フォントのしくみ

第３回　DTPの勉強会
狩野宏樹（株式会社イワタ）

全体の流れ

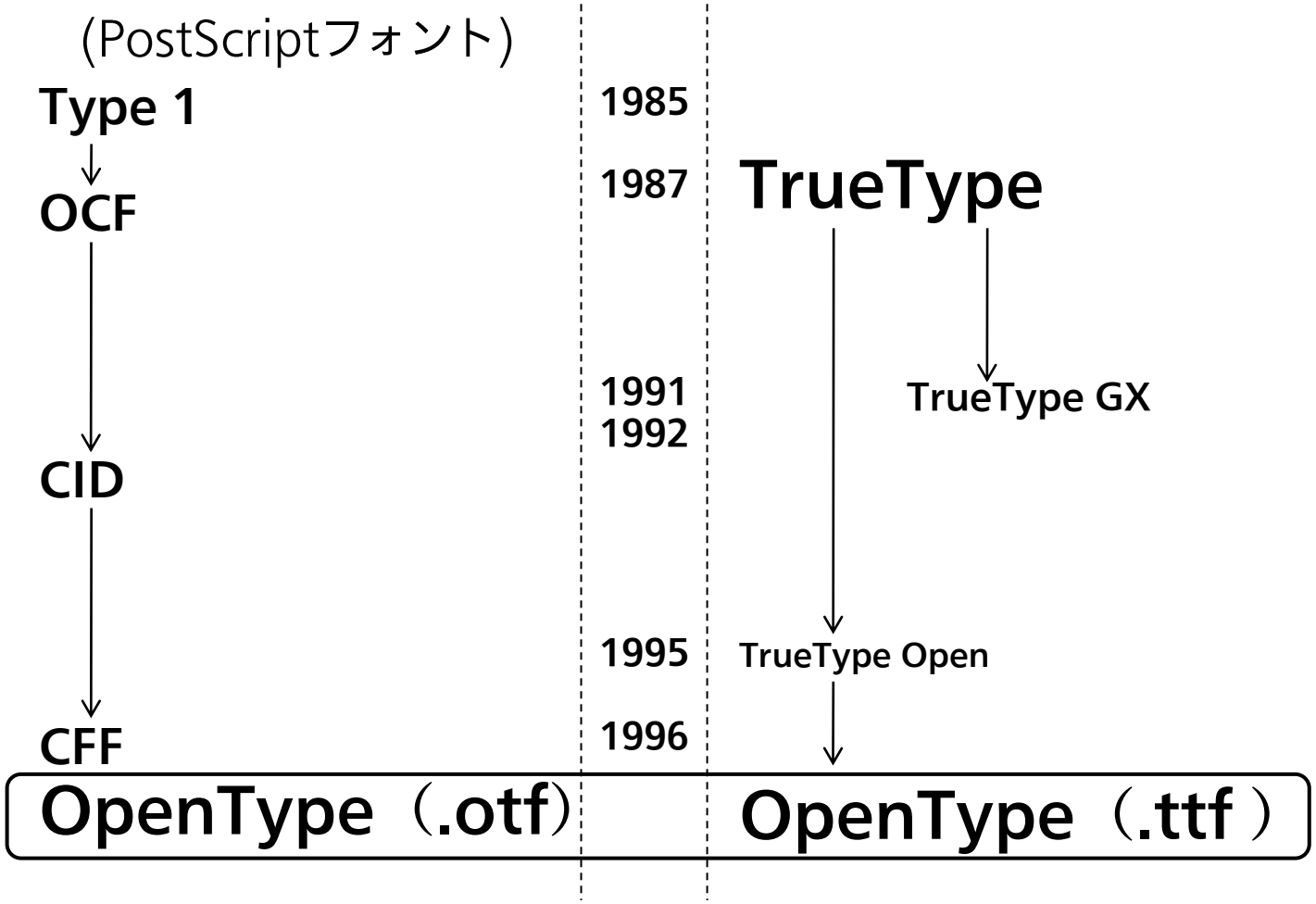
- 2種類の「OTF」について
- フォントデータの中身の概略
- 複雑なテキスト表示処理について

- フォントテーブルの名前は基本的に覚えなくてよい
  - glyf / CFF
  - GSUB, GPOS

フォントのフォーマットについて

～「OpenType」って何？～

いろいろなフォントフォーマット

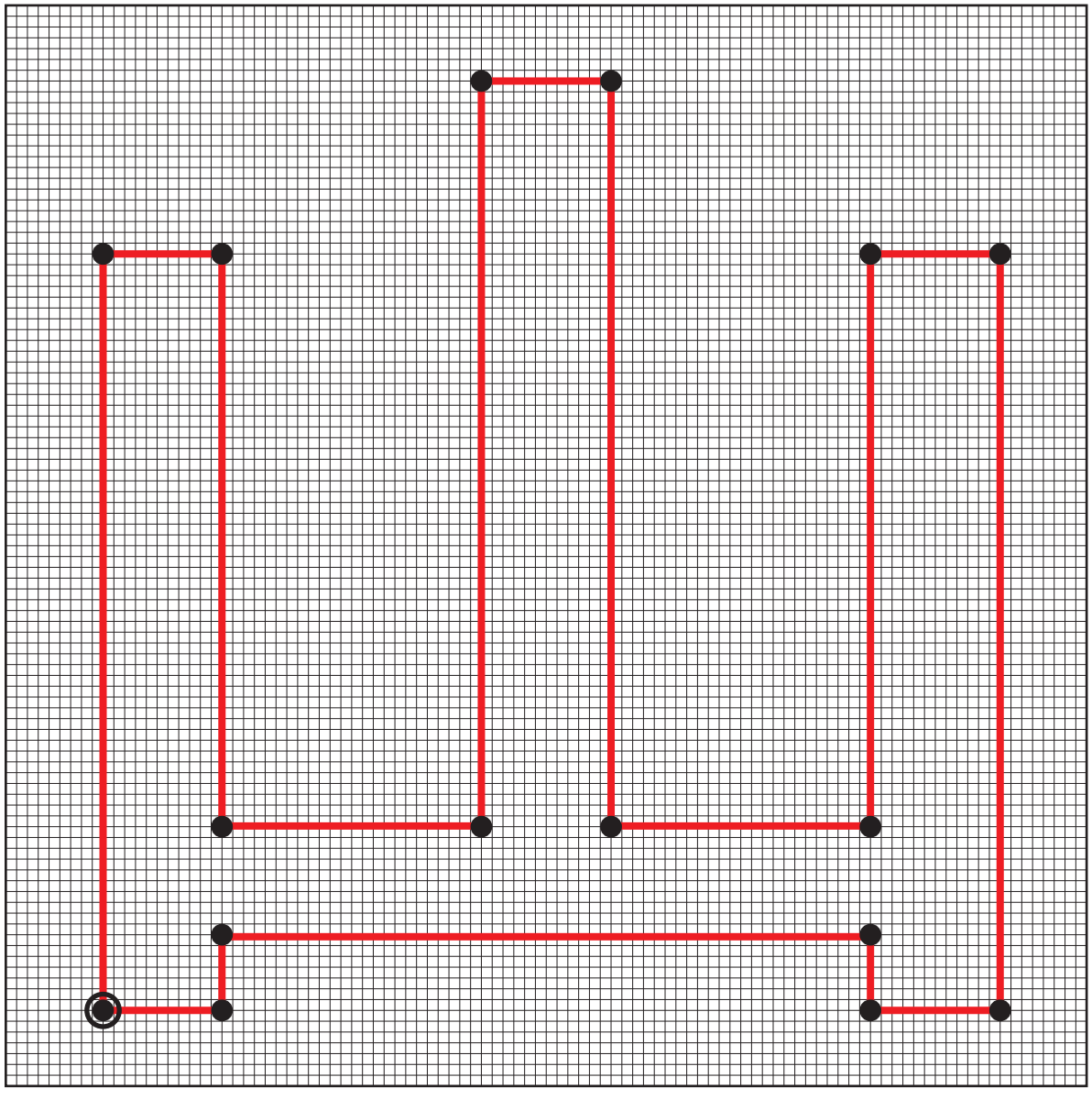


※年号は、製品でなく技術の発表年（見た資料の中で古いのを採用）

フォントのアウトラインには２種類ある

- PostScriptフォント
  - 仕様はAdobeが作っている。
  - Type1, CID, OTF…どれも基礎は同じ
- TrueType
  - Adobeに対抗してAppleとMSが手を組んだ
  - 仕様を読むといかにも「技術者が作った」感じがする

- どちらもBézier曲線
  - PSは3次Bézier、TTFは2次Bézier
    - 日本語文献では「TTF=Bスプライン曲線」の記述多数

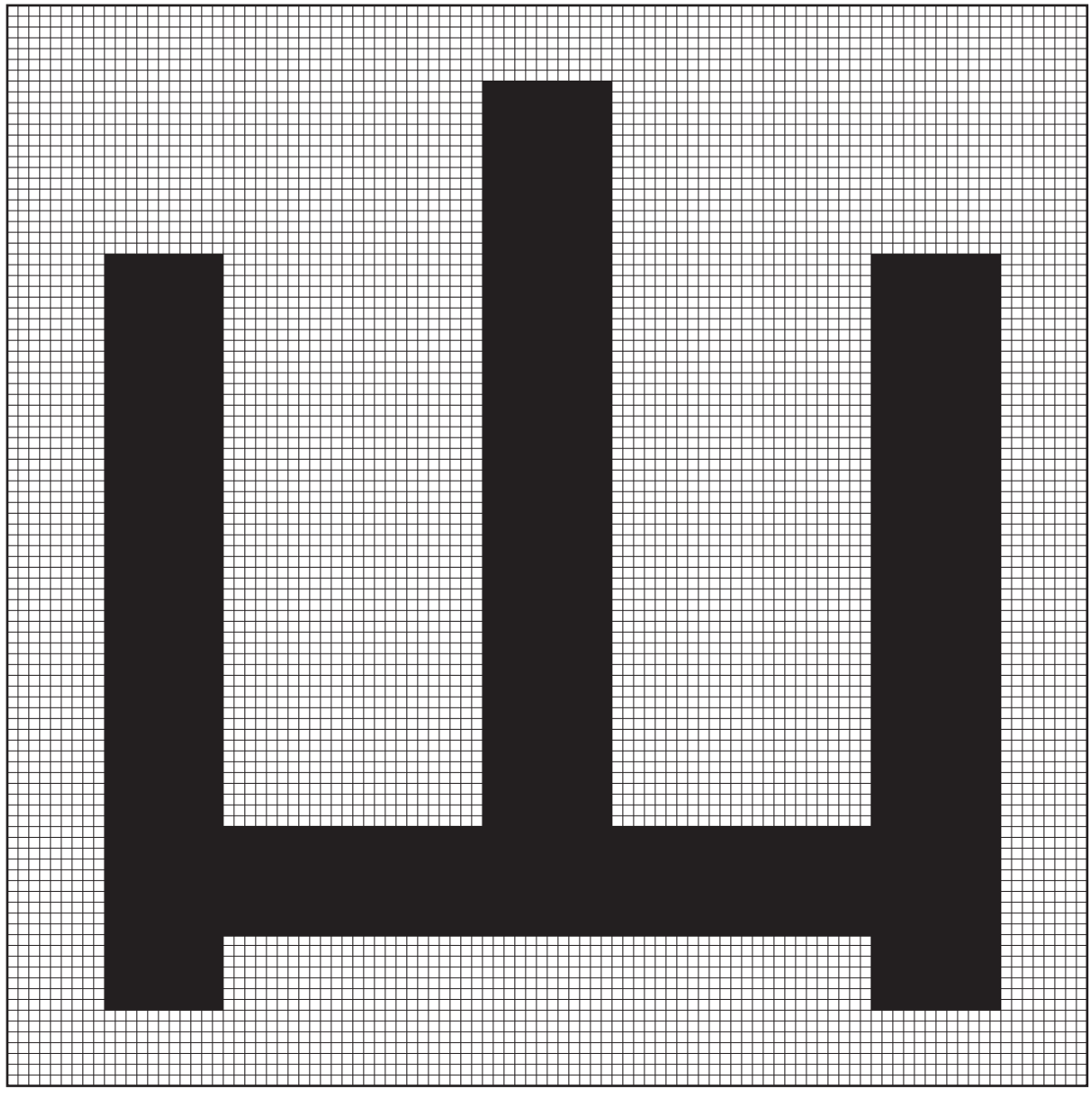


フォントの中には
点の座標が入っている。

(9, 7) - (9, 77) - (20,77) -
(20,24) - (44,24) - (44, 93) -
(56, 93) - (56,24) - (80,24) -
(80,77) - (92, 77) - (92, 7) -
(80, 7) - (80, 14) - (20,14) -
(20, 7) - **(9,7)**

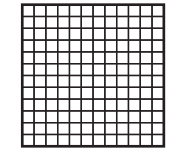
・点の座標は整数
・最後は元の点に戻る

アウトライン表示の仕組み（１）

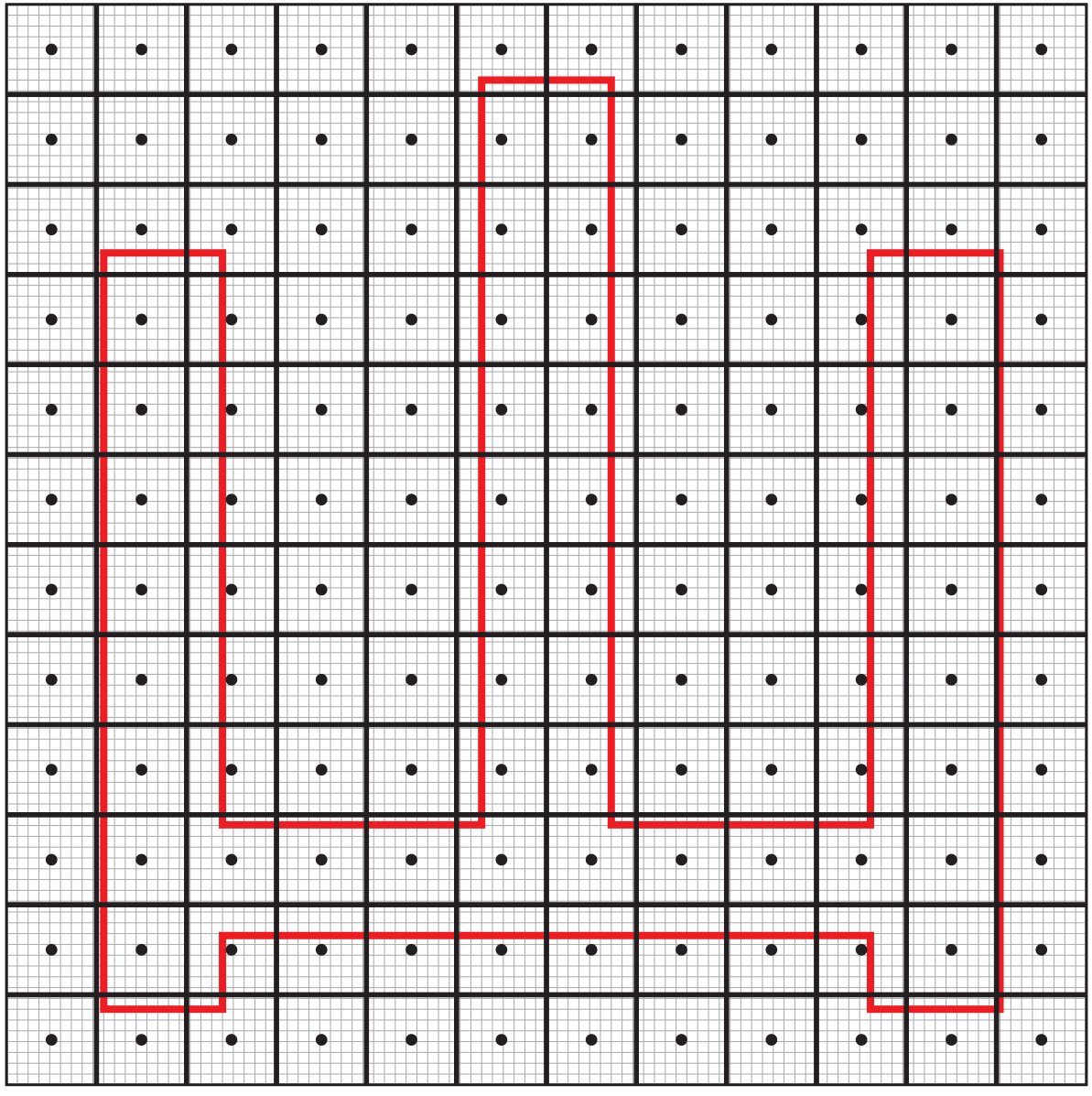


← 囲まれた内側を
塗りつぶした結果

12×12 に
↓ 縮小するには…？



アウトライン表示のしくみ（2）



12×12 のピクセルを
拡大して 100×100 の
升目に重ね合わせると、
ピクセル中心の位置は
次のように計算できる。

⋮

y = (4.5/12) * 100 = 37.5

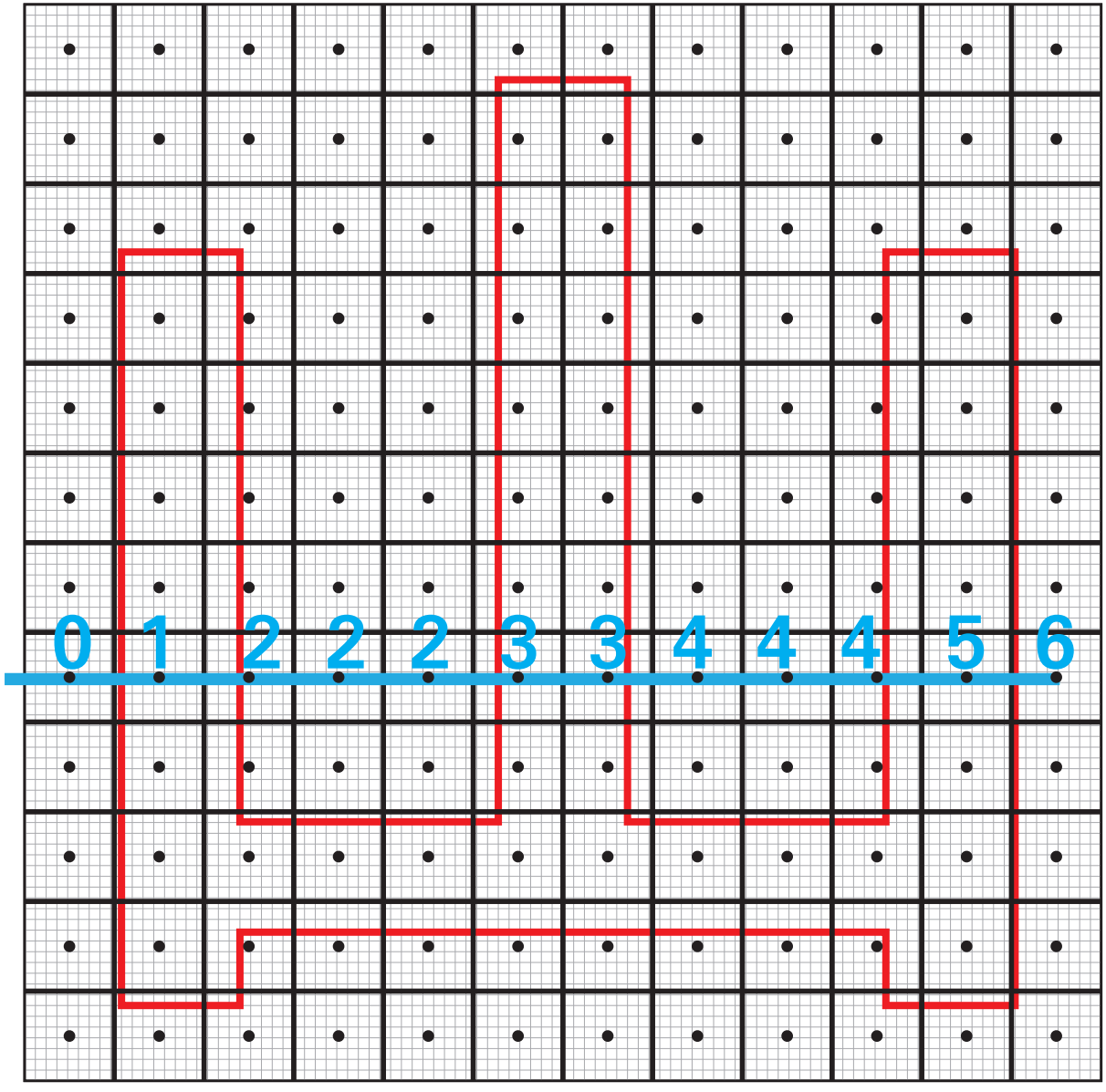
y = (3.5/12) * 100 = 29.167

y = (2.5/12) * 100 = 20.833

y = (1.5/12) * 100 =12.5

y = (0.5/12) * 100 = 4.167

アウトライン表示のしくみ（3）



アウトライン表示のしくみ（4）

y=37.5 の高さの所で
横線を引いてみる。

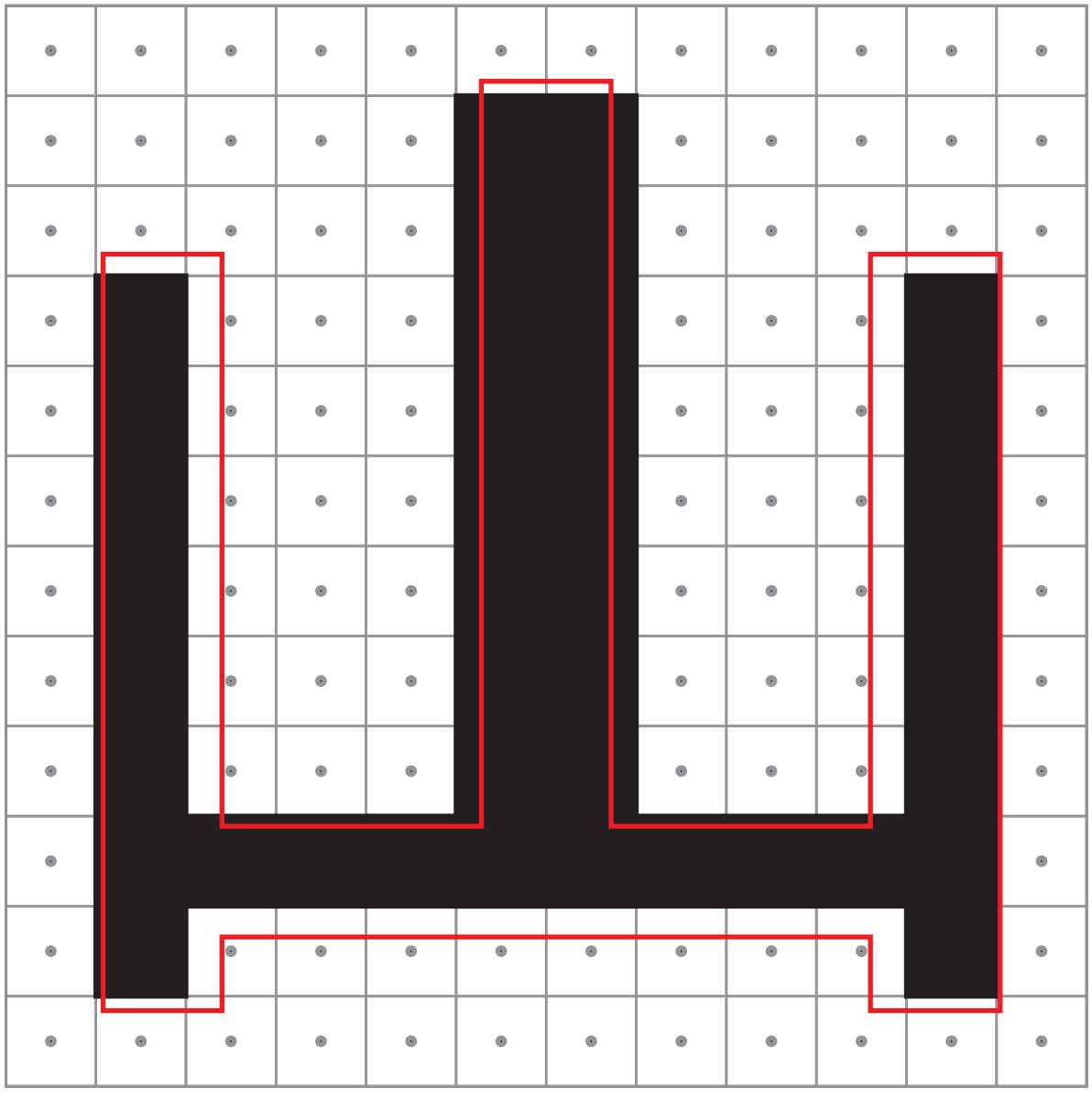
最初はアウトラインの
外側にいて、交差する
たびに内→外→内→…
と移るから、升目の
中心で交差回数

1,3,5 回 = 黒（塗り潰す）

0,2,4,6 回 = 白（そのまま）

となる。

アウトラインを構成する
全ての線分と y=37.5 の
交点の x 座標を求めれば
塗り潰すピクセルが判る。



アウトライン表示のしくみ（5）

機械的に計算すると
線の太さが不揃いに
なることがある。

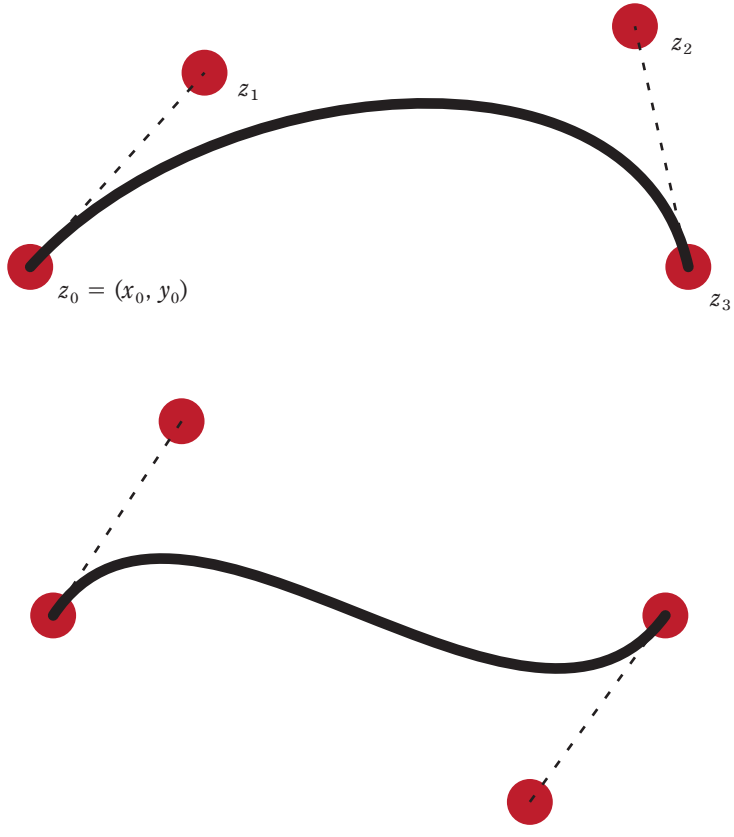
これを避けるために
「ヒント情報」が内蔵
できる。

PS フォントの例
フォント全体の設定：
/StdVW [12] def
/StdHW [11] def

グリフデータの付加情報：
0 12 vstem
35 12 vstem
79 12 vstem
7 11 hstem

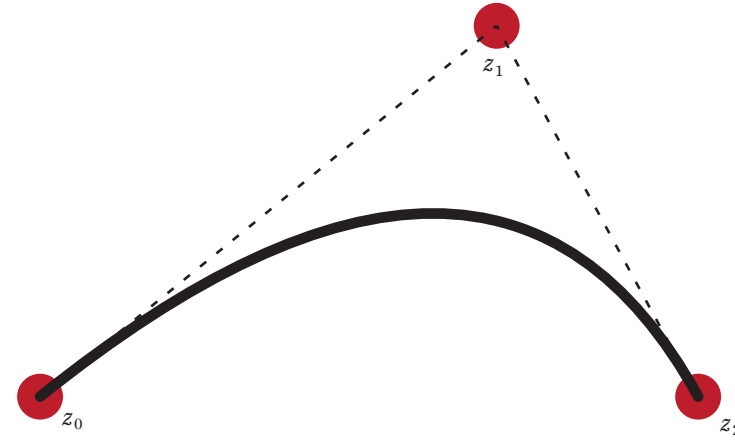
→ 縦線の太さは 1 ドットで統一

2つの Bézier 曲線（1）



$$x(t) = x_0t^3+3x_1t^2(1-t)+3x_2t(1-t)^2+x_3t(1-t)^3$$
$$y(t) = y_0t^3+3y_1t^2(1-t)+3y_2t(1-t)^2+y_3t(1-t)^3$$
$$(0 \leqq t \leqq 1)$$

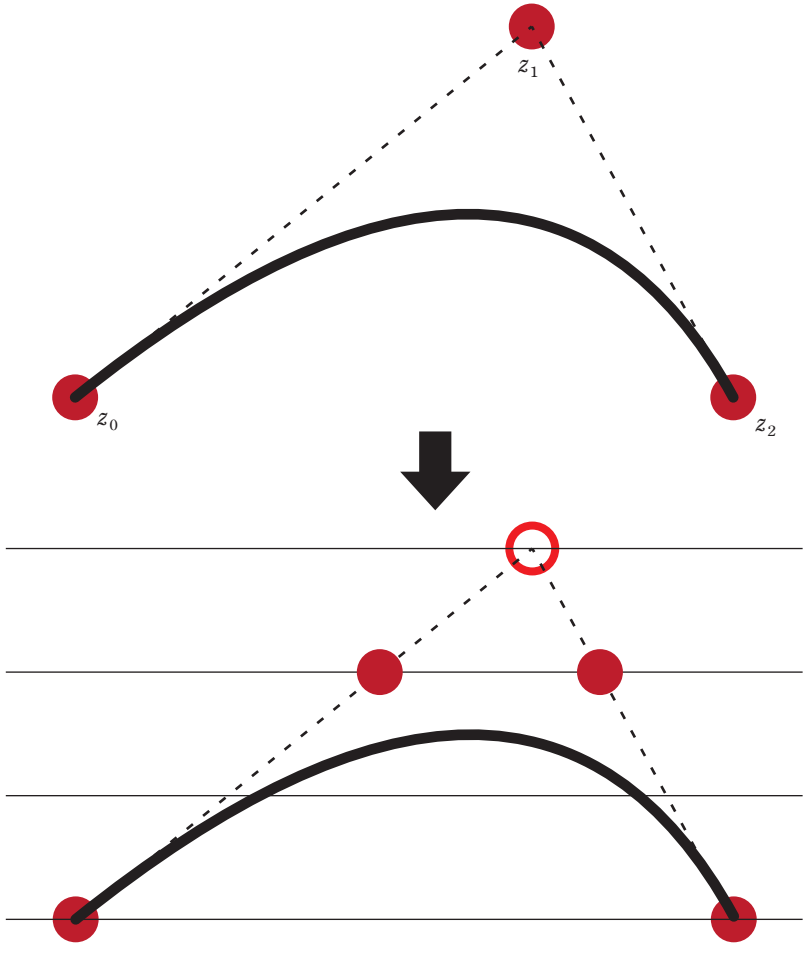
3 次



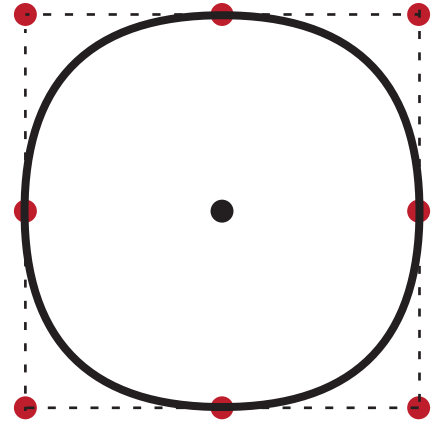
$$x(t) = x_0t^2+2x_1t(1-t)+x_2(1-t)^2$$
$$y(t) = y_0t^2+2y_1t(1-t)+y_2(1-t)^2$$
$$(0 \leqq t \leqq 1)$$

2 次

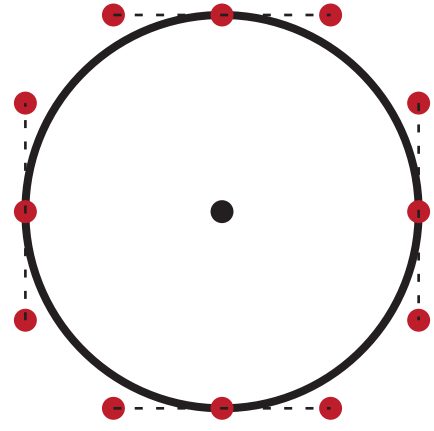
2つの Bézier 曲線（2）



3 次は 2 次を兼ねる



誤差 6.1%



誤差 0.06%

点の数が少なくてすむ

フォントのフォーマットはいろいろある

- PostScriptフォント
  - Type1…欧文用
    - 一度に使えるのは256文字まで（格納は
  - OCF…和文用
    - Composite Font = 多数のType1フォントに割り振る
  - CID…ファイル構造を整理
    - 「文字コード→文字名」から「文字コード→CID（番号）」へ
  - (狭義の) OTF　　etc.
- TrueTypeフォント
  - 基本は1種類だがAppleとMSの仕様がちょっと違う
- OpenTypeフォント（後述）
- Webfontはこれらを変換したもの

OpenType（OTF）って何?

- Appleの多言語技術に対抗して、MicrosoftとAdobeが手を組んだ
- TrueType (MS版) の発展形
  - TrueType Open(多言語技術)＋CFF
- PSフォントをTrueTypeのファイル構造に組込み
  - そのために作ったのがCFF(Compact Font Format)

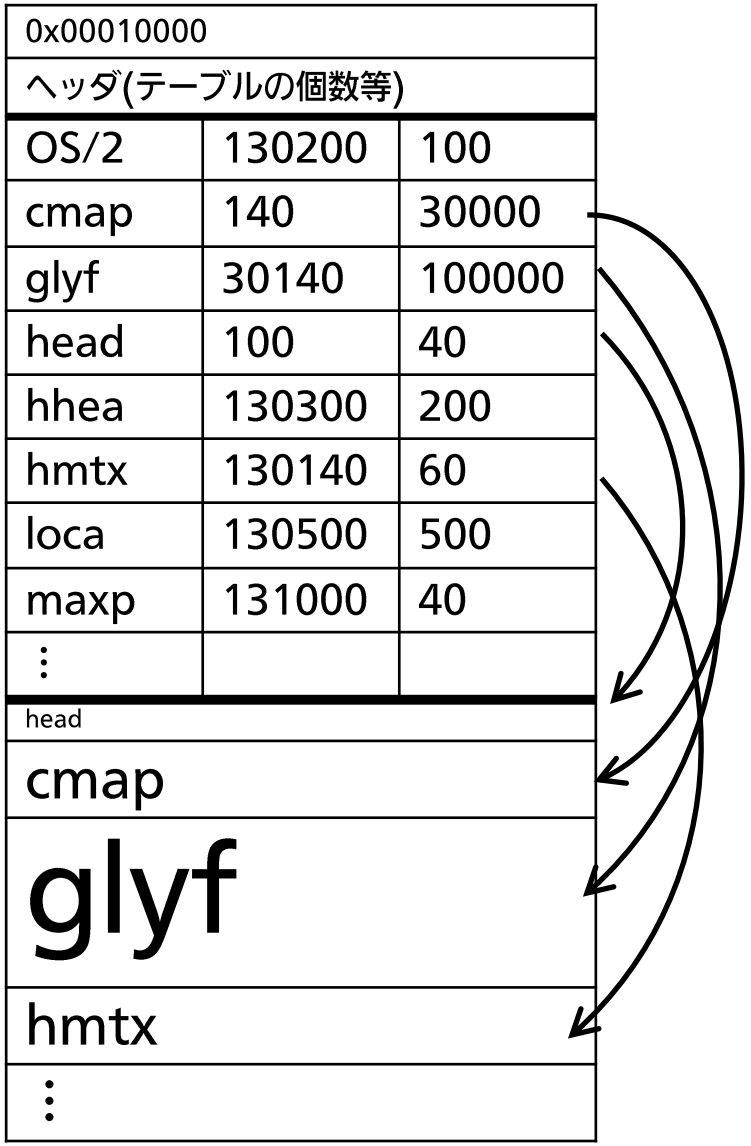
2種類のOpenType

- glyf テーブル(TTF) と CFFテーブル (OTF)
  - 拡張子が違う….ttfと.otf
  - レンダリングエンジンが別→表示が違う
  - アイコンの違いは関係ない
    - デジタル署名（DSIGテーブル）の有無て  か  か決まる
- **前者をOTFと呼ぶのはMS方言**
- **後者も、細かく言えば2種類ある**
  - name-keyed と CID-keyed
  - ふつう、日本語OTFといえばCID-keyedで、Adobe-Japan1 Character Collection に従った物を言う

フォントの内部構造

～各テーブルには何が入っているか～

フォントの全体構造（sfnt構造）



- フォントの内部はデータの種類ごとにテーブルに分かれている
- 各テーブルの場所とサイズを示すsfntヘッダがTTFの先頭に含まれる
- TTCは、含まれるフォントの数だけsfntヘッダが含まれる。
  - グリフ定義テーブルは共有
  - cmapや小さいテーブルはフォント数だけ重複する

フォントにはどんなデータが必要か

- 字形、メトリック、マッピング、メタデータ

- 字形…*glyf, loca, fpgm, cvt, prep* / CFF
- 字形(ビットマップ) …EBDT, EBLC, EBSC
- メトリック等… hhea, hmtx, vhea, vmtx kern, VORG, BASE, JSTF , GPOS, gasp, LTSH, hdmx, VDMX
- マッピング等… cmap, *loca*, GSUB
- メタデータ…head, maxp, OS/2, PCLT, name, post, DSIG

字形

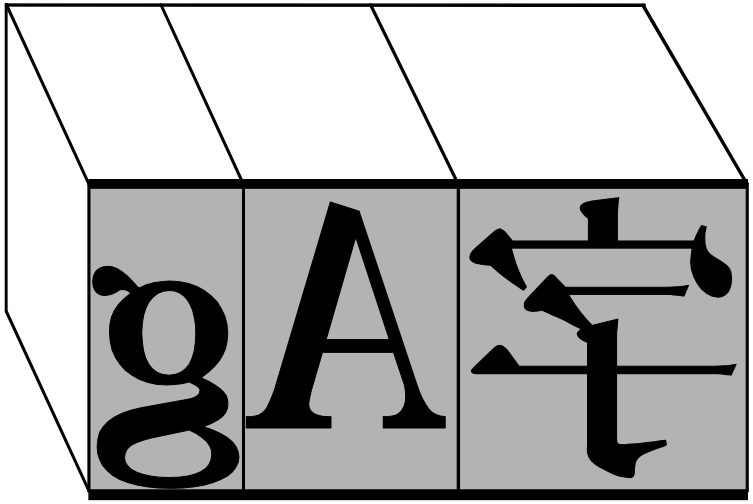
- アウトライン
  - glyf (.ttfの場合) / CFF (.otfの場合)
  - ヒント
    - OTFではCFFに内蔵
    - TTFではglyfの字形記述のほか、以下のようなサブテーブルを使う
      - cvt　StdHW, StdVWのような値を格納する配列
      - fpgm よく使われる
      - prep サイズごとに固有の初期化処理
- ビットマップ
  - EBDT/EBLC/EBSC（MSの書式）
  - グレイスケールも格納可能

メトリック

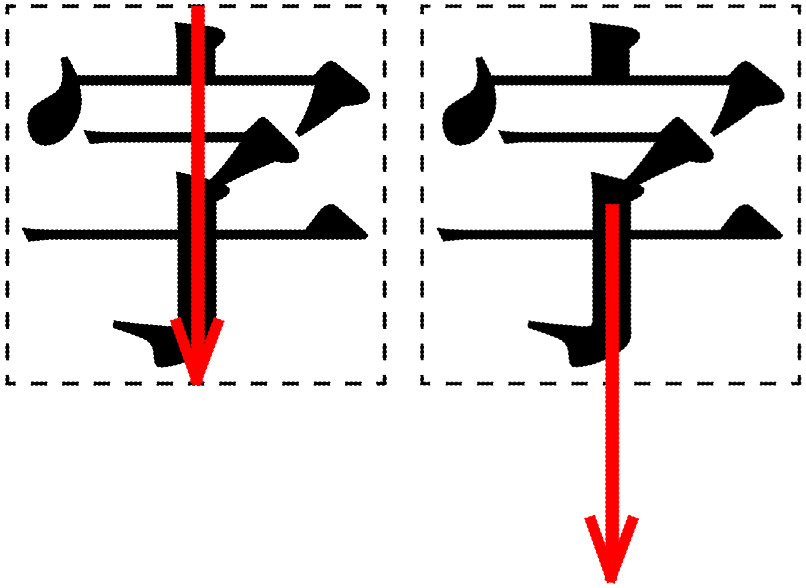
- 活字のボディとか、写植の送りに相当するデータ
  - 活字の「ボディ」という長方形
  - 写植の「基準点＋送り幅」の考え
- フォントでは、
  - hmtx/vmtx で横/縦の送り
  - BASE(混植) / VORG (縦組み) / JSTF (字間調整) で調整
- ボディが並ばない特殊な場合
  - カーニング … kern, GSUB
  - ダイアクリティカルマーク … GSUB
  - インド系文字（入れ替え）

活字と写植の考え方の違い

- 活字にはボディがある
  高さと幅が必ずある



- 写植の送りは仮想的
  基準点は任意

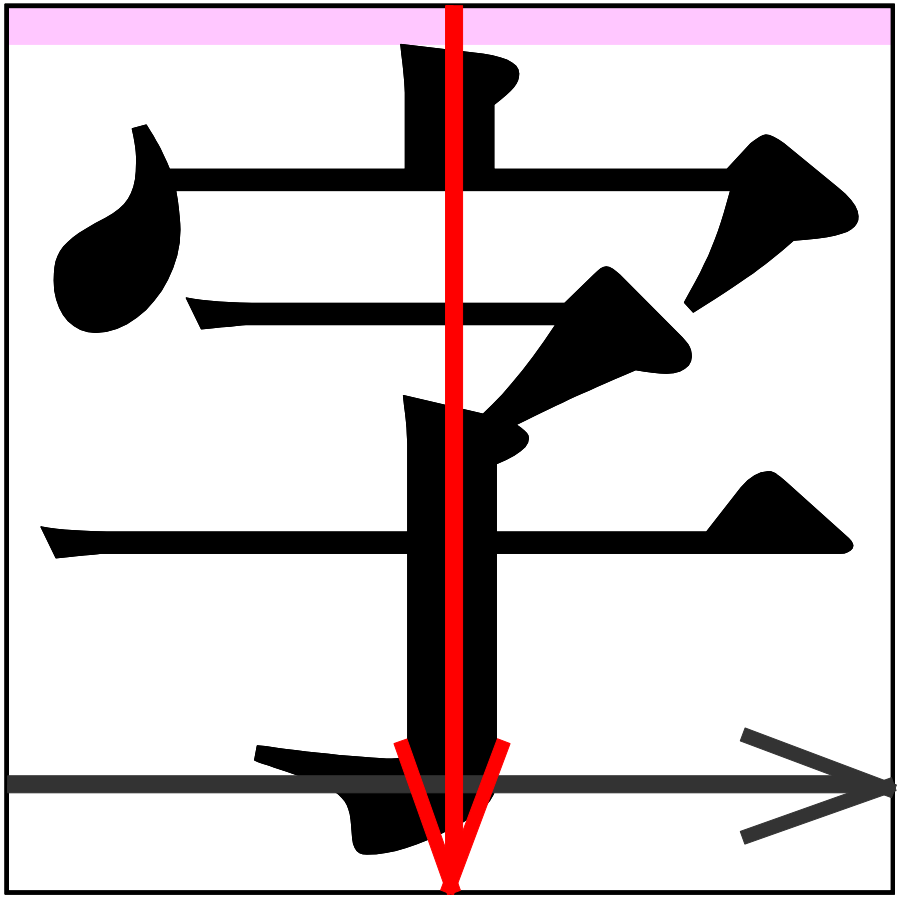


→デジタルフォントの考えは、写植と基本的に同じ

デジタルフォントのメトリック

- 座標上の(0,0)が横書きの基本開始点
  - 欧文のベースライン
  - 文字の上下端はWin/Macで別の所に入っている
    - Windows : OS/2のWinAscent, WinDescent
    - Macintosh: hhea の Ascender, Descender
  - 標準の行送りもWin/Macで別
    - Windows: OS/2のLineGap
    - Macintosh: hheaのLineGap
- 日本語フォントは慣習的に (0, -120) が左下
  - 実際の従属欧文のベースラインは0より上が多い

縦書きのメトリック



- 縦組みのメトリックはvmtxテーブルで定義
  - advanceHeight (赤矢印)
  - topSideBearing
    （ピンクの長方形の高さ）

「高さと上のすき間」で指定。

- X方向の基準位置指定はできない
  - 基本的に全角の中央
  - BASEテーブルで字種別の設定はできる
  - 個別の寄り引き補正とかは不可能

GIDとCID

- 字形データはただデータとして並んでいるだけ
  - 並べばそこに通し番号がつけられる
  - 組版プログラムは、グリフの連番（GID）で把握する
    - フォント内の位置データはプログラムは知らない
- GIDはフォント毎に異なる
  - Unicode順（ふつうのTTF)
  - 漢字の頻度順（メイリオ）
- 日本語OTFでは、どのフォントでも同じ
  - Adobeが日中韓の書体で標準化
  - CIDという
  - 普通はGID=CID (サブセットを除く)
    - 実際には違いがある（後づけで拡張するため）

Cmap

- アプリの外ではテキストデータは文字コードで表す
- 外部リソースの場合もある (CIDFont)
- 複数入っている場合もある
  - 複数の文字コードに対応した場合
  - TTCの場合
- 全部の文字がマッピングされているとは限らない
- TTFの場合、cmapとlocaがペアで使われる
  - loca テーブルは「GID → glyf内のバイトオフセット」
  - OTFのCFFはloca相当のインデックスを内蔵している
- 拡張を重ねてきた規格なのでいくつもフォーマットがある

メタデータ

- name: 人間が読むデータ
  - フォントファミリー名, サブファミリー名, PSフォント名
  - 著作権表示, デザイナー, サンプルテキスト, ……
- OS/2 ほか: 機械が読むデータ
  - アセンダ／ディセンダの標準値
  - 標準の行間
  - 埋め込み可能フラグ
  - 下つき／上付き文字のサイズと高さ
  - 取消線／下線の太さと高さ
  - サポートする文字コードの範囲
  - PANOSE（類似フォント検索用の特徴値）　　等

「複雑な」テキスト表示について

～GSUB, GPOSの構造～

どうしてテキストが表示できるの?

- プログラムがやっている
  - OSのAPI（単純文字列）
    - アウトラインをBMPに変換し、指定された位置に画面表示
    - GDIやUniscribeが代表的
  - アプリケーションの内部処理（組版ソフト）
    - 文字コードからGIDに変換し、
  - ブラウザコンポーネント
- フォントはただのデータです
  - 指示は入っている
  - ただし全てが入っているわけではない
    - インド系文字の順序入れ替えなどはライブラリに組み込まれている
    - 「ルビは半角」なんてどこにも書いてない

「複雑」って何?

- 欧文の最も単純な場合以外は全部複雑と言える。
  - 行端揃えなし、カーニングなし、改行手動指定
- 和文だとベタ組みでさえも調整が必要
  - 約物連続とか
  - 組版プログラム側が処理するのが普通。
- ここでは、フォント内の追加情報を利用する組版に限って言う。

複雑な指示が必要になる場合

- 別のグリフに差し替える (GSUB)
- 位置を移動する(GPOS)
- サイズを変更 (上/下付き、ルビ、混植(BASE))

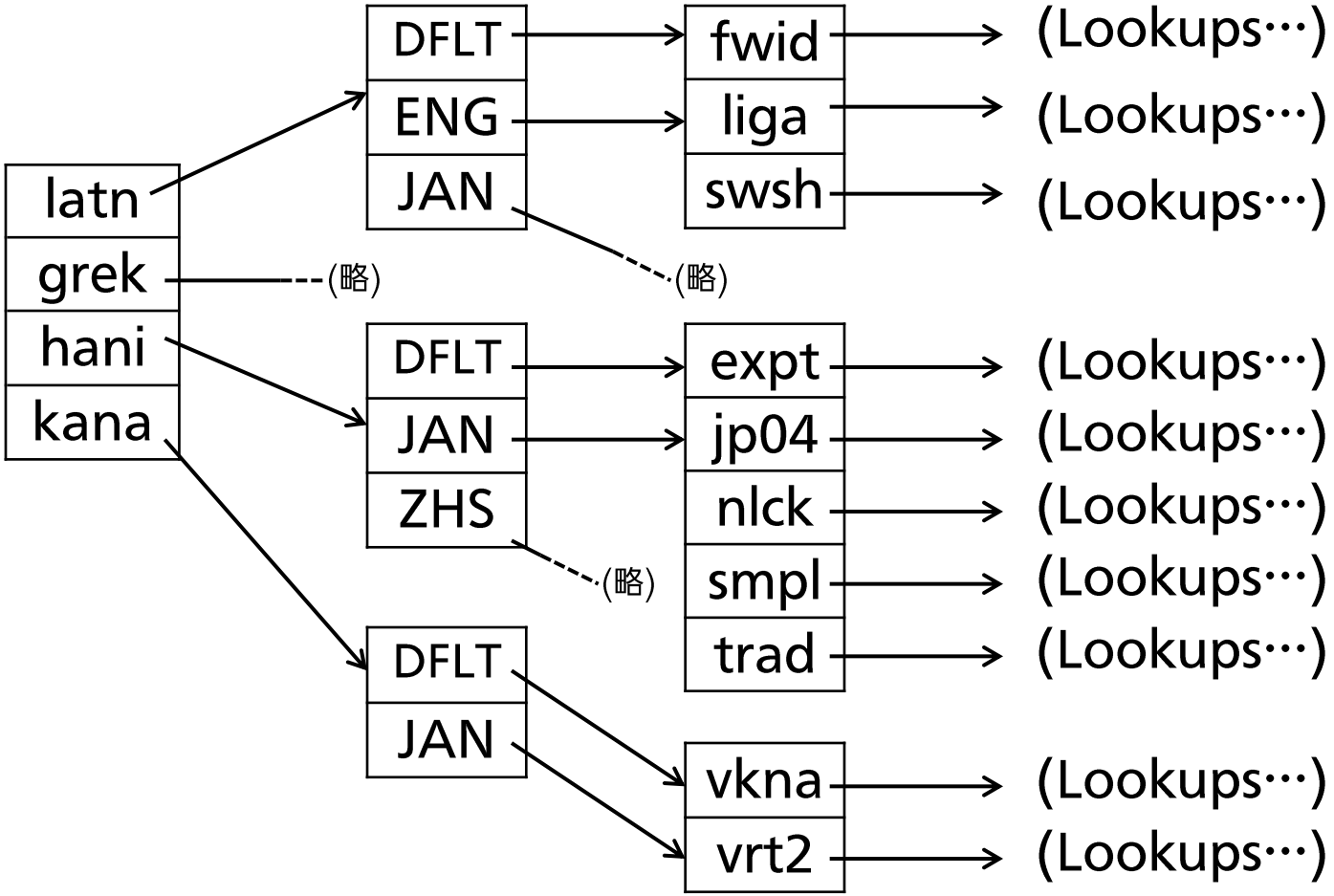
- 以下はアプリ独自処理につき今回取り扱わない
  - 回転・変形（縦書き中の欧文、アプリ独自処理）
  - 非一次元的な組版（割注・ルビ。アプリ独自処理）

「フィーチャ」

- 英語の“feature”
  - よく「機能」と訳されるが「特性」のほうが近い。
- 文字表示に変化をもたらす文脈を表す
- GSUB/GPOS共通で、4文字のタグで呼ぶ
  - 例えば「縦書き」(vert) とか「行頭」(lfbd) とか
  - 登録制でどんどん増える。今115種類くらい
- InDesignの文字パレット「OpenType機能」タブ
  - 例: 「任意の合字」(dlig: discretionary ligatures)
    - 「株式会社→ ㍿」等（「組文字」といった方が分かりやすい）
- 字形パレットでもフィーチャによる絞り込み

レイアウトテーブルの構造

- レイアウトテーブルは3層構造のインデックス
  - Script(用字系)-Language-Feature



Lookupの構造

- 処理の対象とするグリフを検索する表
  - グリフ1文字、またはグリフの列を検索対象とする
  - 検索にマッチした各項目に、置き換えグリフ（列）や位置調整の指定が対応づけられている。
    - GSUB LookupType 1: 1対1の置換
    - GSUB LookupType 3: 選択型の置換
    - GSUB LookupType 4: 合字置換（多文字→1字形の置換）
    - GPOS LookupType 1: 単独文字の位置調整
    - GPOS LookupType 2: ペアでの位置調整（カーニング）
    - GPOS LookupType 3: 筆記体の接続
    - GPOS LookupType 4, 5, 6: アクセント類の接続
      - それぞれ通常の文字、合字、他のアクセントにくっつく
- 1フィーチャを複数のLookupで表現することも

GSUBの例1：縦書き

- コードポイントを変えたりCmapを差し替える方式もあった
  - 前者の例：漢字Talk6〜7
  - 後者の例：日本語PSフォントのフォント名
- GSUB方式では２段構え
  - 文字コード → (横書き) GID →縦書きGID
    - まずcmapで変換
    - 次にGSUBの‘vert’または‘vrt2’で変換

    - vert: 縦書き変形 (TTF)
    - vrt2: 縦書き変形+欧文回転 (OTF)

縦書きの具体例

- 横書き

2399 888 634 2422 635

春 は 、 曙 。

- 縦書き

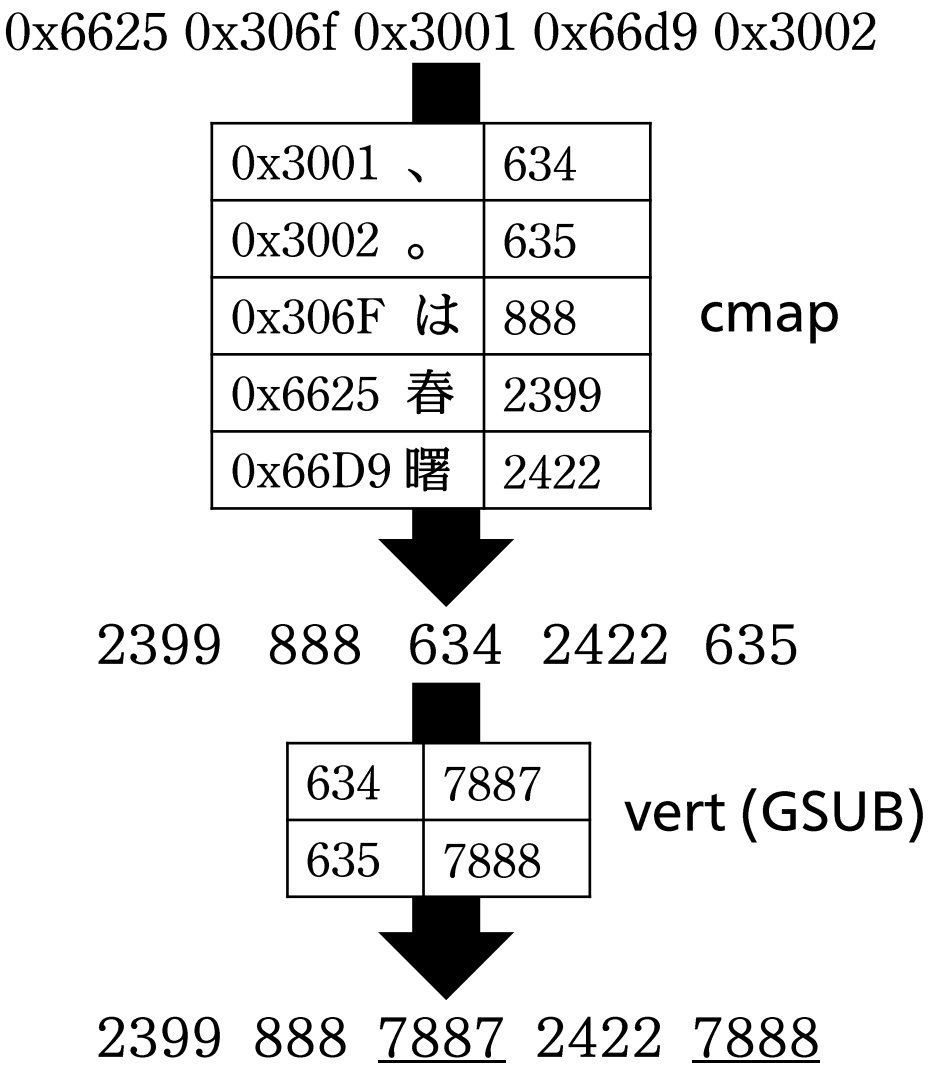
2399 春

888 は

7887 ﹅

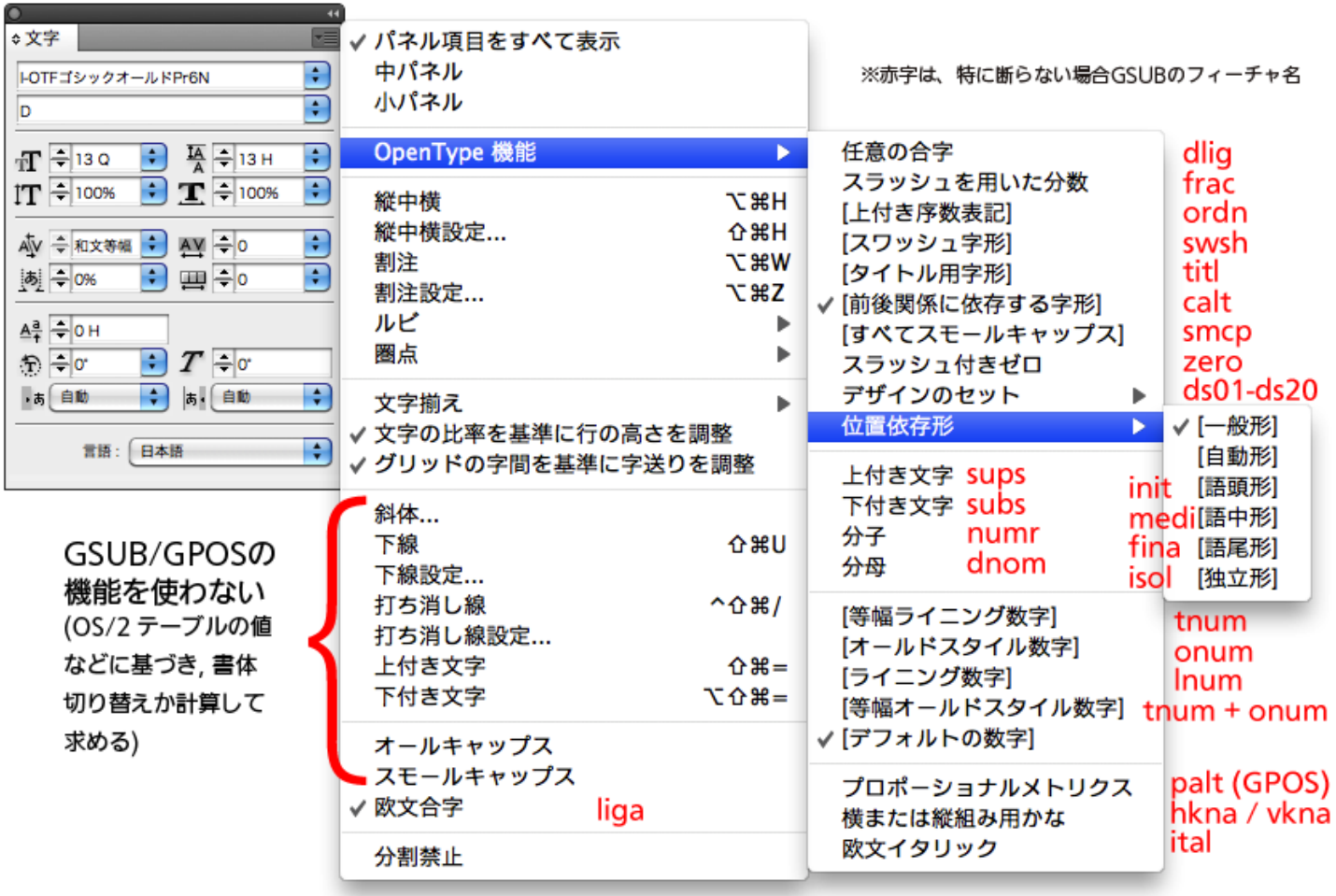
2422 曙

7888 ﹒



GSUBの例2: ユーザの指示による置換

- 例：InDesignの文字パレット



主なGSUBのフィーチャタグ（欧字）

- hwid, fwid, pwid（等幅半角/全角/プロポーショナル）
- twid,qwid （等幅三分/四分数字）
- vert, vrt2 （縦書き/縦書き+縦書き回転）
- ital （欧文イタリック）
- hkna, vkna （横組み/縦組み用かな）
- ruby （ルビ用かな）
- sups, subs （上つき/下つき文字）
- dnom, numr （分数用上つき/下つき数字）
- aalt （全異体字に選択型のアクセス）
- nalt （丸つき数字等の異体字に限った選択型一覧）

主なGSUBのフィーチャタグ（和字）

- expt（エキスパート字形）
- jp78, jp83, jp90 （JIS X 0208各版の例示字形）
- hojo（JIS X 0212 (補助漢字) 字形）
- nlck （印刷標準字体）
- jp04 （X 0213:2004変更字形+第三・第四水準形）
- trad, smpl （旧字体、略字体）
  - 注意：中国語フォントでは繁体字・簡体字変換の意。
- hkna,vkna,ruby （縦書き・横書き・ルビ）
- pkna （プロポーショナル仮名）
  - ツメとは異なり、別グリフなので字形そのものが違う
  - 縦組みはpkna→vrt2の順で２回GSUBを適用

GSUBの例3: 言語による切り換え

- Arial Unicode MS は各国の漢字字体を実装
  - ‘locl’　フィーチャを使用

U+904D				
Arial Unicode MS	遍	遍	遍	遍
GID	25308	29943	32905	37393
言語タグ		KOR	ZHT	ZHS
ISO/IEC 10646:2003				

参考: smpl/trad は日本語と中国語で意味合いが違う

- 日本では漠然と略字・旧字（一意とは限らない）
- 中国では繁体・簡体切り換え

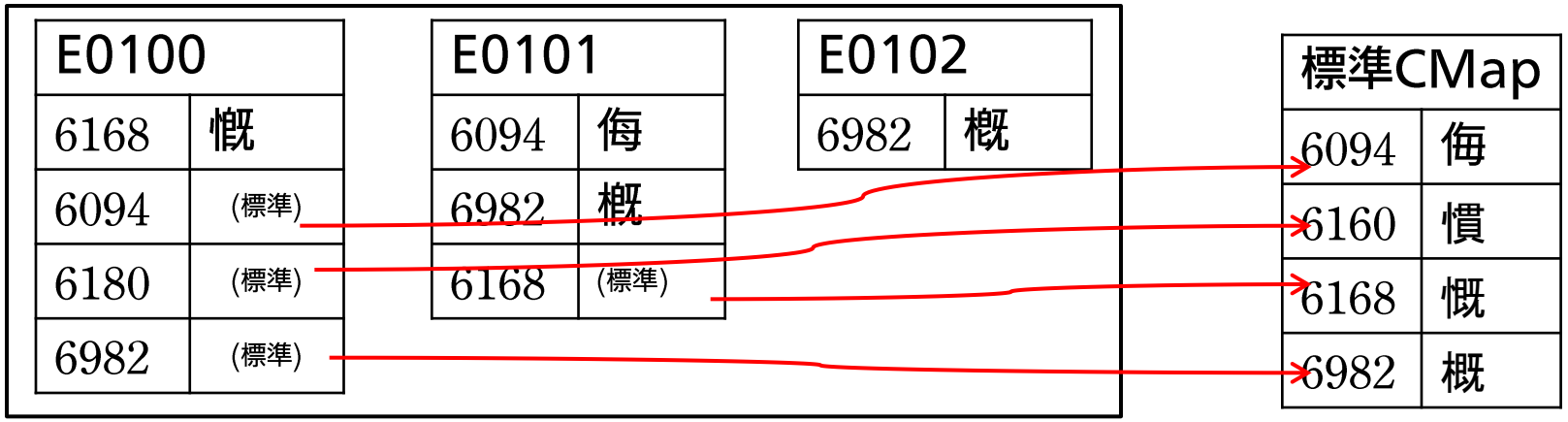
GSUBの例4, 5: 選択型の置換・合字

- 選択型 （aaltやnalt等）は「1→多」の変換
  - 選ばれるのは一つだけ
- ユーザの指示による
  - ランダムに選ばれる ‘rand’ を除く
- 字形パレットの三角表示をクリックして選ぶアレ
  - ‘aalt’ 機能には全ての異体字を登録することになっている
  - この並び順は標準化されていない
- 1対1の変換と併用される例も（trad等）

- 合字は「多→1」の変換
  - 置換文字のほか、合字の途中にカーソルを置くときの位置情報も登録できる

IVS

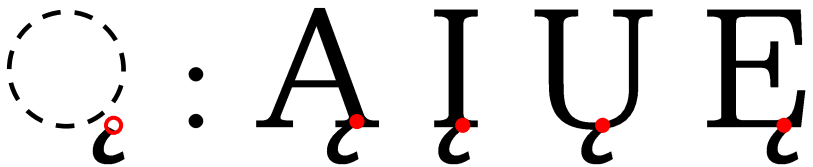
- Unicodeで定められた異体字識別の仕組み
- 異体字番号を標準化してU+E01xxに割り当て
- 漢字の後ろにくっつけると字形が表わされる
  - 考え方としては合字に似ている
- フォントの内部ではcmapの拡張(format 14)
  - 標準の Unicode cmap (format 10) の拡張として働く



※イメージ図であり、実際のマッピングとは異なります

GPOSの例

- カーニング（kern）
  - 昔はkernテーブルを使った
  - GPOSでは(ある程度)複雑な条件判断もできる
    - V.Vのような並びはカーニングしない、とか
- アクセント記号（mark, mkmk）
  - アクセント記号は常に真中につくとは限らない



  - ‘mark’ フィーチャで、「この種類のアクセントはここにつける」という位置を指定できる。
- 仮名のツメ情報 （palt）
  - XPlacement, XAdvance, YPlacement, YAdvance

参考文献（1）

- 規格を決めたベンダーの仕様書
  - OpenType Specification 1.6
    http://www.microsoft.com/typography/otspec/
  - Adobe Font technical notes
    http://partners.adobe.com/public/developer/font/
- Haralambous 著, Fonts & Encodings (O'Reilly)
  http://oreilly.com/catalog/9780596102425/
- PostScript詳細解説（CQ出版社）
  http://www.cqpub.co.jp/hanbai/books/18511.htm/
- Knuth 著, METAFONTブック（アスキー）
  http://ascii.asciimw.jp/books/books/detail/4-7561-0194-1.shtml/
- 三浦曜監修, CAD･CG技術者のためのNURBS早わかり（工業調査会）
- FontForgeのマニュアル
  http://fontforge.sourceforge.net/
  日本語訳は放置中……引き継いでくれる人は大歓迎

参考文献（2）

- 日本語組版関連
  - 向井裕一著, 日本語組版の考え方（誠文堂新光社）
    http://www.idea-mag.com/jp/publication/b024.php
    - JIS X 4051:2004「日本語文書の組版方法」
      （JIS X 0213とフォントメトリックが異なるのに注目）
  - W3C技術ノート・日本語組版処理の要件
    http://www.w3.org/TR/jlreq/ja/
- 各時代の一般的解説書
  オブスキュアインク編,
  - デザイン、DTPのためのフォントの鉄則（2003, 毎日コミュニケーションズ）
    - 小形さんも文字コードについて書かれてます。
  - 藤岡,和田,小西共著, DTPフォント入門 Macintosh編 (1999, MdN)
  - 佐藤友治著, Macintoshフォントブック (1992, みずき出版)