Mixing roman letters into vertical text

Mixing roman letters – the default

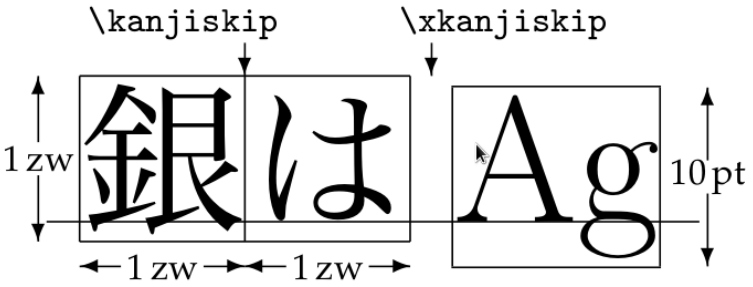
1/4 em space

小口装飾 edge decoration には、天に金箔をつける天金 gilt top や三方に金をつける三方金 gilt edges がある。小口に刷毛などで染料を塗る色染め coloured edges や、エアブラシであらい粒露にして吹きつけるパラ frost edges もある。ほかにマーブル marbling などがある。

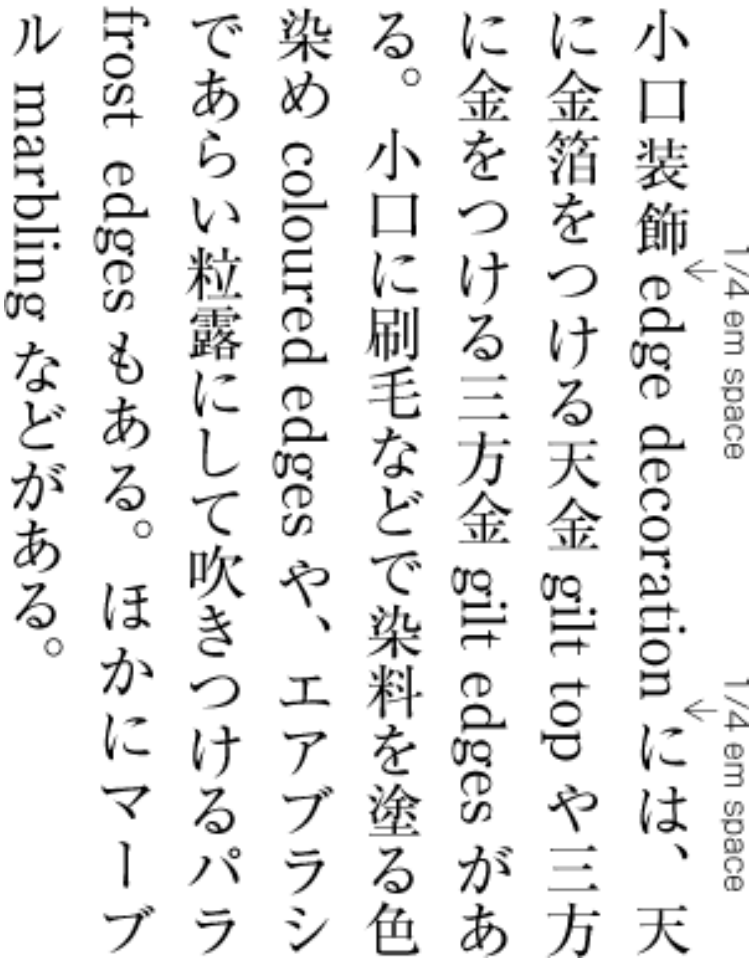
Mixing roman letters – the default



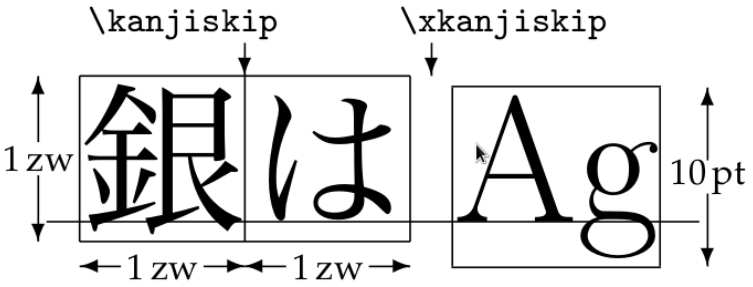
小口装飾 edge decoration には、天に金箔をつける天金 gilt top や三方に金をつける三方金 gilt edges がある。小口に刷毛などで染料を塗る色染め coloured edges や、エアブラシであらい粒露にして吹きつけるパラ frost edges もある。ほかにマーブル marbling などがある。



Mixing roman letters – the default



小口装飾 edge decoration には、天に金箔をつける天金 gilt top や三方に金をつける三方金 gilt edges がある。小口に刷毛などで染料を塗る色染め coloured edges や、エアブラシであらい粒露にして吹きつけるパラ frost edges もある。ほかにマーブル marbling などがある。

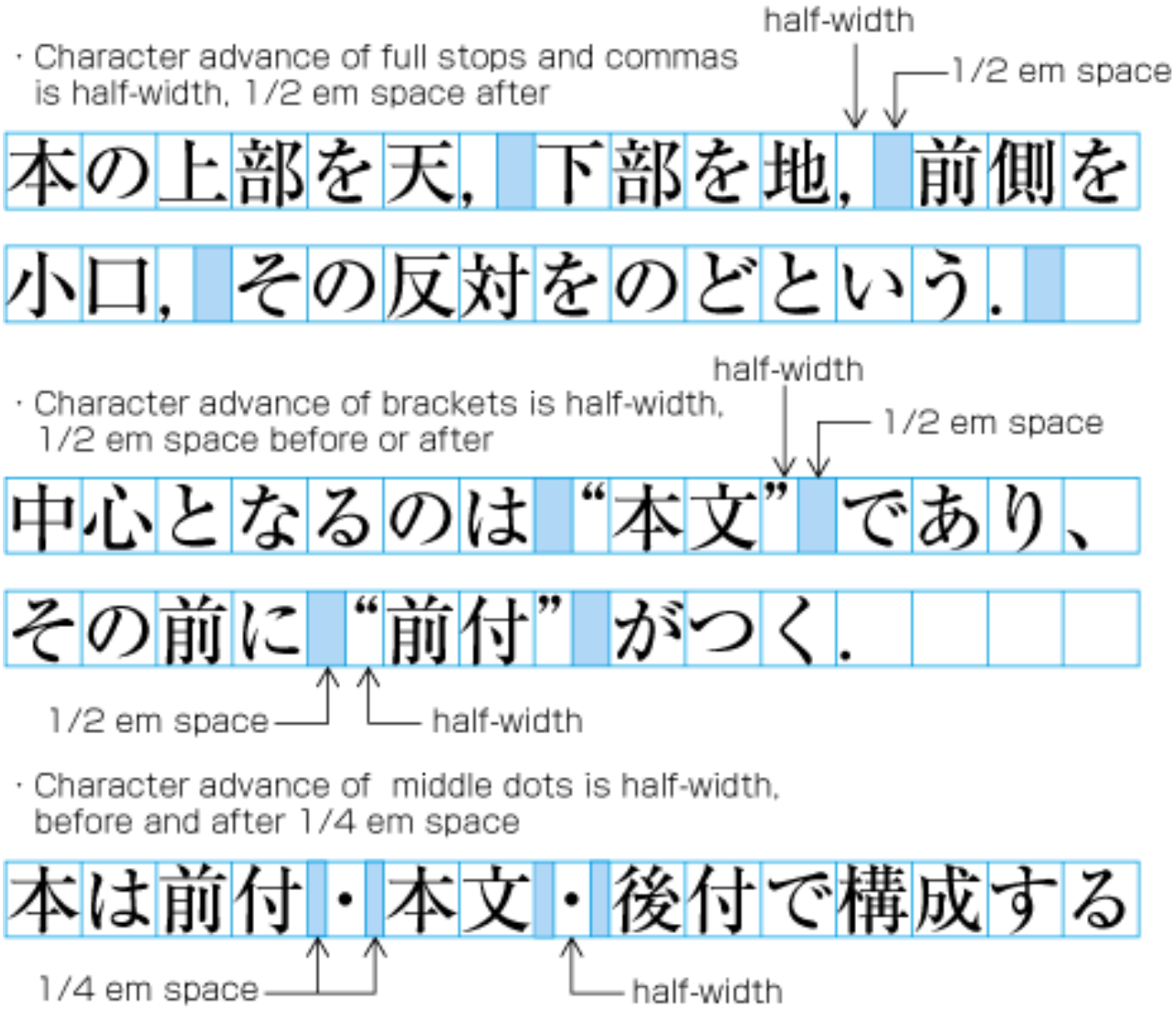


many other options, eg.

日本安全中国

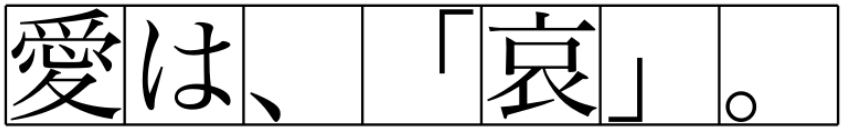
安全第２目全

Rules for punctuation

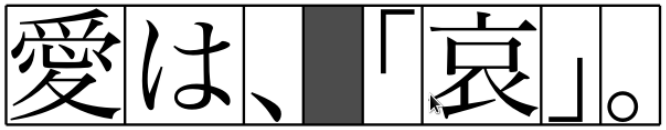


What happens with consecutive punctuations?

text editor – fixed width

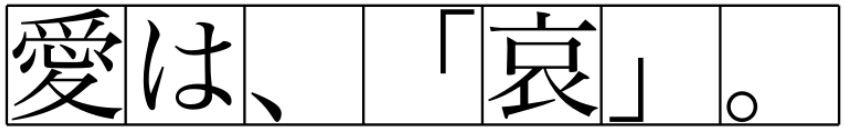


correct output

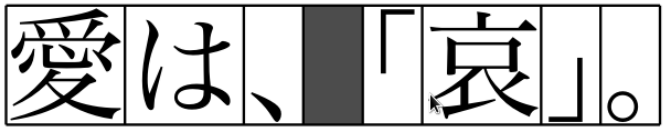


What happens with consecutive punctuations?

text editor – fixed width



correct output

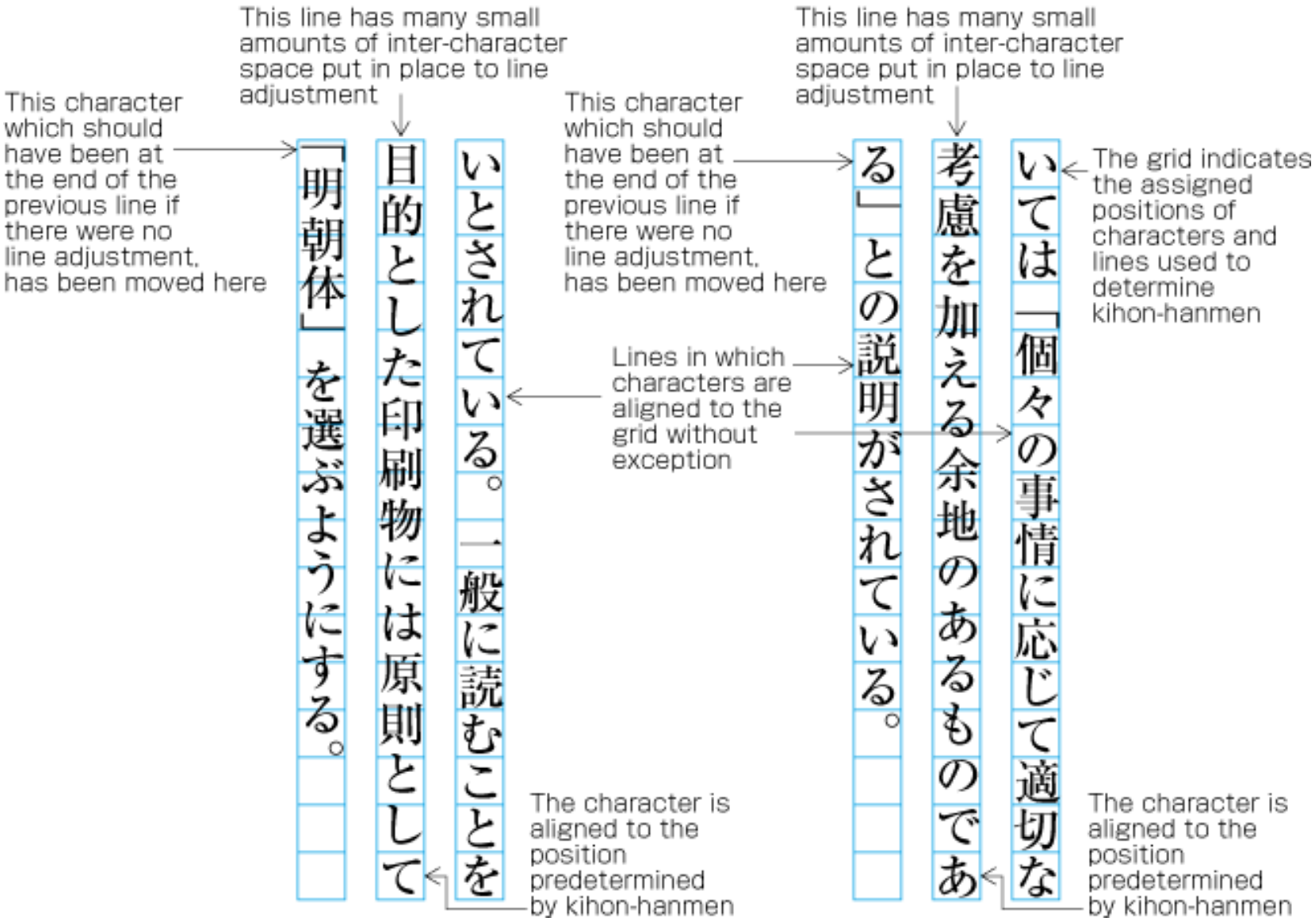


10 kanjis: 安日本中国日本中国全

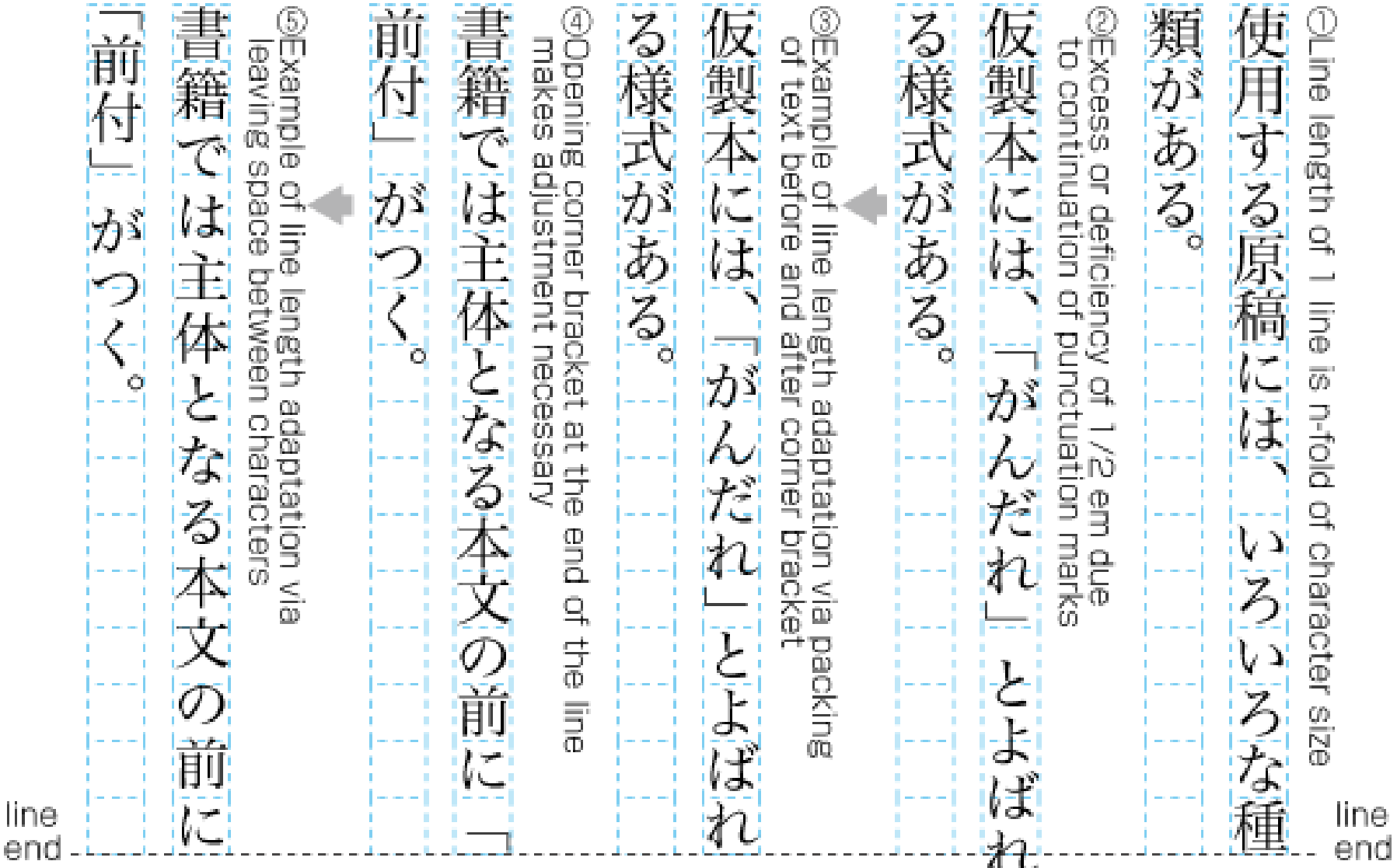
no special rule: 安「日本」「中国」全

special rule: 安「日本」「中国」全

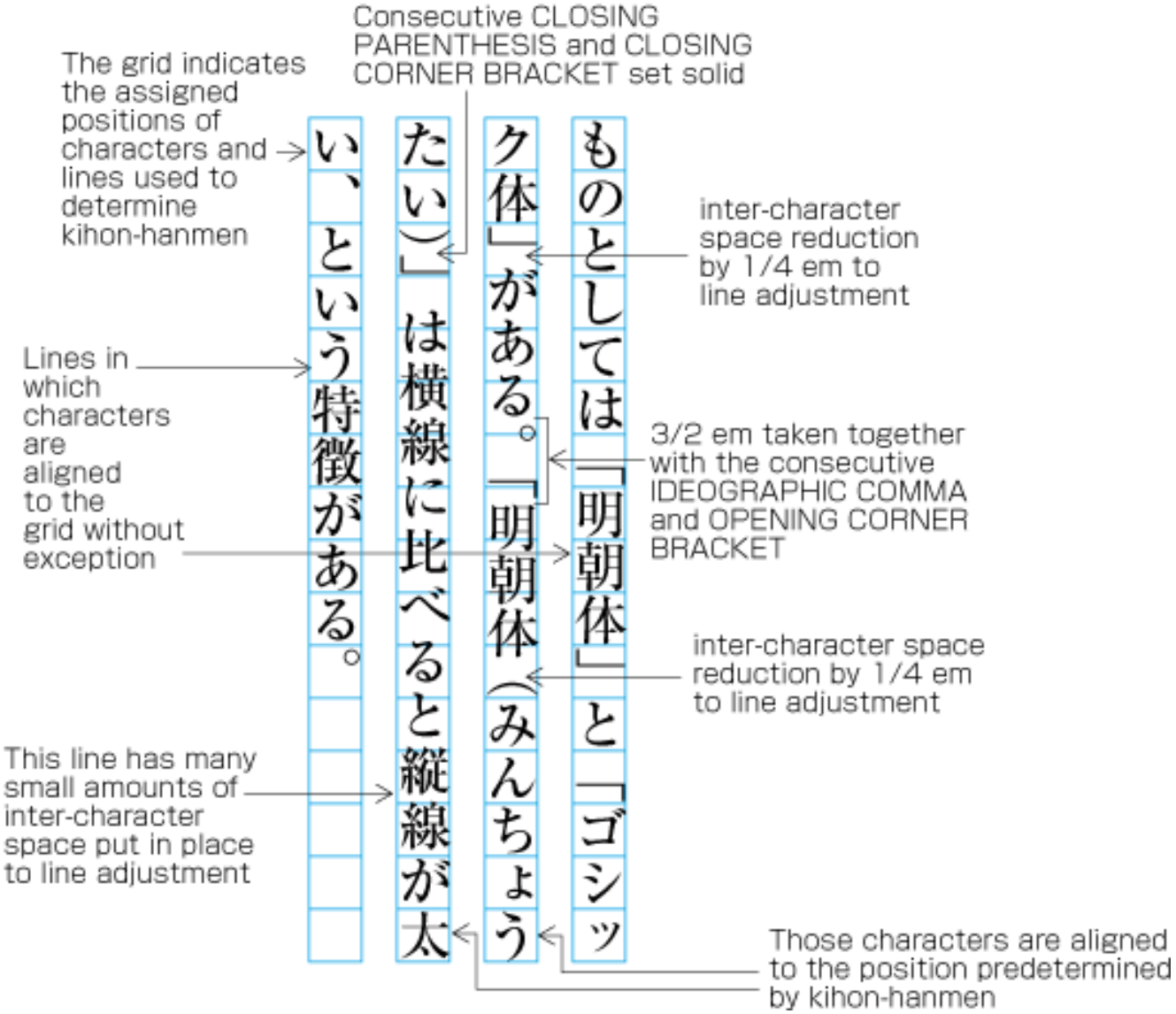
Glyph widows and orphans



Line adjustments



Another example of line adjustments



Implementation in pTEX and friends

File Edit Options Buffers Tools Help

Save Undo

```
  incr(p); if p>255 then p:=0;
  until s=p;
  p:=no_entry;
  end
else
  begin repeat
  if kinsoku_type(p)=0 then goto done1;
  if kinsoku_code(p)=c then goto done;
  incr(p); if p>255 then p:=0;
  until s=p;
done1: p:=no_entry;
  end;
done: get_kinsoku_pos:=p;
end;

@ @<Assignments@>=
assign_kinsoku:
begin p:=cur_chr; scan_int; n:=cur_val; scan_optional_equals; scan_int;
if is_char_ascii(n) or is_char_kanji(n) then
  begin j:=get_kinsoku_pos(tokanji(n),new_pos);
  if j=no_entry then
    begin print_err("KINSOKU table is full!!");
    help1("I'm skip this control sequences.");@/
    error; return;
    end;
  if (p=pre_break_penalty_code)or(p=post_break_penalty_code) then
    begin define(kinsoku_base+j,p,tokanji(n));
    word_define(kinsoku_penalty_base+j,cur_val);
    end
  else confusion("kinsoku");
@:this can't happen kinsoku}{\quad kinsoku@>
  end
else
  begin print_err("Invalid KANJI code for ");
  if p=pre_break_penalty_code then print("pre")
  else if p=post_break_penalty_code then print("post")
  else print_char("?");
  print("breakpenalty (");print_hex(n);print_char(")");
@.Invalid KANJI code@>
  help1("I'm skip this control sequences.");@/
  error; return;
  end;
end;
```

1:--- ptex-base.ch 91% L6173 SVN-26311 (Fundamental)

```
@ Following procedure |adjust_hlist| inserts \.{\\xkanjiskip} between
2byte-char and 1byte-char in hlist which pointed |p|.
Note that the skip is inserted into a place where too difficult to decide
whether inserting or not (i.e, before penalty, after penalty).

If |pf| is true then insert |jchr_widow_penalty| that is penalty for
creating a widow KANJI character line.

@d no_skip=0
@d after_schar=1 {denote after single byte character}
@d after_wchar=2 {denote after double bytes character}

@<Declare procedures needed in |hlist_out|, |vlist_out|@>=
procedure adjust_hlist(p:pointer;pf:boolean);
label exit;
var q,s,t,u,v,x,z:pointer;
  i,k:halfword;
  a: pointer; { temporary pointer for accent }
  insert_skip:no_skip..after_wchar;
  cx:KANJI_code; {temporaly register for KANJI character}
  ax:ASCII_code; {temporaly register for ASCII character}
  do_ins:boolean; {for inserting |xkanji_skip| into prevous (or after) KANJI
}
begin if link(p)=null then goto exit;
if auto_spacing>0 then
  begin delete_glue_ref(space_ptr(p)); space_ptr(p):=kanji_skip;
  add_glue_ref(kanji_skip);
  end;
if auto_xspacing>0 then
  begin delete_glue_ref(xspace_ptr(p)); xspace_ptr(p):=xkanji_skip;
  add_glue_ref(xkanji_skip);
  end;
u:=space_ptr(p); add_glue_ref(u);
s:=xspace_ptr(p); add_glue_ref(s);
if not is_char_node(link(p)) {p1.0.9d}
    and(type(link(p))=glue_node)and(subtype(link(p))=jfm_skip+1) then
  begin v:=link(p); link(p):=link(v);
  fast_delete_glue_ref(glue_ptr(v)); free_node(v,small_node_size);
  end;
i:=0; insert_skip:=no_skip; p:=link(p); v:=p; q:=p;
while p<>null do
  begin if is_char_node(p) then
    begin repeat @<Insert a space around the character |p|@>;
```

1:--- ptex-base.ch 93% L6350 SVN-26311 (Fundamental)

Parts of the pTEX code

History of pTEX and friends (thanks to Okumura-sensei)

- 1987 NTT jTEX: extension of TEX with subfont splitting scheme
- 1987 ASCII Nihongo TEX, ASCII Corporation: multibyte extension
- 1990 extension for vertical typesetting: pTEX
- 1993 JIS X 4051: Japanese Industrial Standard for typesetting
- 1995 revised version of pTEX based on TEX 3.0, pLATEX2e
- 1997 new font metrics conforming to JIS X 4051 (Hajime Kobayashi)
- 1997– new document classes conforming to JIS X 4051 (Haruhiko Okumura)

History cont.

- 2003 `otf` package: accessing variants of characters via CID-keys that are not representable in Unicode due to Han-unification (Shuzaburo Saito)
- 2006 ptetex (and later ptexlive) (Nobuyuki Tsuchimura): complete setup of teTEX (TEX Live) for Japanese, extension to handle some Unicode UTF 8
- 2007– upTEX (Takuji Tanaka): pTEX's internal encoding is EUC, upTEX supports reading UTF-8
- 2008– ε-pTEX (Hironori Kitagawa): LATEX required ε-TEX-extensions, so merging pTEX and ε-TEX became necessary
- 2008– ε-upTEX (Hironori Kitagawa): double merge of upTEX and ε-pTEX

What is pTEX (and friends)

- 16bit extension (various input and internal encodings, depending on engine)
- special Japanese font metrics support:
  - huge number of glyphs
  - only a few different possible dimensions
  - special rules for mixing (kerning etc)
  - variants for vertical and horizontal typesetting
- inter-glyph spacing, spacing between non-Japanese and Japanese glyphs (see first part!)
- line-breaking (space at end of line is mostly ignored!)

Character classes and spacing

All Japanese glyphs are distributed into seven chartype classes:

CHARTYPE	Width	Examples
0	1zw	かなカナ漢字
1	0.5zw	「《“
2	0.5zw	、，」》”
3	0.5zw	・：；
4	0.5zw	。．
5	1zw	—…
6	1zw	？！

Spacing before and after combinations of character classes

	0	1	2	3	4	5	6
0		½ – ½		¼ – ¼			
1				¼ – ¼			
2	½ – ½	½ – ½		¼ – ¼		½ – ½	½ – ½
3	¼ – ¼	¼ – ¼	¼ – ¼	½ – ¼	¼ – ¼	¼ – ¼	¼ – ¼
4	½ – 0	½ – 0		¾ – ¼		½ – 0	½ – 0
5		½ – ½		¼ – ¼		0	
6	½ – ½	½ – ½		¼ – ¼			

Packaging history

- ptetex, ptexlive: Haruhiko Okumura, Nobuyuki Tsuchimura, Hironori Kitagawa 2004–2011
- TEX Live

2010.3	pTEX enters default TEX Live development code
2010.9	TEX Live 2010 release
2011.1-4	ε-pTEX enters development code
2011.7	TEX Live 2011 release
2011.8	upTEX enters development code
2011.10	Japanese TEX User Meeting
2011.10-	missing pieces from ptexlive/ptetex are moved into TEX Live
2012.7	TEX Live 2012 release, ptexlive and ptetex superseeded (besides pxdvi and pmetapost)

Font support – current status

problem field between huge amount of glyphs and the same "boxes"

- actual available fonts are only a few, but increasing
- traditionally only two: Mincho and Gothic
- same font metrics for all the fonts – just copies, but due to big size inconvenient
- embedding currently controlled in the output driver dvipdmf(x) (pTEX et al only supports dvi output!)
- with special packages (otf) one can use up to 7 (?) fonts (sic!) (mincho/gothic regular, bold, extra, plus rounded)
- supported font families in TEX Live: IPA, IPAex, Kozuka, Hiragino, Morisawa (package `jfontmaps`)
- mixture of fonts currently not supported directly

Font support – current status

problem field between huge amount of glyphs and the same "boxes"

- actual available fonts are only a few, but increasing
- traditionally only two: Mincho and Gothic
- same font metrics for all the fonts – just copies, but due to big size inconvenient
- embedding currently controlled in the output driver dvipdmf(x) (pTEX et al only supports dvi output!)
- with special packages (otf) one can use up to 7 (?) fonts (sic!) (mincho/gothic regular, bold, extra, plus rounded)
- supported font families in TEX Live: IPA, IPAex, Kozuka, Hiragino, Morisawa (package `jfontmaps`)
- mixture of fonts currently not supported directly

Font support – current status

problem field between huge amount of glyphs and the same "boxes"

- actual available fonts are only a few, but increasing
- traditionally only two: Mincho and Gothic
- same font metrics for all the fonts – just copies, but due to big size inconvenient
- embedding currently controlled in the output driver dvipdmf(x) (pTEX et al only supports dvi output!)
- with special packages (otf) one can use up to 7 (?) fonts (sic!) (mincho/gothic regular, bold, extra, plus rounded)
- supported font families in TEX Live: IPA, IPAex, Kozuka, Hiragino, Morisawa (package `jfontmaps`)
- mixture of fonts currently not supported directly

Font support – current status

problem field between huge amount of glyphs and the same "boxes"

- actual available fonts are only a few, but increasing
- traditionally only two: Mincho and Gothic
- same font metrics for all the fonts – just copies, but due to big size inconvenient
- embedding currently controlled in the output driver dvipdmf(x) (pTEX et al only supports dvi output!)
- with special packages (otf) one can use up to 7 (?) fonts (sic!) (mincho/gothic regular, bold, extra, plus rounded)
- supported font families in TEX Live: IPA, IPAex, Kozuka, Hiragino, Morisawa (package `jfontmaps`)
- mixture of fonts currently not supported directly

Font support – current status

problem field between huge amount of glyphs and the same "boxes"

- actual available fonts are only a few, but increasing
- traditionally only two: Mincho and Gothic
- same font metrics for all the fonts – just copies, but due to big size inconvenient
- embedding currently controlled in the output driver dvipdmf(x) (pTEX et al only supports dvi output!)
- with special packages (otf) one can use up to 7 (?) fonts (sic!) (mincho/gothic regular, bold, extra, plus rounded)
- supported font families in TEX Live: IPA, IPAex, Kozuka, Hiragino, Morisawa (package `jfontmaps`)
- mixture of fonts currently not supported directly

Font support – current status

problem field between huge amount of glyphs and the same "boxes"

- actual available fonts are only a few, but increasing
- traditionally only two: Mincho and Gothic
- same font metrics for all the fonts – just copies, but due to big size inconvenient
- embedding currently controlled in the output driver dvipdmf(x) (pTEX et al only supports dvi output!)
- with special packages (otf) one can use up to 7 (?) fonts (sic!) (mincho/gothic regular, bold, extra, plus rounded)
- supported font families in TEX Live: IPA, IPAex, Kozuka, Hiragino, Morisawa (package `jfontmaps`)
- mixture of fonts currently not supported directly

Font support – kanji embedding in TEX Live

- new `updmap.cfg` file options `kanjiEmbed`, `kanjiVariant`, and `pxdviUse`, possible (sensible) values are `noEmbed`, `ipa`, `ipaex`, `kozuka`, `morisawa`, `hiragino`
- the value of `kanjiEmbed` replaces the string `@kanjiEmbed@` in the names of map files
- base packages (ptex, uptex, ...) ship the respective 'virtual map files'
  `ptex-@kanjiEmbed@@kanjiVariant@.map`
  `uptex-@kanjiEmbed@@kanjiVariant@.map`
  `otf-@kanjiEmbed@.map`
  `otf-up-@kanjiEmbed@.map`
- replacement of the keys is done during `updmap` run, includes map files like `ptex-morisawa.map`, which ensures that `dvipdfmx` embeds the Morisawa fonts.
- more details: `http://tug.org/texlive/updmap-kanji.html`

Font support – kanji embedding in TEX Live

- new `updmap.cfg` file options `kanjiEmbed`, `kanjiVariant`, and `pxdviUse`, possible (sensible) values are `noEmbed`, `ipa`, `ipaex`, `kozuka`, `morisawa`, `hiragino`
- the value of `kanjiEmbed` replaces the string `@kanjiEmbed@` in the names of map files
- base packages (ptex, uptex, ...) ship the respective 'virtual map files'
  `ptex-@kanjiEmbed@@kanjiVariant@.map`
  `uptex-@kanjiEmbed@@kanjiVariant@.map`
  `otf-@kanjiEmbed@.map`
  `otf-up-@kanjiEmbed@.map`
- replacement of the keys is done during `updmap` run, includes map files like `ptex-morisawa.map`, which ensures that `dvipdfmx` embeds the Morisawa fonts.
- more details: `http://tug.org/texlive/updmap-kanji.html`

Font support – kanji embedding in TEX Live

- new `updmap.cfg` file options `kanjiEmbed`, `kanjiVariant`, and `pxdviUse`, possible (sensible) values are `noEmbed`, `ipa`, `ipaex`, `kozuka`, `morisawa`, `hiragino`
- the value of `kanjiEmbed` replaces the string `@kanjiEmbed@` in the names of map files
- base packages (ptex, uptex, ...) ship the respective 'virtual map files'
  `ptex-@kanjiEmbed@@kanjiVariant@.map`
  `uptex-@kanjiEmbed@@kanjiVariant@.map`
  `otf-@kanjiEmbed@.map`
  `otf-up-@kanjiEmbed@.map`
- replacement of the keys is done during `updmap` run, includes map files like `ptex-morisawa.map`, which ensures that `dvipdfmx` embeds the Morisawa fonts.
- more details: `http://tug.org/texlive/updmap-kanji.html`

Font support – kanji embedding in TEX Live

- new `updmap.cfg` file options `kanjiEmbed`, `kanjiVariant`, and `pxdviUse`, possible (sensible) values are `noEmbed`, `ipa`, `ipaex`, `kozuka`, `morisawa`, `hiragino`
- the value of `kanjiEmbed` replaces the string `@kanjiEmbed@` in the names of map files
- base packages (ptex, uptex, ...) ship the respective 'virtual map files'
  `ptex-@kanjiEmbed@@kanjiVariant@.map`
  `uptex-@kanjiEmbed@@kanjiVariant@.map`
  `otf-@kanjiEmbed@.map`
  `otf-up-@kanjiEmbed@.map`
- replacement of the keys is done during `updmap` run, includes map files like `ptex-morisawa.map`, which ensures that `dvipdfmx` embeds the Morisawa fonts.
- more details: `http://tug.org/texlive/updmap-kanji.html`

Font support – kanji embedding in TEX Live

- new `updmap.cfg` file options `kanjiEmbed`, `kanjiVariant`, and `pxdviUse`, possible (sensible) values are `noEmbed`, `ipa`, `ipaex`, `kozuka`, `morisawa`, `hiragino`
- the value of `kanjiEmbed` replaces the string `@kanjiEmbed@` in the names of map files
- base packages (ptex, uptex, ...) ship the respective 'virtual map files'
  `ptex-@kanjiEmbed@@kanjiVariant@.map`
  `uptex-@kanjiEmbed@@kanjiVariant@.map`
  `otf-@kanjiEmbed@.map`
  `otf-up-@kanjiEmbed@.map`
- replacement of the keys is done during `updmap` run, includes map files like `ptex-morisawa.map`, which ensures that `dvipdfmx` embeds the Morisawa fonts.
- more details: `http://tug.org/texlive/updmap-kanji.html`

Where to go from here

Current problems

- font support
- pdf output – engine renewal

Intermediate solution

- font support: not completely clear where to go: embedding map lines in the dvi file (pxchfon, zhmCJK approach – does not solve the number of accessible fonts, but makes mixing easier)

Real solution on the horizont

- LuaTEX-ja – macro package for typesetting Japanese with LuaTEX (LuaTEX-ja Team 2011–)
  highly influenced by pTEX (and thus JIS X 4051 standard), uses lua callbacks in luatex, gives pdf output, support for otf fonts, ...

Where to go from here

Current problems

- font support
- pdf output – engine renewal

Intermediate solution

- font support: not completely clear where to go: embedding map lines in the dvi file (pxchfon, zhmCJK approach – does not solve the number of accessible fonts, but makes mixing easier)

Real solution on the horizont

- LuaTEX-ja – macro package for typesetting Japanese with LuaTEX (LuaTEX-ja Team 2011–)
  highly influenced by pTEX (and thus JIS X 4051 standard), uses lua callbacks in luatex, gives pdf output, support for otf fonts, ...

Where to go from here

Current problems

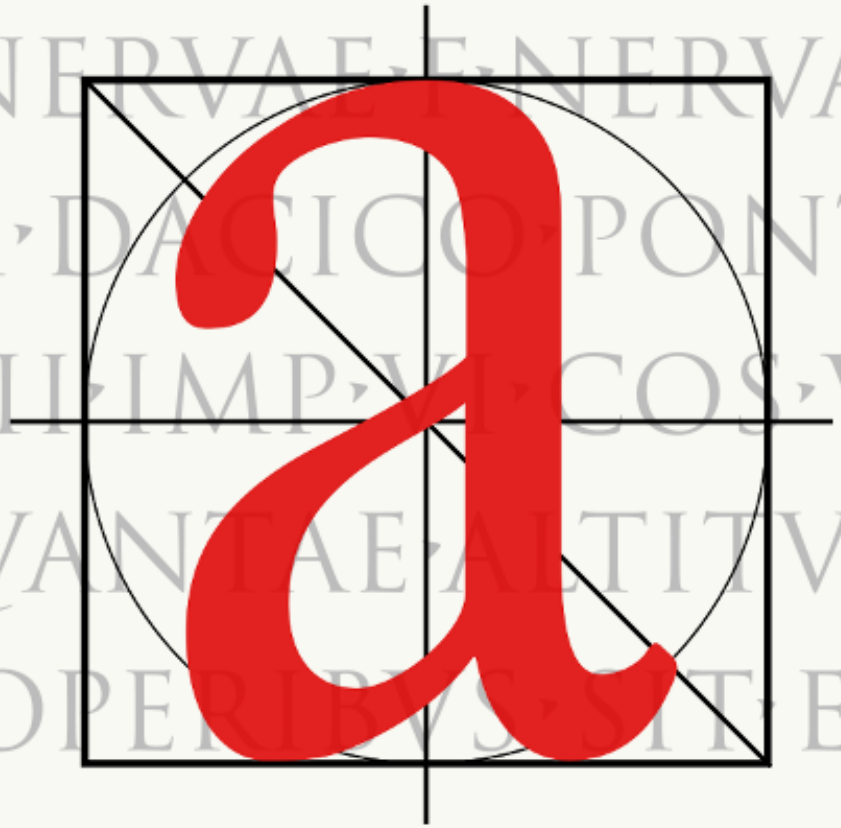
- font support
- pdf output – engine renewal

Intermediate solution

- font support: not completely clear where to go: embedding map lines in the dvi file (pxchfon, zhmCJK approach – does not solve the number of accessible fonts, but makes mixing easier)

Real solution on the horizont

- LuaTEX-ja – macro package for typesetting Japanese with LuaTEX (LuaTEX-ja Team 2011–)
  highly influenced by pTEX (and thus JIS X 4051 standard), uses lua callbacks in luatex, gives pdf output, support for otf fonts, ...



Thanks