

Copyright © 2000 – 2015, George Williams and the FontForge Project contributors.
Shared freely under the project license.
日本語版: 旧オンライン日本語版を元に作成 (2024)

関連文書・総目次 Documentation

※ 黒字部分が本分冊版の収録範囲です。

About FontForge	FontForge プロジェクトについて
Design With FontForge	FontForge 入門：FontForge をはじめるためのオンライン・ブック
Overview	FontForge の概要
Tutorial	チュートリアル
User Interface	FontForge のユーザー・インターフェース・ガイド
Scripting FontForge	FontForge 用スクリプト（定形処理プログラム）
Technical References	FontForge 開発者のための技術関連情報
Utilities	FontForge を補完するユーティリティ・プログラム（フォント検証ユーティリティ）
FAQ	FontForge よくある質問
Appendices	【付録】FontForge に関連する情報
Typographical glossary	フォントとタイポグラフィに関する用語集

この文書のライセンスは、
英語原文版および旧日本語翻訳版のライセンスに
準拠するものとします。

改版履歴

版数	発行日	内容
1.00	2024/03/15	FontForge Documentation 日本語アップデート版 • 旧日本語版を元に、英語版 https://fontforge.org/docs/ 内容に合わせて改訂、分冊化（Book 1） pdf 版
0.00		旧日本語版（底本） [Japanese documentation (outdated)] https://fontforge.org/docs/old/ja/overview.html#intro

FontForge プロジェクトとは

What is the FontForge Project?

FontForge は「フリー」ソフトウェアですが、この「フリー」とは「言論の自由」などというときのような「フリー」を意味しており、「無料のビール」などという場合のような単に価格が「フリー」であるということだけを意味しているわけではありません¹。ソフトウェアの自由とは、ソフトウェアの動作に関して、各ユーザーが開発者と同等の権限を持っていることを意味します。プログラムのコピーを持っている人は誰でもソースコードにアクセスでき、自由にコードを変更してプログラムの動作を変更できます。

さらに言えば、各ユーザーは、プログラムのコピーをいつどのように配布するかについて、開発者と同等の権限を持っています。誰もが、利用料金の有無にかかわらず、コピーを変更せずに、または改良を加えて再配布できます。

どこでも安価なインターネット アクセスが利用できるため、この「自由」ということは、「フリーな libre」ソフトウェアが通常無料で入手できることを意味します。

しかし、さらに重要なことは、ソフトウェアの継続的な開発が「プロジェクト」という広く公開された方法で行なわれているということです。

ユーザーが自らの生活に役立つようプログラムのコードを変更した場合、その変更を非公開のままにすることができます。しかし、ユーザーはプロジェクトから、その改善内容をより多くの人々も利用できるようにするよう勧められています。頻繁にそのプログラムに貢献している人がその改善内容をレビューし、おそらくいくつかの議論と修正を経て、他のすべてのユーザーが楽しんでさらに発展できるようにその改善内容を次のリリースに含めます。

(Git バージョン管理システムと GitHub Web サイト²を使用すると、こうした作業が簡便になります。)

公共の場で何かを行なうと、その内容に対する議論、批評、さらには批判が必然的に起こります。それでも、貢献することは非常にやりがいのあることです。FontForge プロジェクトに貢献するのが楽しいと感じていただければ幸いです。

1 このため、英語では「フリー」ソフトウェアのことを、単に「無料 free」であるだけという誤解を防ぐために、「自由な libre」という語を用いて、「ソースコードを入手・研究・変更・再利用する自由」が保証されていることを示しています。

2 <https://github.com/>

FontForge 入門

Design With FontForge

『Design With FontForge』は、FontForge をはじめるためのオンライン・ブックです。以下のサイトを参照してください。

<http://designwithfontforge.com/en-US/index.html>

《訳注》

ラテン文字を事例にフォント作成作業の流れが説明されています。

日本語に関する直接の言及はありませんが、日本語同様に
字体数の多いインド語系文字（デーヴァナーガリー文字）での
作成事例も紹介されています。

【文書 目次】

1.	はじめに	Introduction
2.	フォントとは	What Is A Font?
3.	自分の目を信頼する	Planning Your Project
4.	プロジェクトを計画する	Trusting Your Eyes
5.	エム（EM）スクエア	The EM Square
6.	FontForge をインストールする	Installing FontForge
7.	FontForge の設定	Configuring FontForge
8.	ユーザー・インターフェース（UI）の概要	General UI Introduction
9.	FontForge の描画ツールを使用する	Using the FontForge Drawing Tools
10.	螺旋ツールで描画する	Drawing with Spiro
11.	文字「o」と「n」を作成する	Creating 'o' and 'n'
12.	フォント情報とメタデータ	Font Info & Metadata
13.	単語間スペース	Word Spacing
14.	フォントの DNA を生み出す	Creating Your Type's DNA
15.	大文字	Capital letters
16.	行間スペース	Line Spacing
17.	句読点と記号	Punctuation and Symbols
18.	小文字を完成させる	Completing the Lower Case
19.	発音区別符号とアクセント記号	Diacritics and Accents

20.	数字	Numerals
21.	太字、その他の文字太さ	Bold
22.	斜字体	Italic
23.	スペース、メトリクス、および カーニング	Spacing, Metrics and Kerning
24.	作成したフォントの検証	Making Sure Your Font Works, Validation
25.	最終出力、フォント・ファイルの生成	The Final Output, Generating Font Files
26.	FontForge に問題が発生したら	When Things Go Wrong With FontForge Itself
27.	デーヴァナーガリー文字書体のデザイン	Designing Devanagari Typefaces
28.	他のプログラムからグリフをインポートする	Importing Glyphs from Other Programs
29.	アラビア語フォントにグリフを追加する	Adding Glyphs to an Arabic Font
30.	参考文献	Further Reading
31.	フォント・エディター・ツールの歴史	Font Editor Tool History
32.	用語集	Glossary

FontForge の概要

Overview — an overview of FontForge’s documentation

【蛇足的補足】 本内容は、旧日本語版（狩野宏樹氏全訳）を踏襲し（旧版はあまりにも膨大で扱い辛いので）現行英語版に基づいて再構成しつつ、（必要に応じて加筆・補正・改竄・改訳／改悪・注釈・注記し）書籍風に分冊化したものです。悪しからず。

《本章の構成》

1. FontForge の関連文書.....	7
2. 索引とテーブルの情報.....	9
3. えーっと…、フォントエディターって何ですか？.....	9
4. 初歩的な 概念：フォント、スプライン、線、点、パスと em ユニット.....	10
4.1 文字とグリフのはどう違うのか？.....	10
4.2 アウトラインフォントとは何か？ ビットマップフォントとは何か？.....	10
4.3 スプライン曲線.....	11
4.4 線、点、パス.....	12
4.5 Em ユニット.....	13
4.6 アセンダーとディセンダー.....	14
5. フォント：TrueType、PostScript、OpenType、SVG.....	15
5.1 OpenType とは何か？.....	15
5.2 SVG とは何か？.....	16
5.3 Type3 フォントとは何か？.....	16
5.4 Type1 フォントとは何か？.....	16
5.5 Type2 フォントとは何か？.....	17
5.6 CFF フォントとは何か？.....	17
5.7 Apple 高度組版機能（ATT）とは何か？.....	17
6. ヒント情報.....	17
6.1 Postscript.....	17
6.2 TrueType.....	19
7. 参照.....	19
8. レイヤー.....	20
8.1 Type3 フォント.....	21
8.2 Guidelines.....	21
9. 用字系.....	22
10. グリフ名と名前リスト.....	22
10.1 グリフ名.....	22
10.2 名前リスト.....	22
11. 検索、機能、スクリプト、言語.....	23
11.1 検索 Lookups.....	23
11.2 機能 Feature.....	23
11.3 文字 Script.....	23
11.4 検索サブテーブル.....	26
11.5 検索はどのように行なわれるのか.....	26
11.6 GPOS と GSUB.....	27
12. アンカーポイントとアンカークラス.....	28
12.1 アンカーポイント.....	28
12.2 アンカーポイントの仕組み.....	29
13. ベースライン.....	30
14. フォントビュー.....	31
15. アウトライングリフビュー.....	33
16. ビットマップビュー.....	34
17. メトリックビュー.....	34

1. FontForge の関連文書

FontForge では、Postscript、TrueType、OpenType フォントを作成および修正できます。フォントをさまざまなアウトライン形式で保存し、ビットマップを生成します。

イーヨーは、地面においてある三本の棒をながめていました。二本は、かたほうのはしでぶつかっていて、もう一方のはしでは、はなれていました。そして、その二本の上に、もう一本の棒がのっていました。コブタは、きつとなにかの**わな**だろうと思いました。

「あのイーヨー。」と、コブタはもういちどいいました。「ぼく、ちょっと——」

「コブちゃんかな？」イーヨーは、まだ棒をながめながら、いいました。

「ええ。イーヨー。ぼくね——」

「おまえ、これ、なんだか知っとるか？」

「いいえ。」

「これは、**A** の字じゃ」

「ああ。」と、コブタがいました。

「アーじゃないぞ。エーじゃ。」イーヨーは、こわい声でいいました。

— A. A. ミルン, 1928 年
『プー横丁にたった家』
(石井桃子訳)

フォントデザインは**罠**になることもありますが、むしろ楽しい**わな**であることを私は知っています。用心して遣ひませう³。

3 原文ラテン語「Caveat utor.」(= I use caution.)

2. 索引とテーブルの情報

- [Index](#) 【索引】用語の一覧（英語）
- [Module Index](#) 【Python Module Index】
- [Search Page](#) 【検索ウィンドウ】 ※複数語の検索では全単語一致のみです。

3. えーっと…、フォントエディターって何ですか？

フォントエディタは、フォントを作成および修正するために設計されたプログラムです。

最も明白な点は、文字の輪郭を描画できる FreeHand⁴、Inkscape⁵、Illustrator⁶ などのような描画プログラム（ベクター形式の画像を作成できるドローソフト）であることです。ただ、他のそのような描画プログラムとは異なり、フォントエディターでは、一度に多くの絵（文字ごとにひとつ以上の形状）を描画することと、作成した文字をひとつのデータベースに収集するようにすることが求められています。

これにより、互いの画像がどのように相互作用するかを記述することができます（ひとつの画像をもうひとつの画像の後に配置する場合に、通常ふたつの画像はフォントの「メトリクス〔設定値〕」によって特定の距離だけ離されますし、あるいは、ふたつの画像が互いに密着して配置される場合は、それらの画像が互いに相互作用して三番目の画像〔フォントの合字「リガチャ」〕に変化するなど、です）。

最後に、フォントエディターは、ユーザーが描いたすべての画像データと、それらの画像がどのように組み合わせられるかということに関するすべての「メタデータ〔データの付属情報〕」を丸ごとひとつにまとめて、コンピューターがテキストの表示に使用できるフォントに変換するのです。

4 [Macromedia FreeHand](#)（商用版）： 2003 年 3 月開発終了。

5 [Inkspace](#)： 無料のオープンソースソフトウェア。

6 [Adobe Illustrator](#)（商用版）：




4. 初歩的な概念：フォント、スプライン、線、点、パスと em ユニット

単純化すると、フォントとは「グリフ〔文字画像〕」を集めたものです。しかし、コンピュータフォントにはそれ以上の情報が必要です。最低でも、入力されたデータの列を目に見える形のグリフ〔出力画像〕に変換するための情報も含まれていなければなりません。この対応情報を「エンコーディング〔符号化〕」と呼びます。多くのフォントには、隣接するグリフをどのように配置するかルールも含まれています。例えば、ラテン文字のフォントでは、「f」と「i」が隣接した時には、それらの代わりに特別なグリフである「fi」と「il」の合字を出力するように指示するルールがしばしば含まれています。ラテン文字のフォントでは一般にはそのようなルールを必要としませんが（しかしそれらを含むように拡張することはできます）、その他の用字系、例えばアラビア文字では合字は欠かせない物です。

4.1 文字とグリフのはどう違うのか？

文字というのは抽象的な概念であり、グリフはその概念を実体化した物です。「A」という字はひとつの文字であり、その一方、

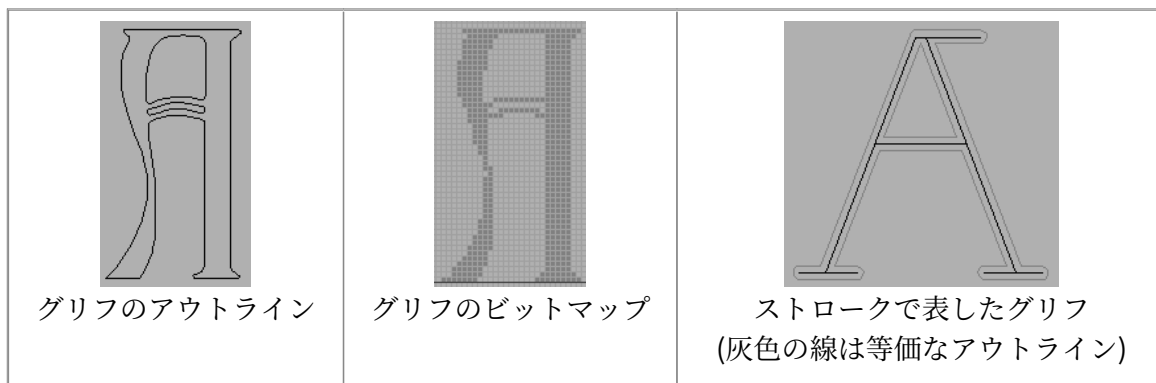


はすべてが文字「A」を表現するグリフの一例です。ラテン文字のフォントでは、しばしば各文字ごとにひとつのグリフが割り当てられていて、各グリフはひとつの文字を表していますが、絶対にそうだとは限りません——ルネッサンス期の印刷では、小文字の「s」にふたつのグリフ 、、「短いs」と「長いs」がありました。ふたつ（以上）の文字を表現するひとつのグリフの例としては、合字が挙げられます。アラビア文字では、ほとんどの文字は少なくとも四つの異なるグリフを持ち、適切なグリフはその前後にある文字により決まります。たとえば、アラビア文字の「シーン」という文字には四つの異なる形  があります。

4.2 アウトラインフォントとは何か？ ビットマップフォントとは何か？

フォント内の各グリフはひとつの図形ですが、コンピュータ上での図形の表現にはいろいろな方法があります。**PostScript および TrueType 形式のフォント**では一般に、図形の「アウトライン〔輪郭線〕」を描画してから、そのアウトラインの内部を塗りつぶします。この塗りつぶし処理のことを「ラスタ化」（ラスタライズ）と言います。それに対して、**ビットマップフォント**はピクセル（画素）が長方形に並んだグリッドを塗りつぶして各グリフを表現しています。第3のタイプもあり、**ストロークフォント**とありますが、これはグリフを各画線の中心線によって表現し、後からその線を所定の幅で肉付けします。ほとんどの場合、ストロークフォントを直接扱うことはしませんが（**〔輪郭を太らせる(E)...〕** コマンドを使うと、ストロークをアウトラインに変換することができます）、ストロークフォン

トを編集したい場合には「[複数レイヤ編集](#)」の章を参照してください。



4.3 スプライン曲線

一本のアウトラインは、一組の輪郭線〔パス＝経路〕から構成されています。左上の図では文字「A」が三本の異なるパスによって記述されています。主要な線は外側にあり、その内側に短い二本のパスがあります。これらのパスは（ベジェ）スプライン曲線と線分とを連ねて構成されています。FontForge は三次および二次のスプライン曲線を取り扱うことができます。PostScript フォントは三次スプライン曲線を、TrueType フォントでは二次スプライン曲線を使用します（SVG フォント⁷ではどちらも使用されています）。

各々の三次スプライン曲線は、4つの点によって定義され、そのうちの二つはスプライン曲線の終端点を示し、残りの二つはその終端点におけるスプライン曲線の傾きを示します。以下の例は、二本の三次スプライン曲線とそのすべての点を表示したものです：

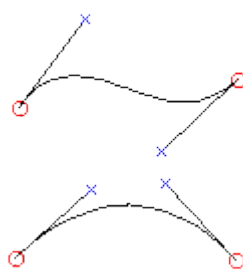


図4.3_1

傾きを指定するための点は「**制御点**」（control points）と呼ばれます。この点は、この図のように（FontForge でも同様ですが）色付きの小さな×印で表されます。制御点を動かすとスプライン曲線の形は変わりますが、スプライン曲線の始まりと終わりの位置は変わりません。

連続するスプライン曲線と線分は、繋がり合っひとつのパス〔輪郭先の経路〕を構成します。こうした線の繋がり方には、次の図 4.3_2 に表したようないくつかの形があり得ます。そのどの例においても、三つの点の相対的な位置はほぼ同じになっています。三つの点はスプライン曲線や直線によって繋がっていますが、その繋がり方によって結合部の形は変わります。

最初の例では制御点は存在せず、その結果、二本の直線となり二番目の例では、一本の直線とスプライン曲線が交点で接した形になり、三番目の例では、二本のスプライン曲線がこちらも交点において接しています。最初の例の交点は「**角の点**」（corner point）と呼ばれ、その点から出るまたはその点に入るスプライン

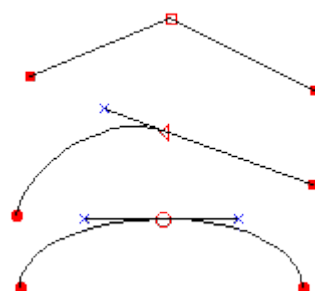


図4.3_2

7 SVG = Scalable Vector Graphics。縮尺可能なベクトル・グラフィックス。

曲線の向きは任意の方向を取ることができます。「角の点」は FontForge では「四角い箱 □」で表されます。二番目の例の交点は「接点」(tangent point) と呼ばれ、ひとつの三角形「△」で表されます。三番目の例では交点は円形「○」で表されます。

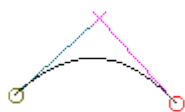


図 4.3_3

FontForge は (TrueType フォントで使用される) 二次のスプライン曲線を編集することもできます。この場合には制御点はひとつしかなく、曲線両側の終端点における傾きを制御します。二次スプラインは三次スプラインほど自由に折り曲げることができないために (図 4.3_1 の上側の曲線を二次スプライン曲線で表すには二本以上のスプライン曲線に分ける必要があります)、同じ曲線を得るためには、より多くのスプライン曲線を必要とすることになるでしょう。フォントが三次スプラインを含むか、二次スプラインを含むかは [エレメント(L)] ⇒ [フォント情報(F)...] コマンドの「レイヤー」項目で指定できます。

世界にはさまざまな種類のスプライン曲線が存在します。上で説明したものは「ベジェ曲線」(Bézier splines) と呼ばれるものです。この曲線は数値的に単純で、多くの優れた特性を持っており、そのため、フォントの内部で使用されています。しかし、さらに優れたスプライン曲線もいくつかあります。ベジェ曲線はどちらか制御点にも同じ傾きを与えることは簡単にできますが、曲線はどちらの制御点も同じ曲率を持っているとより美しく見えます。「クロソイド曲線」(Clothoid splines) にはこの特性があります。

ソフトウェア開発者のラフ・レヴィン Raph Levien 氏は、クロソイド曲線をベジェ曲線に変換するためのライブラリをオープンソース・コミュニティに寄付してくれました。

クロソイド曲線はベジェ曲線よりもはるかに簡単に操作できます。制御点はなく、曲線上にいくつかの点だけがあり、その点の間に魔法のように美しい曲線が表示されます。FontForge は、曲線をベジェ曲線としても、あるいはクロソイド曲線としても編集できます。

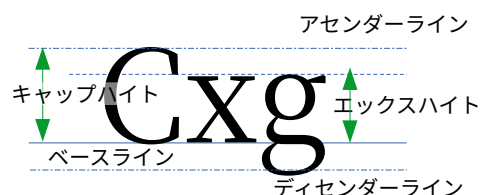
4.4 線、点、パス

パス〔輪郭線の経路〕は、最初の点に戻って来て始点と終点が結合した時に閉じられます。アウトラインフォントに含まれるすべてのパスは閉じていなくてはなりません。パスが閉じられたとき、そのパスは向きを持っていると呼ぶことができます。言い換えると、基本的には「パスは時計回りに作られたか?」、それとも「反時計回りに作られたか?」と考えることができます (この見方は実際には少し単純すぎます。もしパス自体が交差していて「8」の字形になっていたならば、その方向は時計回りとも反時計回りとも言い切れません。しかし殆どのパスは交差しません〔し、フォント内のパスは**交差すべきではありません**〕から、私達はパスを時計回りまたは反時計回りと呼ぶことにして人生を簡単にすることにしましょう)。

グリフのアウトラインを描画する時には、アウトラインの内側はすべてを黒く塗りたいわけですが。でも上の「A」の字を見てください。主要な外側のアウトライン・パスの内側に二本のパスがあり、その内側の二本のパスの内部は黒ではなく白くしたいのです。どうすればこれらを区別することができるのでしょうか? それにはパスの方向によって区別を行なう

のです。グリフ内の任意の点を考えましょう。その点から無限遠まで（任意の方向へ向けて）想像上の直線を引き、その線がパスに交差する回数を数えるならば、時計回りのパスと交差するごとに回数に「1」を加え、反時計回りのパスと交差するごとに「1」を引きます。結果として得られた回数が「0」でなければその点を黒く塗り、そうでなければ白いままにします。このやりかたで、上の「A」の文字がうまく塗分けられるようにするためには、外側のパスを時計回りに、内側のパスを反時計回りに設定します（逆周りにしてもうまくいきますが、慣習として外側のパスを時計回りにします）。

これらの点はすべて（終端点と制御点のどちらも）数学的に表現されています。各点は、グリフの「ベースライン」（「ディセンダー」を持たない文字が置かれている線）上に原点を持つデカルト座標系〔直交座標系〕に置かれています。ほとんどのフォントフォーマットは、座標値が -32768 から 32767 の間の整数値であることを想定しています。FontForge は内部的には実数を使用していますが、フォントファイルの生成時には（通常は）最も近い整数への丸めを行ないます。



一般的には、整数座標に限定することはよいことです。TrueType フォントは整数座標のみしか扱うことができないため、整数以外の値を使用すると、FontForge はそれを整数に丸めざるを得ず、それにより曲線の形状が変更されてしまいます。PostScript フォントでは整数以外の値を扱えますが、さまざまな形式が整数用に最適化されていますので、整数が使える場合には整数を使いましょう。さらに精度が必要な場合は、フォントの「em サイズ」を大きくすることができます。

4.5 Em ユニット

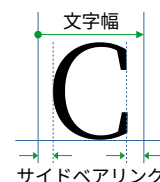
フォントを表示するときには、フォントのサイズを表現するために「em」という語を使います。あるフォントが 12 ポイントで表示されている場合、その時 1 em は 12 ポイントとなります。そのフォントが 18 ポイントで表示されると、em は 18 ポイントとなります（伝統的には、12 ポイントのフォントは字面の上下の高さが 12 ポイントである金属活字の塊——あるいは、行間を空けずに組んだテキストのベースラインとベースラインの間隔が 12 ポイントのこと——を指していました。ですから、em とは活字片の大きさだったのです）。

アウトラインフォントをデザインするときには、もはやポイント単位で物考えるわけにはいきません——アウトラインフォントは伸縮自在であり、どんなポイントサイズにも拡大・縮小できるのです。したがって「em」「em の分数」で考える必要があります。フォントを作成する場合、1 em が内部的なサイズの単位（これを「**em ユニット**」といいます）で何ユニットにあたるかを設定しなければなりません。ほとんどの PostScript フォントでは、em を 1000 ユニットとしており、TrueType フォントでは em を 1024 または 2048 ユニットのどちらかとすることがよく見られます（TrueType では 2 の累乗 2^n にすることを推奨しています）。しかし、いくつでも好きな（正の）数値を選択することができます（ただし、32767 よりも小さくなくてはなりません）。上記の数値は習慣的な物であり、必須事項ではありません。

たとえば PostScript フォントでダッシュ (-) を作り、その幅を 500 ユニットに設定した

ときにそのフォントの em が 1000 ユニットだったとしましょう。誰かがそのダッシュを 12 ポイントで表示したときに、そのダッシュは $500/100 * 12 = 6$ ポイントの長さをで描画されます。72 dpi, つまり 1 インチあたり 72 ドットの解像度をもつ画面では、1 ピクセルがほとんど正確に 1 ポイントとなるので、そのダッシュは 6 ピクセルの長さになるでしょう。

あらゆるグリフはそれ自身の座標系をもっています。フォントの「ベースライン」（ほとんどのラテン文字の下端となる仮想基準線）は垂直座標が 0 となる位置です。水平方向の原点は、グリフの描画が始まる点です（描画のし始めは何も「描画」されないのが普通です——なぜかと言うと、通常は原点とグリフの端との間にある程度の空白があるからです）。原点とグリフの左端とのこの水平距離のことを「**左サイドベアリング**」と呼びます（この値は 0 より小さい場合も、大きい場合も、ちょうど 0 の場合もあります）。すべてのグリフには「**文字幅**」（「**送り幅 advance width**」と呼ぶこともあります）があり、幅は現在のグリフの原点から次のグリフの原点までの移動距離です。グリフの右端と横幅の間の距離は「**右サイドベアリング**」と呼びます。



CJK（中国・日本・韓国）のテキストは縦にも横にも書くことができますので、CJK のフォントには通常、より一般的な横書き用の送り幅に加えて、縦書き用の送り幅（および上部と下部に付随するサイドベアリング）を備えています。

4.6 アセンダーとディセンダー

「g」や「y」のような文字のグリフにはベースラインよりも下に延びる部分があり、これを「**ディセンダー**」と呼びます。同様に、「k」や「l」のような文字のグリフにはエックスハイトよりも上に延びる部分があり、これは「**アセンダー**」と呼びます。ほとんどのラテン文字（やギリシャ文字及びキリル文字）のフォントにはいくつかの文字高さの基準線があります：アセンダーのない小文字の高さは「**エックスハイト**」（x-height）と呼ばれます。大文字の高さは「**キャップハイト**」と呼ばれます。アセンダーの高さは「**アセンダーハイト**」と呼ばれます（すべてではありませんが、一部のフォントはアセンダーと大文字がほぼ同じ高さで並びます）。通常は、これらはすべてのグリフで正確に同じ値ではなく、大文字「O」は大文字「I」よりもわずかに高くなっていますが、どちらの高さも、フォントのキャップハイトの範囲内に収まっています。

フォント自体には、それに付随する「高さ」（アセント）と「深さ」（ディセント）が定められています。昔の金属活字では、どの線も活字本体の「高さ」値や「深さ」値よりも上や下にはみ出すことはできませんでした。現在では、アクセント記号やアセンダーをフォントの「高さ」より上にしたり、ディセンダーをフォントの「深さ」よりも下にしたりすることができますが、今でもこの概念を残しておくのが便利です。ほとんどすべての場合で、グリフはフォントの「高さより上」や「深さより下」にはみ出すことがないのは事実なのです。この「高さ」と「深さ」の合計がフォントのサイズです。ひとつの金属活字のポイントサイズは、この値によって決まります（必然的に、グリフを保持している金属ブロックの高さとなります）。PostScript フォントでは、フォント内部のローカル座標系は、フォントが描画

される最終的なサイズには関係がありません。⁸

5. フォント：TrueType、PostScript、OpenType、SVG

FontForge はさまざまなタイプのフォントを出力することができます。それらはどう違うのでしょうか？

PostScript は **三次スプライン曲線** 技術に基づいていますが、TrueType は **二次スプライン曲線** 技術に基づいています。一般に、編集が行ないやすいのは三次スプライン曲線のほうです（より多様な形を表現することができます）。どの二次スプライン曲線も基本的に損失なしに三次スプライン曲線に変換することができます。三次スプライン曲線は任意の精度で変換することができますが、ほとんどの場合わずかな精度の低下があります。⁹

FontForge の内部では、フォントに三次または二次のスプライン曲線を使用することができます。それらは、フォントが生成される時に適切なフォーマットに変換されます。

この二つの曲線のもうひとつの大きな違いは、ポイントサイズが小さくなった時にどのように美しい字形を保つのかという問題の解決方法にあります。PostScript ではこれを「**ヒントづけ Hinting**」と呼び、TrueType ではフォントへの「**命令 instructing**」と呼びます。

5.1 OpenType とは何か？

残念なことに「OpenType」という語には二つの意味があります。

第一の意味としては、フォントのグリフ（とスプライン曲線）のフォーマットのことです。この意味では、OpenType は単に PostScript フォントを TrueType フォント・フォーマットに埋め込んだものにすぎません——外から見ると、アウトライン記述が TrueType 方式でなく PostScript Type2 フォント方式を用いていることを除いては、TrueType フォントと同様に見えます。技術的な言い方をすれば、「sfnt¹⁰ ラッパー内にある CFF¹¹ フォント」ということです。

第二の意味としては、OpenType という語は高度な組版に関する情報を含む一連のテーブルのことを指します。このテーブルは、二次スプライン曲線（TrueType）で記述されたフォントにも、三次スプライン曲線で記述されたフォントにも追加することができます。

最後に、MS Windows が OpenType アイコンを使って表示するのは「DSIG¹²」（デジタル署名）テーブルを持つフォントです。

8 《旧日本語版には以下の追記あり》「PostScript では慣習的に高さや深さの和を 1000 ユニットにすることがなっています。TrueType では通常は 2 の冪乗の値（多くは 2048）を使用します。」

9 《旧日本語版には以下の追記あり》「これは、TrueType のアウトラインを PostScript アウトラインに変換することは容易であるが、PostScript から TrueType に変換するのはより困難であることを意味します。」

10 sfnt = spline font または scalable font の略。

11 CFF = Compact Font Format の略。

12 DSIG = Digital SIGNature の略。

更に話をややこしくさせることには、OpenType の仕様には昔の TrueType の仕様が含まれているので、どの TrueType フォントも技術的には OpenType フォントと呼ぶことができるのです。

FontForge は、DSIG テーブルに対応していません（必要な場合は、DSIG テーブルを追加する MS の無料ツールがあります）。FontForge は、フォントを生成するときに、sfnt ラッパー内のフォントが PostScript（CFF）形式であることを示すために「OpenType（CFF）」という用語を使用します。OpenType タイポグラフィ・テーブルを含めるには、**【オプション】** ボタンを押して、**【☐ OpenType** チェックボックスを選択します*。

《※ 訳注：〔20230101 バージョンの場合〕 メニュー項目「ファイル」⇒「フォントを出力 Generate Fonts」⇒「オプション」ボタン⇒「☐ OpenType の仕様」関連チェックボックス、という流れになるのではないかと思います。》

5.2 SVG とは何か？

SVG（Scalable Vector Graphics、伸縮可能ベクトル画像）は、XML¹³ の一方言で書かれた比較的新しいフォントフォーマットを提供しています。SVG フォントでは三次か二次どちらかの曲線を使用することができます（両方を使用することもできますが、FontForge の機能制限で、フォント全体を三次か二次のどちらかに揃える必要があります）。SVG では二種類の異なる型のフォントが使用可能です。ひとつは PostScript Type1 フォントとほぼ同じで、もうひとつは、PostScript Type3 フォントとほぼ同じです¹⁴。

5.3 Type3 フォントとは何か？

PostScript の初期の頃、Type1 フォーマットは著作権で保護されており、Adobe だけが Type1 フォントを作成することができました。しかし、Type3 フォントは誰でも作成できました。Type3 フォントにはヒント機能がありませんでしたが、任意の PostScript コマンドを含めることができたので、より広い範囲の図形（多色のグリフ、ストローク表示のグリフ、画像の拡大縮小によるグリフなど）を生成することができました。ほとんどの場合、こうした拡張機能は不要で、ヒント機能が無いという欠点は煩わしいものでした——しかし、時にはこうした機能を使ってみたいと思うこともあるでしょう。¹⁵

5.4 Type1 フォントとは何か？

Type1 フォントは、1980 年代と 1990 年代の標準的な単独で動作する PostScript フォントでした。

13 xml = extended markup language 「拡張マークアップ言語」

14 《旧日本語版には以下の追記あり》「（SVG フォントでは画像が使用できない点が異なります——使用できるとしても、私にはそうは読み取れませんでした）」

15 《旧日本語版には以下の追記あり》「FontForge を複数レイヤー機能つきでコンパイルした場合、Type3 フォントを作成することができます。」

5.5 Type2 フォントとは何か？

Type2 フォントは、Type1 フォントと同じ情報を表現するのに、僅かに異なる内部フォーマット（より圧縮できます）を使用します。このフォーマットは、OpenType フォントの内部に存在する CFF フォントで用いられています。

5.6 CFF フォントとは何か？

CFF フォントとは、PostScript Type2 のフォント情報を格納するために作られたフォント・フォーマットです。OpenType (PostScript) フォントでは最も一般的なものです。

5.7 Apple 高度組版機能 (ATT) とは何か？

これは、タイポグラフィ情報（合字、カーニング、グリフ置換など）を指定する別の方法です。OpenType フォントよりも前に登場し、Apple コンピュータでのみ使用されます。この機能は「ステートマシン¹⁶ state machines」に基づいており、OpenType よりも操作が少し難しくなります。

6 ヒント情報

小さなピクセルサイズ〔画素〕では、アウトラインフォントの内部を描画するプロセスをうまく処理するのは非常に困難です。PostScript と TrueType では、小さなフォントを描画する方法に関する追加情報をどのようにラスタイザーに提供するかという問題に対して異なるアプローチをとっています。

TrueType へのヒント情報追加によるラスタイズ処理の改良。
Windows でラスタイズした 18 ポイントのフォント (拡大率 2 倍)

HI HI
ヒントなし ヒントあり

6.1 Postscript

Adobe は、処理をどう行なうかに関するヒント情報をラスタイザーに与えるために、いくつかの異なる方法を組み込みました。その中の最も分かり易いものは水平と垂直のステムヒントです。各グリフには、どこにステムがあるかをラスタイザーに伝えるための、そのグリフ独自のヒント・セットを持っています。FontForge では、通常、妥当と思われるステムヒント・セットを検出しますが

16 状態機械 state machines： 外部イベント（入力条件）と現在の状態に応じて処理を切り替えて次の状態が決まるように動作させる論理回路構造。

（「[自動ヒントづけ](#)」と呼ばれる処理です）、一部のグリフでは字形が複雑すぎて、好ましくないステム情報の選択をすることがあります。FontForgeはこの自動選択を上書きできるようなモードも提供しています。

Adobe の方式には、くぼんだ「**セリフ**¹⁷」を扱う「flex ヒント」というものもあります。これは、水平線（または垂直線）から僅かにずれているだけの曲線を、小さなポイントサイズでは直線として表示し、大きなポイントサイズでは曲線として表示するというものです。FontForge は、適切だと判断した時にこれを使用します。



ヒント情報の最初のバージョンでは、ヒント同士が重なり合うことができませんでした。つまり、ほとんどのセリフは記述することができなかったということです。Adobe はその後「ヒント置換」という方法を開発し、グリフの異なる部分ごとに重なり合わないヒントのセットを使用可能にしました。それでも完全にヒントづけが行えない図形は存在しますが、ヒント置換はひとつの改善でした。コマンドの **【ヒント(I)】⇒【自動ヒント(H)】** を使用すると、ステムの重なり合いを検出し、現在のヒント・セットをどう変えるべきかを表示します。 **【ヒント(I)】⇒【ヒントが置換する点(S)】** コマンドでも、どこでヒントの変更が起こるかを表示し、 **【点の情報】** ダイアログの **【ヒントマスク】** タブでユーザがそれを直接設定できます。

ごく最近〔2015 以前?〕になって、Adobe はグリフ内の「**カウンター**」（ステム間の空き）¹⁸を制御する手段を提供しました。オリジナルの Type1 記法では、「m」の水平カウンターや「E」の垂直カウンタをサポートしていましたが、より複雑な形を取り扱うことはできませんでした。（Type1 フォントの）カウンター・ヒントは、非ラテン・非ギリシャ・非キリル文字のグリフセットでのみ使用することができます。Type2（OpenType）フォントでは、事情はもう少し複雑です。FontForge は、適切だと判断した時にはカウンター・ヒントを生成します。



Adobe はフォント単位のヒントづけ機能をいくつか提供しています。そのいちばん分かりやすいふたつのものは、 **【フォント情報】** の「**プライベート辞書**」に含まれる BlueValues と StemSnap 設定です。BlueValues は、垂直方向の領域のリストを提供しますが、この領域は（ラテン文字のフォントでは、この領域にはアセンダーハイト、キャップハイト、エックスハイト、ベースラインおよびディセンダーハイトが含まれているはずですが）興味深い事が起こるところです。それは、小さなポイントサイズでは、これらの領域のいずれかに入ったものはどれも同じ高さに揃えられますが、より大きなポイントサイズでは、高さにはわずかな差がつけられているのです（例えば、「o」と「x」とは、通常わずかに異なる高さを持っていますが、小さなポイントサイズでその違いを表示すると汚く見えるのです）。同様に、StemSnap 変数は、フォントの標準的なステム幅を指定します。FontForge は、この変数に適切な値を推定しますが、 **自分で値を設定**することもでき、その場合は推

17 セリフ serif：アルファベットの画線の端にある小さな飾り。

18 カウンター counter：文字の中で、筆画線で完全にもしくは部分的に閉じられた空白部分。

定値よりも優先して用いられます。プライベート辞書とその機能に関するより総合的な説明については、Adobe の『Type1 フォントの仕様書』を参照してください。

FontForge の【自動ヒント】コマンドは、BlueValues が設定されているとより適切に動作します。ですから、自動ヒントづけを実行する前に、【エレメント(L)] ⇒ 【フォント情報(F)...】 ⇒ 【PS Private 辞書】でこの配列に値を設定してください。自動ヒントづけが終わった後、StemSnap を再生成する必要があるでしょう（これも、【エレメント(L)] ⇒ 【フォント情報(F)...】 ⇒ 【PS Private 辞書】で行ないます）。

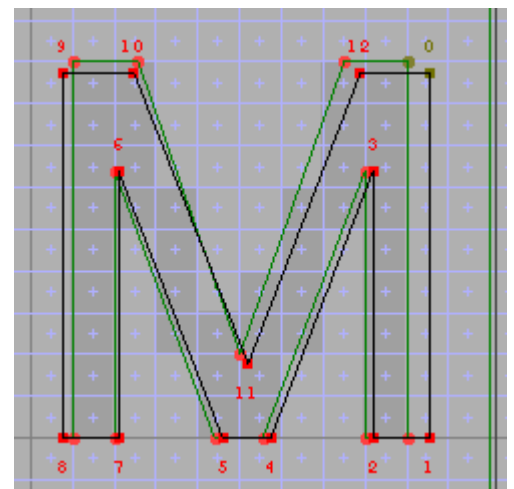
6.2 TrueType

一方、TrueType では、各点とスプライン曲線が正しい場所（すなわちピクセル境界線上）だけに置かれていれば、どのピクセルを塗りつぶすのかの判定はずっと単純になると述べています。そのため、TrueType では、指定されたサイズにおいて正しい位置になるまで各点を移動するための命令セットを用意しています。

FontForge の TrueType ヒントづけ（正確には「命令づけ instructing」と呼ばれます）の機能は、PostScript でも用いられているステム・ヒントと対角ヒントにに基づいています。これらのヒントは、どのポイントをどこに移動するか決定するのに用いられます。¹⁹

FontForge は、TrueType フォントを読み込むときに使用されている命令を格納し、次にそのフォントを出力するときに、（グリフに重大な変更が発生していないときに限り）その命令を用います。

FontForge の「ヒントづけユーザインタフェース」のほとんどは、PostScript フォントを念頭に置いて設計されています。



M のグリッド合わせ

7. 参照

グリフはアウトライン（スプライン曲線）に加えて、他のグリフへの参照によっても組み立てることもできます。これによって、特にアクセント付きのグリフを手軽に作ることがで

19 《旧日本語版には以下の追記あり》「それは直角方向ステムの概念をサポートし、それらにヒントづけを行ってそれらのステムがほぼ同じ幅をとるようにします。それに加えていくつかの追加の作業として、垂直ステムに接合したセリフを検出し、それにも同様にヒントづけを行います。ヒントづけ機能には、PostScript の BlueValues の等価物を、小さなポイントサイズでグリフを同じ高さに強制的に揃える cvt エントリを生成するのに使います。私は、TrueType 用に生成している命令の品質に満足していません。」

きます。例えば、文字「à」はグリフ「a」への参照と、別のグリフ「`」への参照から組み立てることができます。この結果、大幅な容量の節約となり、デザイン作業が楽になります（グリフ「à」でグリフ「a」への参照を行なった後で、もとのグリフ「a」を変更した場合には、その変更はグリフ「à」に自動的に反映されますが、参照ではなく単にグリフ「a」を「à」にコピーしていただけた場合は、再び変更したグリフ「a」のコピーを行なわなければなりません）。

FontForge は参照を取り扱う三種類のコマンドを持っており、アクセントつきグリフを作成するための、より高水準のコマンドをひとつ備えています。【編集(E)】⇒【参照をコピー(Q)】コマンドは、現在のグリフへの参照をクリップボードにコピーし、次に【編集(E)】⇒【貼り付け(P)】コマンドを使えば参照を貼り付けることができます。【編集(E)】⇒【参照を解除(N)】コマンドは、別のグリフへの参照を、そのグリフを構成するスプライン曲線に変換します。【エレメント(L)】⇒【情報を見る(I)】コマンドは、(参照に適用されたときは) どのグリフが参照されているかを示すダイアログを表示し、そこから、そのグリフに対する編集ウィンドウを呼び出すことができます。そのグリフへの参照を部品となるスプラインとして別のグリフに張り付けることができます。【エレメント(L)】⇒【組み立て(U)】⇒【アクセントつきグリフを構築(B)】コマンドは、この複合グリフを構築するのにどのグリフを参照するべきかを検出し、それぞれのグリフへの参照を行ない、適切な位置に配置します。たとえば「à」を選択し、【エレメント(L)】⇒【組み立て(U)】⇒【アクセントつきグリフを構築(B)】コマンドを実行した場合、FontForge は「a」への参照と「`」への参照を行ない、アクセントをグリフの上に置き、中央に揃えることになります。

一部のビットマップ形式（知る限りひとつだけですが）は参照機能に対応しているので、FontForge ではアウトライン・グリフだけでなくビットマップ・グリフでも参照が可能です。

悲しいことに、人生はそれほど単純ではありません。FontForge が内部的に使用している参照は、出力フォントで必ずしも同等のものに変換できるとは限りません。一般的には、これは問題とはなりません（FontForge は単に出力内の参照リンクを解除するだけなので、それによりフォントが少しだけ大きくなります。TrueType への命令づけ [= ヒントづけ]、および、比較的度は低いですが、PostScript のヒントづけに影響があるようです。ほとんどの方はこれを無視されて結構ですが、興味をお持ちの方は[このリンク](#)先をお読みください。

8. レイヤー

フォントには複数のレイヤーがある場合があります。FontForge が新しいフォントを作成すると、ふたつの基本レイヤー（前面レイヤーと背面レイヤー）と、すべてのグリフで共有されるガイドライン用のミニ・レイヤーのようなものを持ちます。通常のフォントでは、すべてのグリフのレイヤー数は同じです（これは Type3 フォントには当てはまりません。これについては次項「Type 3 フォント」で説明します）。

前面レイヤーと背面レイヤーには、両方とも予め定義された意味があります。**前面レイヤー**は、レイヤーを指定する明確な方法がないダイアログのデフォルト「既定」レイヤーです。**背面レイヤー**には、あらゆる種類のもの、基本的には何でも含めることができますが、その中の画像はすべて前面レイヤーに自動トレースされます。

【**エレメント(L)**】⇒【**フォント情報(F)...**】⇒「**レイヤー**」タブで他のレイヤーを追加できます。各レイヤーには名前を付け、異なるスプライン曲線（二次または三次）を取り込みます。

追加レイヤーの潜在的な用途のひとつ：多くの人は三次スプライン曲線を使用する方が編集が簡単だと感じていますが、二次スプライン曲線を使用して TrueType フォントを作成することを好みます。ひとつのレイヤーを三次スプライン曲線の保持に使用し、別のレイヤーを二次スプライン曲線の保持に使用できます。レイヤーどうしを比較したり（【**エレメント(L)**】⇒【**レイヤーを比較...**】）、あるレイヤーから別のレイヤーにコピーし（新しいレイヤーに適合するようにスプライン曲線を修正く【**編集(E)**】⇒【**レイヤーからレイヤーにコピー**】くしたり）するコマンドがあります。

データベースに有効な三次曲線フォントと有効な二次曲線フォントの両方がありながら、そのうちのどちらかからしかフォントを生成できないのは残念に思えますので、「フォントビュー」ウィンドウ〔フォント一覧画面〕ではレイヤーを切り替え（【**表示(V)**】⇒【**レイヤー**】⇒【（表示するレイヤーを選択）】）、【**ファイル(F)**】⇒【**フォントを出力(G)...**】コマンドで任意のレイヤーをフォントに出力できます（すべてのレイヤーが同じヒント情報、命令づけ、文字セット、ビットマップなどを共有することに注意してください。異なるのはアウトラインだけです）。さらにいえば、【**フォントを出力(G)...**】画面の【**オプション**】ボタンにある「PfaEdit テーブル」の「☐ **Save Layers**」（すべてのレイヤーを選択）を使用して、フォントのすべてのレイヤーを sfnt（TrueType / OpenType フォント）に保存することを選択できます。

8.1 Type3 フォント

ほとんどの人は Type3 PostScript フォント（または同等の SVG フォント）の編集に興味がありませんが、その気になれば、FontForge で編集できます²⁰。編集する場合、各レイヤーには異なる描画操作が表示され、異なる色やストローク〔筆画〕、塗りつぶしなどを指定できます。レイヤーはフォント幅ではなく、各グリフには（名前のない）独自のレイヤー・セットがあります。「アウトラインウィンドウ」にあるレイヤー・パレットを右クリックして表示されるポップアップ・メニューの【**New Layer**】を使用して、新しいレイヤーを作成できます。

8.2 Guidelines

FontForge では、ガイドラインは独自のミニ・レイヤーにあります（このレイヤーはフル・レイヤーではありません。というのも、グリフが個々にレイヤーを持っているのではなく、すべてのグリフが同じガイドラインを共有しているからです）。ガイドライン・レイヤーは他のレイヤーと同様に編集でき、お望みの形状のガイドラインを作成できます。また、アウトライン・ウィンドウの端から水平および垂直のガイドラインを引き延ばすこともできます。

²⁰ 詳しくは Book 2「ユーザー・インターフェース」の章にある「ダイアログ：Type3 および SVG フォントの複数レイヤー編集」の項を参照してください。

9. 用字系

用字系 (script) とは、字体と、文字を組み立てるルールの集合体 [コレクション] です。例えば、ラテン文字は文字「A」から「Z」までから成り、左から右に書かれ、大文字から小文字への変化があり、少数の合字およびアクセント文字を生成するルールが存在する用字系です。アラビア文字は独自の文字セットからなり、右から左に書かれ、語頭形、語中形、語尾形および独立形の変化があり、多数の合字および母音記号に対するルールがあります。

10. グリフ名と名前リスト

10.1 グリフ名

FontForge 内部では、どのグリフにも名前があります。一般に、「**グリフ名**」はそのグリフがどのように見えるかを表す情報を提供するものであり、例えば「9」を表すグリフは「nine」という名前です。Adobe は、**各グリフにどのような名前を割り当てるかを定めた仕様書**を定めており、一般的な用途のフォントを出力する時にはその標準を厳守するのが最良の方法です (Acrobat はグリフ名を使用しており、標準のグリフ名を認識します。もし「nine」の代わりに「neuf」と名付けてみると、Acrobat はそれを認識せず、予期しない事象は機能しません)。

Adobe の標準では、ほとんどのラテン文字とギリシャ文字のグリフにまあまあ判り易い名前 (英語の話者にとっての話ではありますが) をつけていますが、キリル文字とヘブライ文字に対してはほとんど訳の分からない名前 (afiiXXXXX) をつけられており、それ以外のほとんどのグリフは Unicode の符合位置で識別されています (uniXXXX または uXXXXX という形で、「X」は 16 進数を表し、XXXX は 16 進数の Unicode 符号位置です)。

あなたが英語の話者で、ラテン文字のグリフを作成しているなら、これは非常に便利でしょう。しかしあなたがフランス人ならば、「adieresis」 [= ä] には「atréma」という名前を付けてほしいと思うかもしれません——この件に関しては、英語の話者でさえ、「aumlaut」という名前の方を好むかもしれません。あなたが何者であれ、自分が作業中のコードブロックにはより覚えやすい名前を好むことでしょう。

10.2 名前リスト

FontForge には「**名前リスト**」という概念があります。このリストの各項目は、Unicode の符合位置から名前への単純な対応づけです。いくつかの名前リストを定義することができ、異なるフォントには異なるリストを適用することができ、ある名前リストから別のリストに切替えることによって、フォントに含まれるすべてのグリフを改名することができます。

名前リストはフォント内を移動するのにも役立ちます。リストが特定のフォントに適用されていない場合でも、そのリストにある名前は、**【表示(V)】** ⇒ **【移動(G)】** ダイアログでグ

リフを探すのに使用することができます。ですから、フランス語の名前リストを読み込んであれば、「atréma」を検索することができ、FontForge はグリフの名前が「adieresis」でも、「uni00E4」でも、「aumlaut」でも、その他の何であっても、そのグリフを表示します。

いくつかの注意点： Adobe のグリフ名標準は、すべてのグリフ名が ASCII 文字であることを前提としています。構文的には PostScript はより広い範囲のグリフ名を受け付けることができますが、ASCII 以外の文字を使用した場合、いくつかのラスタライザでは問題が起こる可能性があります。fontforge 内では非 ASCII 名を使い続け（フォントを生成する前に、グリフの名前を Adobe の標準名に変更する）のが最善です。

11. 検索、機能、スクリプト、言語

OpenType（および Apple 高度組版機能）を使用すると、入力グリフ・データにさまざまな変換を適用して、出力グリフ・データを生成できます。この変換には、合字の形成、グリフ同士を近づけるカーニング、文字への母音記号や発音識別符号の付与、周囲のグリフに応じた条件付きのグリフ置き換え、などが含まれる場合があります。

11.1 検索 Lookups

それぞれの変換用コマンドは、いわゆる「**検索**」に格納されています。ラテン文字の合字を扱う「検索」のセットを設計している場合、ほぼ常に発生する「ff」「fl」と「ffl」用の合字検索をひとつ、トルコ語では発生しないが他の言語では発生する「fi」と「ffi」の合字用をひとつ、それに英語では滅多に起こりませんがドイツ語ではより頻繁に発生する（と思う）「ct」と「st」用の合字検索をひとつ作成することになるかもしれません。

11.2 機能 Feature

「**機能**」とは、タグ付けされた「検索」の集合体〔コレクション〕です。「**タグ**」はすべて標準化された意味を持つ四文字の（「liga」のような）配列〔シーケンス〕です。

「liga」タグは標準的な「合字（ligatures）」を意味しています。「dlig」タグは「任意の合字（discretionary ligatures）」を意味し、「kern」タグは「カーニング（kerning）」を意味します。他にもたくさんあります。

11.3 文字 Script

すべての「**文字**」（ラテン文字、キリル文字、漢字〔表意文字〕など）には、それによって意味のある一連の独自「機能」があります。時には、状況はさらに複雑で、「文字」にも各言語ごとに独自の機能セットが存在することがあります。上記の例では、文字、言語、機能、検索を次のように割り当てます。

文字	言語	機能	検索対象
ラテン文字	デフォルト	liga (標準)	「ff」 「fl」 「ffl」
			「fi」 「ffi」
	トルコ語	liga	「ct」 「st」
			「ff」 「fl」 「ffl」 ^[1]
	ドイツ語	liga	「ff」 「fl」 「ffl」
			「fi」 「ffi」
			「ct」 「st」 ^[2]

[1] 検索は共有される場合があるため、これはデフォルトと同じ検索になります（トルコ語では「i」と「点なしのi」とが区別されますが、合字の「fi」はこの区別が隠されます）。

[2] 検索は異なる機能のタイプと共用されることもあります。英語とスペイン語では「ct」は「任意の合字」としてカウントされる可能性があり滅多に使用されませんが、ドイツ語ではより一般的であり、標準の合字セットと考えられているかもしれません。

「デフォルト」言語とは、その文字で特に言語名が指定されていない言語を意味します。上記の例では、「デフォルト」言語は、トルコ語とドイツ語を除いた、ラテン文字を使用する可能性のある任意の言語になります。

また、「デフォルト」文字ということも可能です。「0～9」の数字はラテン語だけでなく多くの書記体系で使用されており、ほとんどの文字に共通であると考えられています。数字に適用する検索設定（旧形式の数字、可変幅の数字、上付き文字、…への変換など）がある場合は、その検索設定を使用できるすべての文字（と言語）の「機能」に付加しておくべきです（念の為、デフォルト文字にも）。

文字	言語	機能	検索対象
ラテン文字	デフォルト	pnum	「可変幅の数字」 proportional number
		onum	「旧形式の数字」 old style number
		subs	「下付き数字」 subscript
	トルコ語	pnum	「可変幅の数字」
		onum	「旧形式の数字」
		subs	「下付き数字」
	ドイツ語	pnum	「可変幅の数字」
		onum	「旧形式の数字」
		subs	「下付き数字」

文字	言語	機能	検索対象
キリル文字	デフォルト	pnum	「可変幅の数字」
		onum	「旧形式の数字」
		subs	「下付き数字」
デフォルト	デフォルト	pnum	「可変幅の数字」
		onum	「旧形式の数字」
		subs	「下付き数字」

さまざまな情報を含む検索がおよそ 15 種類あります。さまざまな種類の検索とは、合字データ、グリフ置換データ、カーニング・データを指定するために必要です。

検索は、**【エレメント】** ⇒ **【フォント情報】** ダイアログで指定できます。

11.4 検索サブテーブル

上記では、検索が単純な事柄であるかのように説明しました。残念ながら、そうではありません。それぞれの検索は、実際の作業を実行する「サブテーブル」で構成されています。

検索には複数のサブテーブルが含まれることがありますが、たいていはひとつだけです。（検索にひとつもサブテーブルが含まれていない場合には、何も行なわれません。）

約 15 種類の検索があるといいましたが、サブテーブルにはさらに多くの種類があります。たとえば、グリフ間のカーニングを指定するサブテーブルは、グリフ・クラス間のカーニングを指定するものとは異なります。

一目見ただけでは、サブテーブルは不必要な複雑さのように見えるかもしれませんが、ひとつ検索に幾つものサブテーブル・データが含まれているほうが望ましいことがわかります。たとえば、カーニング箇所の検索には、グリフ・クラスのカーニング情報を提供するサブテーブルと、そのカーニング情報の適用除外指示をグリフごとに提供する別のサブテーブルがあるという場合です。

11.5 検索はどのように行なわれるのか

ワードプロセッサがテキストを取り扱う場合、どのような文字がテキスト内に用いられているのかを判断します。どういうわけか、ワードプロセッサは言語が何であるかを判断します（おそらく「デフォルト」言語を推定しているだけかもしれませんが、あるいはユーザーがテキストはフランス語であると伝えたのかもしれません、おそらくロケール〔言語地域設定〕がスペイン語に設定されているからかもしれません…）。

どの文字／言語でも、デフォルト〔初期設定〕で適用される機能もあれば、ユーザーの裁量で適用される機能もあります。英語では「liga」機能は常に適用する必要がありますが、「dlig」機能はユーザーが要求した場合にのみ適用します。（現時点ではすべてのプログラ

ムがこれに従っているわけではありませんが、それが原則です）。

ワードプロセッサは、アクティブ〔適用対象〕となる可能性のある機能のリストを作成します。次に、フォントを調べて、どの機能が使われている文字と言語で利用できるかを確認します。順番に、「検索」候補から各機能を構成し、すべての適用対象機能にあるすべての検索は「適用」としてマークされます。

検索は、検索リスト内の順序で実行されます（これは、**〔フォント情報〕** ⇒ **〔Lookups〕**〔検索〕タブに表示される順序でもあります。各「機能」がリストされた検索リストの順序とはかなり異なる場合があります）。すべての適用対象の検索が実行されます。

この実行順序は重要です。もし小文字検索と合字検索の両方が適用対象で、テキストが「fi」の場合は、おそらく検索の順序が重要となるでしょう。小文字検索が先に行なわれる場合、「fi」は「FI」²¹になり、合字機能は適用されません。合字検索が最初に行なわれる場合には、おそらく小文字グリフが存在しない合字の「fi」が表示されます。一方では小文字が得られ、他方では合字が得られます。おそらく、ここでは、小文字の検索を最初に実行することになるでしょう。

検索サブテーブルは生活をさらに面白くします。検索が実行されると、その検索内のサブテーブルのどれかが適用されるまで順番にアクティブ〔適用対象〕化されます。適用された場合は、それ以上のサブテーブルの適用は行なわれません。

したがって、上記のカーニングの例では、最初に特殊なグリフ間のサブテーブルを、次にグリフ・クラス間のサブテーブルに配置することになるでしょう。特殊なグリフの組み合わせが発生した場合、最初のサブテーブルが適用され（そのグリフの組み合わせにカーニングが実施され）、クラス間のサブテーブルは使用されません。特殊なグリフの組み合わせがなかった場合には、最初のサブテーブルは適用されず、二番目のサブテーブルが適用されます。

11.6 GPOS²² と GSUB²³

これまで検索リストというものがひとつあるかのように説明してきました。それは話を単純化したものです。検索にはふたつの広範囲なクラスとふたつのリストがあります。グリフ自体の操作に関するもの（グリフの置換、連結、分解など）と、他のグリフとの相対的な位置関係に関するもの（カーニング、マークの添付など）です。

「**GSUB テーブル**」にはすべてのグリフ置換検索が含まれており、それらはすべて位置検索が試行される前に処理されます。

「**GPOS テーブル**」には、すべてのグリフ位置検索が含まれています。

21 《訳注》「FI」と大文字表記になっていますが、文意からは小文字表記の「f」と「i」が小文字で（合字なしで）表示されるべきところではないかと推定。

22 GPOS = Glyph Positioning Table の略。フォントがもっているプロポーショナルメトリクスやペアカーニング情報によって字詰めを実現するためのテーブル。

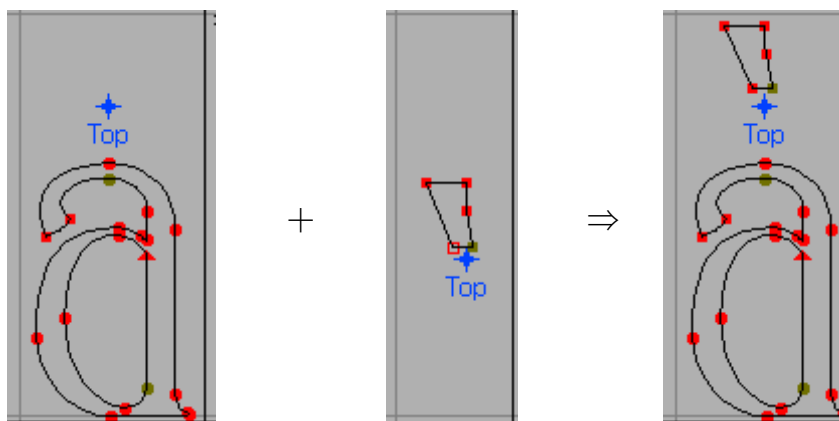
23 GSUB = Glyph Substitution Table の略。縦書きグリフへの変換や、合字への変換など、単一または複数のグリフを、別のグリフに置き換えるためのデータを格納。

12. アンカーポイントとアンカークラス

12.1 アンカーポイント

「アンカーポイント」を使うとふたつ（またはそれ以上）のグリフの配置を細かく調節することができます。アンカー〔碇の意〕に基づいた連結にはいくつか異なるスタイルがあります。筆記体、マークから基底グリフへ、マークから合字へ、そしてマークからマークへの位置指定です。最初のスタイルは、あるグリフから出る線が次のグリフの書き始めとなるような流麗な手書き文字（筆記体）においてグリフ同士をつなぐためのものです。二番目のスタイルでは、グリフ群の内のひとつが基底グリフとなって通常通り配置され、残りのグリフが典型的には発音付加記号〔ダイアクリティック〕や母音記号として、各グリフのアンカーポイント同士が重なり合うように配置されます。マークから合字グリフへの連結は、マークから基底グリフへの連結と同様ですが、マークが合字のどの要素に連結するののかによってそれぞれ異なる複数のアンカーポイントをもつことができます。最後に、マークからマークへの位置指定を使用すると、ふたつのマークを相互の位置関係によって指定することができます（ベトナム語やヘブライ文字など、ひとつのグリフに複数のマークを追加できる場合に必要となるでしょう）。

以上の説明はあまり上手ではありません。ここで以下のふたつのグリフを例に、それらが組み合わされてどのように最終的な結果をもたらすかを考えてみましょう。



グレイブ・アクセントは上にずらされてから右に移動し、そのアンカーポイントは“a”にあるアンカーポイントの位置に合わせられます。

FontForge はこれらのグリフの合成を行ないません（できるにはできますが、それは全く別問題です）。代わりに、テキスト・レイアウト・プログラムがグリフの合成を実行できるようにするための情報をフォントに追加します。

ほとんどのラテン文字フォントにはグレイブ記号付き「a」の合成済みグリフが既に作成されているので、上の例はあまり役に立たないでしょうが、同様の考え方は、より多くの異体字グリフを含み、合成済みのグリフが少ないアラビア文字やヘブライ文字のフォントにもあてはまります。

以上が、アンカーポイントがどう使われるかの簡単な解説です。その機構についてこれか

ら解説します。

12.2 アンカーポイントの仕組み

上記のさまざまな連結スタイル（筆記体、マークからベース、マークから合字、マークからマーク）はそれぞれ、異なるタイプの GPOS 検索です。連結箇所の位置決めを行なう場合は、まず適切な「検索」（**[エレメント(L)] ⇒ [フォント情報(F)...] ⇒ [Lookups] ⇒ [GPOS]**）を設定し、次に「検索サブテーブル」を、さらに検索サブテーブル内のアンカー・クラスを作成する必要があります。アンカー・クラスは、名前を付けることによって特定されます（上記の例では、クラスは「Top」という名前と呼ばれていました）。

筆記体の連結は、他の連結スタイルとは少し異なります。各グリフには、入口と出口という二つの連結点があります（上記の「マークからベース」への連結例に示されているように連結点がひとつだけではありません）。テキスト・レイアウト・プログラムは、グリフの出口連結点を次のグリフの入口連結点に接続します。

グリフには複数の「**アンカーポイント**」を含めることができますが、「マークからベース」のアンカー・クラスに対してはひとつだけです（合字にはひとつのクラスに複数のアンカーポイントがある場合があります、「curs」クラスと「mkmk」クラスには通常ふたつあります）。グリフビュー内（アウトラインウィンドウ）では、**[点(P)] ⇒ [アンカーを追加(A)]** コマンドを使用してアンカーポイントを作成できます。この「アンカーを追加」ダイアログでは、アンカー・クラスと、この点が基底グリフ内にあるか記号グリフ内にあるか、および位置を指定する必要があります。したがって、上記の例では、「a」のアンカーポイントは「Top」クラスを持ち、基底グリフにある一方で、「grave」のアンカーポイントもクラスは「Top」ですが、こちらは記号グリフにあります。「トップ」という名前のアンカーポイントを持つ基底グリフは多数存在し、同様に「トップ」というアンカーポイントを持つ多くの記号グリフが存在する可能性があります。「トップ」アンカーポイントを持つ基底グリフはどれでも、「トップ」アンカーポイントを持つどの記号グリフとも組み合わせることができます。このため、正確に構成された字形で可能なすべての組み合わせを指定する代わりに、同じ効果に対して遥かに少ないアンカーポイントを指定します。

しかし、実際には人生はそれほど単純ではありません。合字では、同じクラスのアンカーポイントが複数必要な（たとえば、各コンポーネントの上にひとつずつ）場合があります。アンカーポイントを作成するときには、それが合字内にあるべきだということかもしれません。その場合、合字インデックス（数字の「0」から「合字内のコンポーネント数-1」までの数値）も求められます。これにより、複数のアンカーポイントを同じクラスに配置できます。テキスト・レイアウト・プログラムは、最初のマークを最初のアンカーポイントの上に配置し、2 番目のマークを 2 番目のアンカーポイントの上に…、等々（これは大幅な単純化です）。

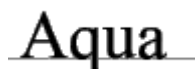
最後に、同じアンカーポイントにふたつの記号を配置する必要がある場合があります。たとえば、「**ā** (a_dieresis_macron)」グリフにはダイエレスिस〔分音符号 (¨)〕とその上にマクロン〔長音符号 (ˉ)〕があります。通常はどちらも「Top」アンカークラスに配置される記号です。これに対しては、ひとつの記号からもうひとつの記号への位置を指定する特

別なクラスを定義します。アンカークラスを作成するときに、「長音符号（マクロン）用の」クラス「mkmk」²⁴（マークツーマーク）のタグを指定し、「TopMark」のような名前を付けてください。分音符号（ダイエレシス）に対する新しいアンカーポイントを作成する時に、それが**基底の記号**「**ベース・マーク**」であると指定し、それをグリフの上部に配置します。ここで、長音符号（マクロン）に追加する新しいアンカーポイントはタイプ「mark」として、「Top」と同じ場所に配置します。このようにすれば、「ā (a_dieresis_macron)」グリフのように「分音符号+長音符号」が続くグリフ・データが入力された場合、テキスト・レイアウト・プログラムは「a」を通常どおりの位置に配置し、「分音符号」をその「Top」アンカーポイントが「a」の「Top」アンカーポイントと一致する場所に配置し、最後に「長音符号」をその「TopMark」アンカーマークが「分音符号」の「TopMark」アンカーマークと一致する場所に配置します。

アンカーポイントには、選択、ドラッグ、座標転換、切り取り、コピー、貼り付け操作が可能です。

13. ベースライン

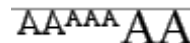
ラテン文字の使用者にとって、グリフの垂直方向の原点が**ベースライン**で、ほとんどの文字がその上に置かれていることは理にかなっています〔13.1 図〕。すなわち、グリフのポイントサイズを変更しても、正しく整列されることを意味します〔13.2 図〕。これはあまりにも自明なので、考えることすらしません。しかし、垂直方向の起点を文字の上部に置くと、文字の配置はまったく異なり、私たちの目には意外に見えます〔13.3 図〕。



13.1 図



13.2 図

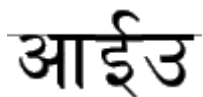


13.3 図

文字が異なれば、グリフをどのように配置するかについての期待値も異なります。ラテン文字（キリル文字、ギリシャ文字、アラビア文字、ヘブライ文字など）では、位置合わせ位置、つまりベースラインは、グリフの可能な範囲内——すなわち文字はベースラインより上にアセンダー、下にディセンダーがあり、その範囲——のどこかにあります。

しかし、すべての文字がそのようになっているわけではありません。「デヴァナーガリー文字」〔インド系文字〕では、整列位置はグリフの上部近くにあります。実際、ほとんどのグリフはベースラインを明示的に描画します〔13.4 図〕。文字のポイントサイズを変更する場合、グリフの底部に沿った下揃えでなく、ではなく〔13.5 図〕グリフをこのベースラインに沿って上部に並べます〔13.6 図〕。

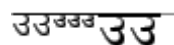
²⁴ mkmk = mark-to-mark の略。「記号対記号」の意味。



13.4 図



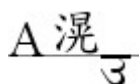
13.5 図



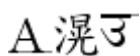
13.6 図

CJK〔中国・日本・韓国〕表意文字では、通常、ベースラインはグリフを包む正方形の下端に描画されます。

ただし、こうした異なる文字をそれぞれベースラインに合わせて一緒に組み合わせようとすると、混乱が生じます。



このような異なる文字が、グリフ同士が適度に組み合わせられるように、そのベースラインを異なる場所に持たせることができれば、一層綺麗に見えるでしょう。



Apple 組版技術と OpenType の両方でこれが可能です。各文字（または各グリフ）のベースラインを他のベースラインに対してどのように合わせるのかを示すベースライン・テーブルを持っているからです。

14. フォントビュー

「**フォントビュー**」²⁵は、フォント内のすべてのグリフ（またはウィンドウに収まるかぎりのグリフ）を表示します。各行にはふたつの部分からなり、上段は「文字ラベル」で（しばしばくっきりした文字が小さく表示されます）、その下にあるやや大きな文字は、あなたがデザインしたフォントを 24 ピクセルでラスターライズしたものです。「**アウトライングリフビュー**」²⁶（次項 15. を参照）で文字を編集すると、編集が進むにつれて、そのグリフのフォントビュー内の小さな表示も変更されます。

グリフをダブルクリックすると、そのグリフを編集可能なアウトライングリフビューが画面に表れます。²⁷

複数のグリフをセレクトすると、それらのグリフすべてにさまざまな操作を適用したり、それらのカット&ペーストが可能になります。

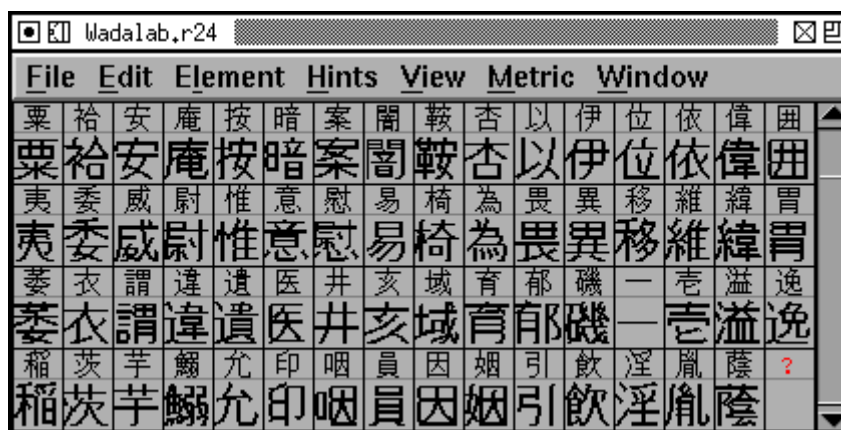
25 Font View： 「フォントビュー」は FontForge を起動するとオープニング画面の次に関くメイン・ウィンドウで、この日本語版では「フォント一覧」ウィンドウなどと表記しているものです。

26 Outline Glyph View： 「アウトライングリフビュー」は「アウトラインウィンドウ」とも表記されます。

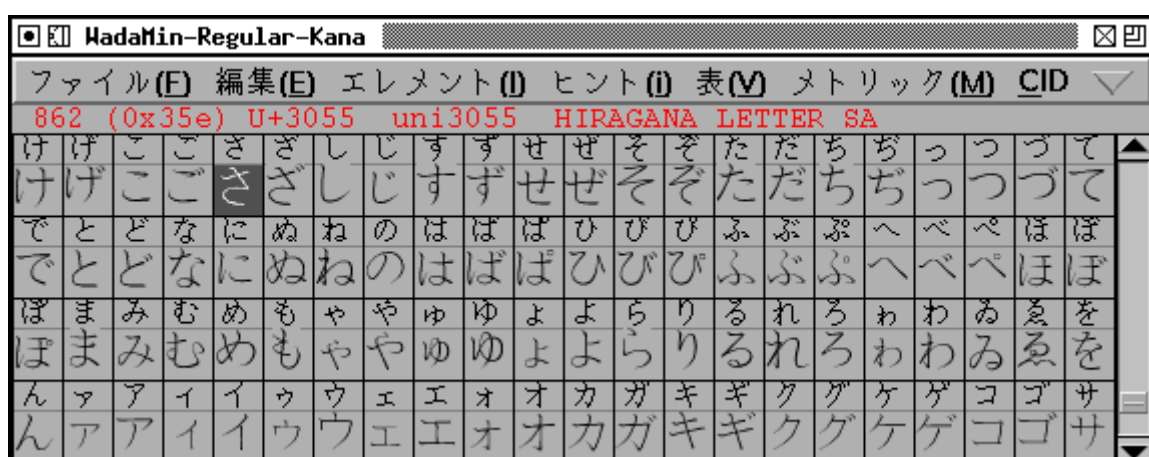
27 《訳注》ダブルクリックでアウトライングリフビューが開かない場合には、右クリックで【**アウトラインウィンドウを開く(U)**】を選択してください。



フォントは必ずしもラテンアルファベットだけを含むわけではありません。以下は、[和田研](#)のパブリックドメイン漢字フォントのうちの一部を表示したものです。



CID キー指定フォントとしてひとつにまとめられた別の和田研フォントです。

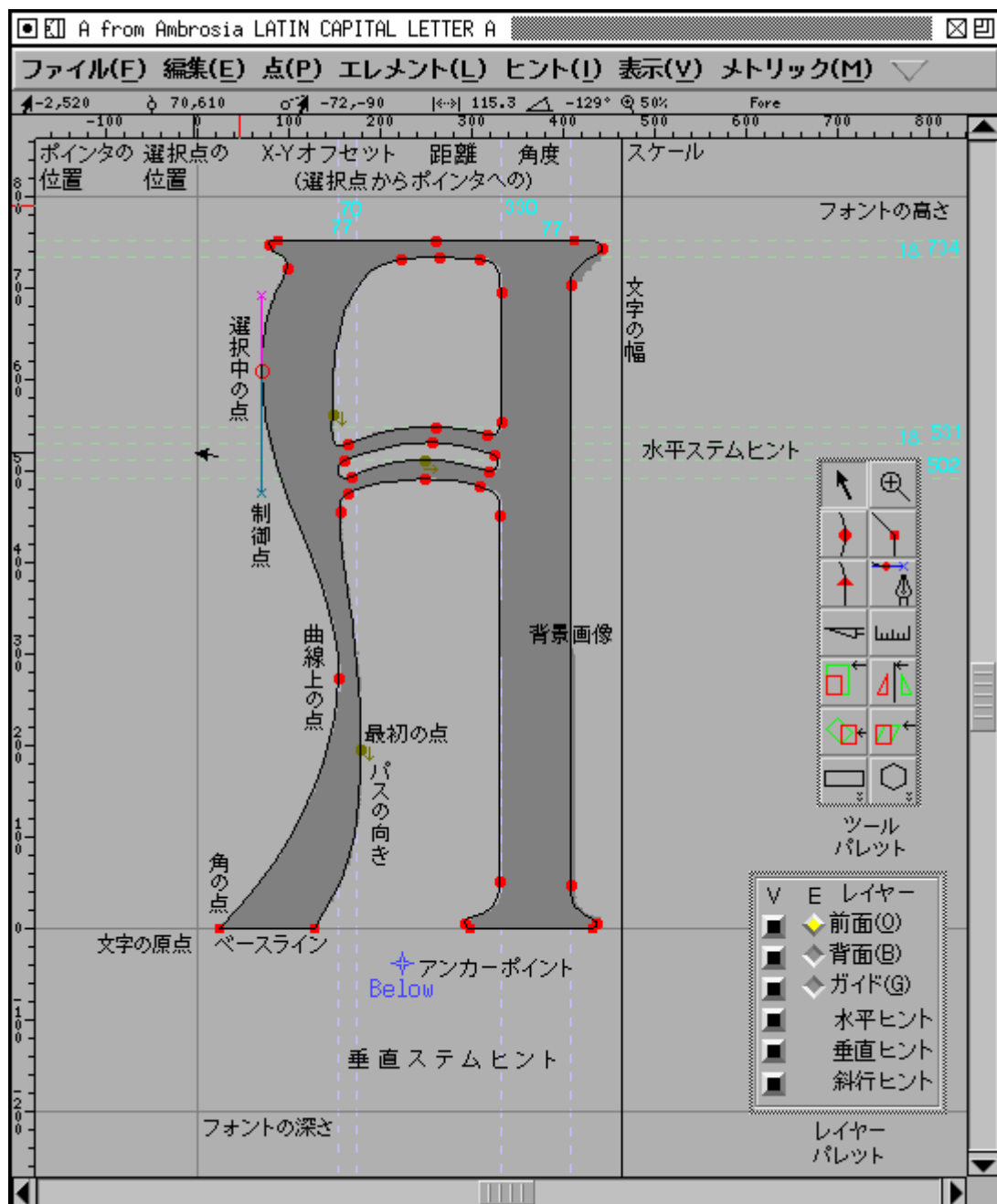


15. アウトライングリフビュー

「**アウトラインビュー**」は、おそらく FontForge で最も複雑な操作ウィンドウ〔データ表示・編集ウィンドウ〕でしょう。これが、グリフを構成する「**スプライン曲線** (Bézier スプライン)」を実際に編集できる場所です。

作業中の字形をトレースできるように、背景画像をビューに読み込み表示させることができます（灰色で表示されます）。多数の格子状の線が画面に見えますが、そのうち一部（ベースライン、アセントライン、ディセントラインおよび $x=0$ を表す線）はデフォルトで作成されたもので、その他のものは自分で追加することができます。

グリフのヒントを表示するレイヤーもあります。



前面にあるのはグリフそのもののスプライン曲線と点〔操作点〕、それにグリフの幅を示す線です（この線を動かすことにより、グリフの幅を調整できます）。点を選択しているときは、その制御点が表示されます。

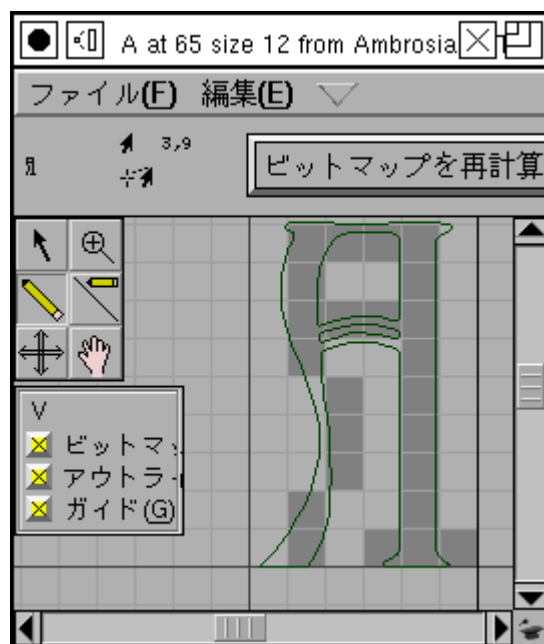
メインウィンドウの左側にふたつのパレットがあります（上図では右側に表示しています）。これは単に表示の邪魔にならないようにするためで、パレットは動かすことができます）。ひとつは「**ツールパレット**」で、もう一つのパレットは、どのレイヤー（前面、背面、グリッド、ヒント）を表示・編集可能にするかを選択するための「**レイヤーパレット**」です。

16. ビットマップビュー

「**ビットマップビュー**」は、上記の「アウトラインビュー」と似ています。ツールとレイヤーのパレットがある点も同様です。

これを使い、上記で作成したアウトライン・グリフのビットマップ版を編集することができます。

背面レイヤーにあるのは、アウトラインを小さくレンダリングしたもので、前面レイヤーにあるのはビットマップ像です。【**ビットマップの再計算**】ボタンを押すことにより、どのビットを塗りつぶすかをプログラムに計算させることができます。左上の角にはビットマップグリフをフルサイズで表示した画像があります。

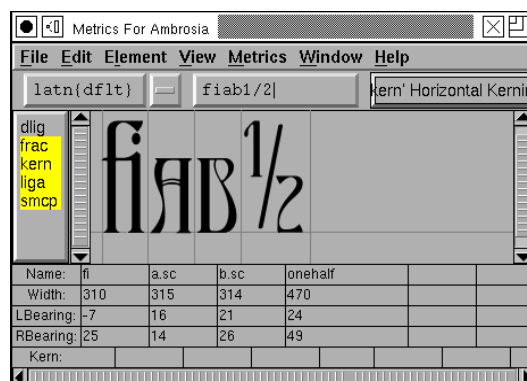


17. メトリックビュー

「**メトリックビュー**」を使えば、グリフを並べたときにどのように見えるかを試すことができます。

ウィンドウの上部をドラッグするか下の欄に適切な数値を入力する方法で、グリフの幅や左／右のサイドベアリングを変更することができます。

また、隣接する任意のふたつのグリフ間のカーニング情報を与えることもでき、さまざまな OpenType 機能がどのようにグリフの並び方に影響するかを確認できます。



ヘブル文字やアラビア文字（その他すべての右から左へ表記するアルファベット）では、グリフは右から始まり左に向けて配置されることに注意してください。また、CJK フォントでは、グリフを縦に並べた時の表記を見たいと思うはずです。

