［ディジタルタイプ―文字情報処理基盤の今とこれから―］

3 フォントエンジニア
―フォント技術を支える仕事―

齊藤暁男 | Monotype 株式会社

応般

フォントエンジニアとはどんな仕事か

フォントエンジニアの仕事には，フォントの制作そのものとそれに関連する技術的な知識が必要であると考えている．

私の本来の業務は，フォントラスタライザ（後述）などのフォント関連技術の組込みサポートだが，フォント生成時のデータ出力や評価用ツールを作成するなど，後述するフォントの仕様や使用環境などについてかかわる機会が多くありその経験を買われて，Monotype が初めて制作する日本語書体のフォントエンジニアとして携わることになった．

フォントの仕様としては 2016 年にバリアブルフォントが発表され，現在ではそれを利用できる環境が整いつつある状況ではあるが，実際に使用されているフォントデータの構造は大きく変わってはいないのが現状である．しかし，フォントデータを扱う環境は劇的に変化している．携帯電話がスマートフォンに置き換わり，腕時計型のウェアラブル端末が一般的になり，VR ゴーグルといったまったく新しいディスプレイの登場もある．また，継続して使われ続けている Windows や Mac といったデスクトップ環境もディスプレイの解像度という点では，非常に大きく変化している．詳細は後述するが，フォントデータにはいくつかのフォーマットが存在し，それぞれのフォーマットに対応したラスタライザがあることでディスプレイ上の文字の形を表現することができている．このように変化するデスクトップ，モバイル，ウェアラブル，組込みといったさまざまな環境でフォントデータを不自由なく使えるようにすることは，フォントエンジニアの重要な仕事の 1 つだ．

文字デザインがフォントになるまで

フォント制作には，フォントをデザインするタイプデザイナーおよびフォントエンジニアの共同作業が必要である．そのフォントデザインの方法は，デザイナーにもよるが，フォントのデザインを決める上で重要となる文字を手書きでスケッチし（もちろん，スケッチを使用しないでコンピュータ上でゼロからアウトラインを作成するケースも考えられる），それをスキャンしたものを下敷きにしてコンピュータ上でアウトラインを引いていく．ここで使用するグリッド空間（デザインユニットと呼ぶフォントデータ内でグローバルに使用されるグリッド数）のデザインが文字のアウトライン情報となる．フォントから文字を出力する際には，このアウトライン情報をスケーリングして描画したものを使用する．

アウトラインの制作環境は Monotype の場合，Glyphs というアプリケーションを使用している．図 -1 は Glyphs における文字の一覧画面のスクリーンショットである．フォントデータと同様に基本的には単一のファイルで文字のデザインを管理し，ここから出力したアウトラインデータの集合に，後に説明するキャラクタマッ

■図 -1　Glyphs アプリの字形一覧画面

プやデータを編集・付加してフォントデータとして仕上げる（フォントデータの要求仕様や使用環境次第ではフォント制作アプリケーションから出力したものをそのまま使用することも可能である）.

また，フォント作成時の以下の作業もフォントエンジニアの重要な仕事である.

- フォント制作のサポート，およびデータ生成のプロセスを担当する業務
- フォントデータの品質を検証する業務

フォント制作のサポート，およびデータ生成のプロセスを担当する業務

Monotype の日本語フォントは，日本オフィスに所属するデザイナーを主体にして制作されている．この日本語を含む CCJK フォント（簡体字中国語，繁体字中国語，日本語，韓国語）は一般的に収録文字数が非常に多く，収録には膨大な単純作業・反復作業を必要とする．これらデザイン作業の負荷を少しでも軽減できる仕組みや自動化ツールを作成したり，作業手順についてデザイナーと相談し効率化を図ったりする．たとえば，文字の構成をデータベース化して共通するエレメントをあらかじめ配置し，再利用できるような仕組み作りである.

フォントデータ生成プロセスではフォントの使用環境に合わせたデータの生成を行う．自動化されている作業も多いが，要求仕様次第でエンジニアが手を加えてデータを生成する場合もある.

フォントデータの品質を検証する業務

アウトラインの検証，データの整合性，使用環境での確認等幅広い作業を行う.

上記プロセスで生成されたデータについて OpenType の仕様を満たしているか，対象環境や顧客からの要求仕様を満たしているか，Monotype として販売・ライセンスするための品質は満たしているか，等の確認を行う．こちらも自動化されている作業は多いが，特殊な環境や一部のカスタムフォントの場合等は必要とされる検証業務を行う.

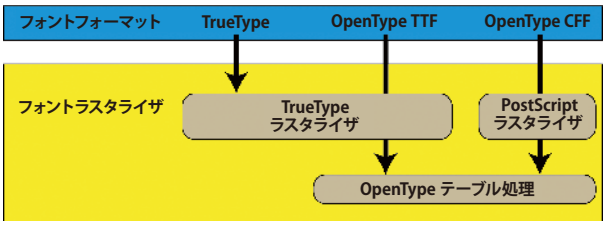
次章ではフォントを扱う上で把握しておくべき情報について説明する.

フォントデータのフォーマットの違いについて

フォントデータは大きく分けて TrueType（拡張子が TTF のもの），TrueType ベースの OpenType（以後 OpenType TTF，拡張子が TTF のもの），PostScript ベースの OpenType（以降 OpenType CFF，拡張子が OTF のもの）の 3 種類のフォーマットが主流である（ビットマップフォントや PCL フォント等，ほかにも多数のフォーマットが存在するが，ここでは省く）.

現在のほとんどの環境ではこれらのフォーマットの違いをユーザレベルでは意識せずに不自由なく使用できるが，これはフォントデータを扱う OS やアプリケーションで同じように扱えるように対応しているためである.

もう少し詳しく説明すると，TrueType フォント（TrueType および OpenType TTF），OpenType CFF はそれぞれ文字を描画するために別々の処理系統を持っている．TrueType フォントから文字を出力する場合には，TrueType 用の処理（ここではラスタライザと呼ぶ），OpenType CFF の場合には OpenType CFF 用のラスタライザが必要となる．もちろん，これらはアプリケーションプログラミングインタフェース（以降 API）レベルでは統合されていることが一般的であり，現代の環境であれば，前述のようにユーザレベルではフォーマットの違いを意識することなく両方のフォントデータを扱うことができる．図 -2 はフォントフォー



■図 -2　フォントフォーマットとラスタライザとの関係図

マットとフォントラスタライザとの関係の一例．両方扱えないケース事例としては，Windows 環境での旧来のグラフィックデバイスインタフェース（以降 GDI）や GDI+ の API を使用しているアプリケーション等は，OpenType CFF フォントを扱えない．

また，OpenType 機能を備えるフォント（OpenType TTF および OpenType CFF）は TrueType と比べて異体字の対応や文字詰め情報の収録といった点で機能的な優位性がある．ただし，これらの機能を使用するには，フォントを扱う側（OS，ラスタライザ，レイアウトエンジン，アプリケーション等）にて対応している必要がある．

文字コード，文字エンコーディングと文字セット

ここでは，フォントデータから文字の出力を得ようとする際にどのようにその文字を指定しているのかを説明する．

まず「文字セット」とは特定の文書を表現するために必要な文字の集合である．フォントデータは通常ある一定の文字セット（たとえば JIS X 0213:2004 や CP932 等）をカバーすることでその役割を十分に果たすことができる．

「文字コード」と「文字エンコーディング」という用語について，両者はしばしば混同して使用されている場面が見られるが，正確にはそれぞれ違うものを指している．

大まかに言うと，文字コードとは文字を示す数値，文字エンコーディングとは文字コードを文字と紐づける方式，という形で理解すればよい．昨今の環境は，Unicode エンコーディングを前提としたものがほとんどを占めるために上記のような現象を目にする機会は減ってきているが，たとえば 10 ～ 20 年前のデータを閲覧する際にテキストデータが文字化けして読むことができない場面もある（同様の現象は，やはりその当時に作成された Web ページなどでも発生する）．これはテキストファイル内で文字に指定されている文字コードが正しいエンコーディングで解釈されていないことによるものだ．なお，テキストエディタの中には文字エンコーディングを指定してファイルを開くことができるものもあり，その際に正しくない文字エンコーディングで開くと上記と同じ現象を確認することができる．

フォントデータにはさまざまな文字エンコーディングに対応する仕組みがある．フォントデータに収録されている文字には，その文字を表す名前（グリフネーム）が付けられているが，たとえば文字エンコーディング “A” で文字コード “10” の場合にどの文字にアクセスするのかといった対応表のような情報が cmap テーブルというフォントデータ内の領域に組み込まれている．これはフォントデータ制作時に人為的に付加されるものであり，通常は標準となる Unicode エンコーディングのみを含むことが主流となっているが，そのほかにフォントデータのターゲット地域の文字エンコーディングを実装するといった対応もある．

フォントデータに含まれる文字にはグリフネームが付与されているが，現在主流となっている OpenType CFF の場合，プロユーザにとっては事実上の業界標準である Adobe-Japan1 にて定義されている CID の割り当て（CID-Keyed）が必要となる．

CID の割り当てとは，Adobe-Japan1 にて定義されている文字のマッピング情報を基にして文字に番号（CID）を割り当てることを言う．

上で説明している文字コードと文字エンコーディングに当てはめると，Adobe-Japan1 が文字エンコーディング，CID が文字コードと解釈可能である．

この CID による入力は特に異体字の扱いにおいて有用である．Adobe-Japan1 のキャラクタマップにおいては CID と文字とが 1 対 1 で決まっている．一見当たり前のようではあるが日本語の漢字においては異体字があるために，Unicode では 1 つの Unicode 番地に対して複数の字形が存在している．

もちろん Unicode エンコーディングにおいても異体字の入力には対応しているが，CID 入力ほどに簡便では

なく，また使用環境もあまり浸透しているとは言えない．

この Adobe-Japan1 は文字セットとして機能しており，0 ～ 7 までのサブセットも用意されている（実用上は 9,354 文字をカバーする Adobe-Japan1-3 以上が現実的）．

一番大きい Adobe-Japan1-7 は新元号「令和」の合字文字（㍿）を含む 23,060 文字をカバーしている．

フォントに入れ込む文字以外の情報

フォントデータの内部は大きく分けると「テーブル」という単位で情報の種別ごとに分割されている．このテーブルには文字のアウトライン情報以外にもさまざまな種類のテーブルが定義されているが，特に重要な情報はフォントデータの名前や開発元，権利の所在を示す情報を含む name テーブルである．ほかにも不可欠な要素はたくさんあるが，現在の環境では name テーブルの情報を基にしてフォントを使用・指定するシステムが一般的であり，Monotype のようなフォントデータのライセンスをビジネスとしているフォントファウンダリーとしても，この name テーブルは最も重要である．図 -3 ～ 5 は Mac や Windows での name テーブルを示したものである．

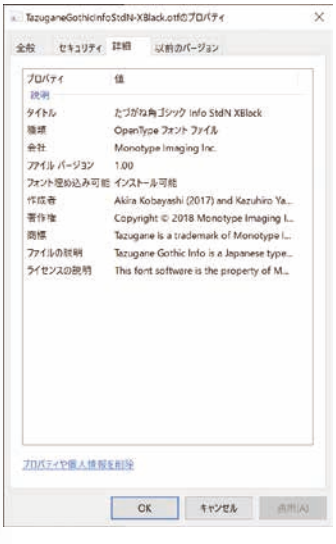
フォントデータは通常の環境であればシステムにインストールされているものを使用・参照することが基本となっている．そのため，この name テーブルが正しく設定されていないと，プラットフォーム間でフォントが埋め込まれていないオフィス文書等を移動した際に正しく表示されなかったり，以前作成した文書がフォントのアップグレードによってフォントが違って表示されたりする問題の原因となる．

これ以外にも，フォントデータ生成時の情報やメトリクス情報，後述する複雑な機能を実装するための OpenType テーブルの情報等が含まれる．

フォントにおける複雑な機能の実現

代表的なものは，OpenType 機能によるカーニング情報（kern，vkrn テーブル），およびプロポーショナル幅（palt，vpal テーブル）だ．カーニング情報は文字と文字の間隔を詰めたり広げたりして表示する際に参照

■図 -3　macOS の表示例



■図 -4　Windows の表示例

■図 -5　Windows のフォント設定表示例

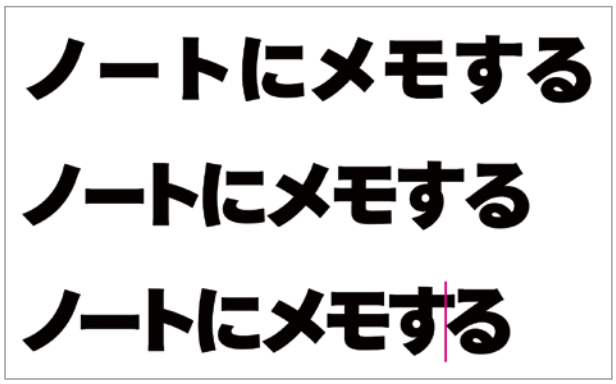
される情報となる.

たとえば図 -6 は, 上から順に「等幅」,「プロポーショナル幅」,「プロポーショナル幅+カーニング」の表示例である. 上段の“す”と“る”の組合せでは“す”の右下の空白部分がかなり広いように見える. これをプロポーショナルの字幅で表示することで中段の例はかなり自然な文字の並びが再現できている.

下段ではカーニング情報を適用し,“る”の左側を食い込ませて配置することで, より自然な文字の間隔を再現することができる.

この際の詰める (または広げる) 数値は, デザイナーによってデザイン時に設定される. もちろん, ツールやエンジニアリングの技術によって自動で設定することも可能だが, その場合においても最終的な確認・調整はデザイナーによって行われることが一般的である.

このカーニング情報は 1 文字対 1 文字, 1 グループ対 1 文字, 1 文字対 1 グループ, 1 グループ対 1 グループの組合せで右 (または縦組みの場合は下) に来る文字の位置をデザインユニット上でどれくらい水平移動(縦組みの場合は垂直移動) させるかという数値のことを言う. ここでの 1 グループとは, カーニングを設定する上で同じ特徴のアウトラインを持つ文字の集合を言う. (たとえばアルファベットの“c”と“e”は左側のカーブの形状に同じアウトラインを使用しているケースが比較的多く存在する. これらを“Y”の右側に配置する際には“c”であろうが“e”であろうが同じ詰め方でよいため,“c”と“e”とを同じグループとして扱えることになる).



■図 -6　プロポーショナル・カーニングの適用事例

この情報はすべての文字の組合せを網羅する必要はなく, デザイナーが必要と判断した組合せを実装することとなる.

カーニング情報は欧文フォントにも実装されているが, プロポーショナル幅に関しては日本語フォント特有の情報となる.

日本語フォントは文字の幅という大きなくくりでは, 等幅とプロポーショナルという 2 種類に分けることができる. TrueType ではこれらは通常別々のデータであるが, プロポーショナル幅情報を持つ OpenType フォントであれば 1 つのフォントデータで両方を備えることができる.

ただし, このような機能をユーザが使用する場合には, アプリケーション等で上記の設定を読み込む仕組みに対応している必要がある.

図 -7 および図 -8 では Adobe 社の InDesign 上で OpenType 機能やカーニング情報へのアクセス方法を示す.

こういった情報は, フォントデータに収録されている文字を描画しても文字を配置する際にはデザイン空間上での原点を基準として配置するために必要である (タイ語等のアクセントや声調を含む一部の TrueType はそれらの文字を原点からマイナス方向に配置して制作されているものもある).

これらの機能に対応しているアプリケーションでは, フォントデータに埋め込まれている情報を基にしてフォントデザイナーの意図した通りの文字詰めを再現することができる.

フォントエンジニアという仕事の将来

冒頭ではフォントの使用環境が大きく変わってきていると述べたが, フォントの制作環境も大きく変わってきているように感じる. 先進的なものではディープラーニングを使用したフォントの生成なども試みられている.

先にも述べているが, フォントエンジニアの仕事には

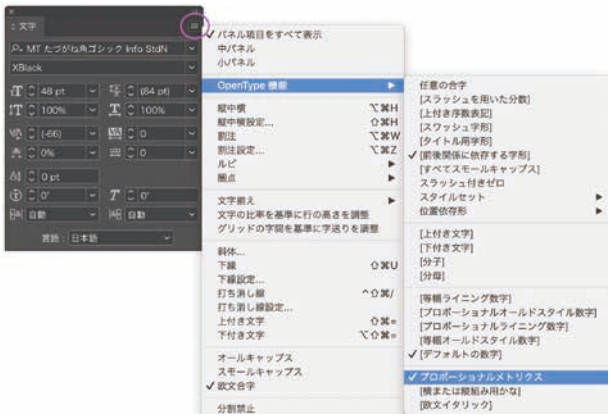
フォントの制作とともにフォントを使用される環境に対応させるように整えることも含まれている．

たとえば近年の最も大きなトピックであるバリアブルフォントは，任意のパラメータを設定するとフォントデータに埋め込まれている情報から任意のウエイトや字幅の文字を出力することができる．そのために，フォント制作時にバリアブルフォント用の情報を入れ込むとともに使用環境（OS，ラスタライザ，レイアウトエンジン，アプリケーション）においてはそれを正しく解釈して文字として出力する対応が必要となる．ごく近い将来にバリアブルフォントが当たり前に使用できる環境が広く普及してくると予想されるが，どのようなバリアブルフォントを制作するのか，または可能なのかといった点でフォントエンジニアの重要性が増してくると考えている（たとえば，現状の一般的な制作環境では適さないバリアブルフォントを制作する場面も考えられる．その際にいかにして制作するかという点はフォントエンジニアの仕事の領分である）．図 -9 はバリアブルフォントの概念図．

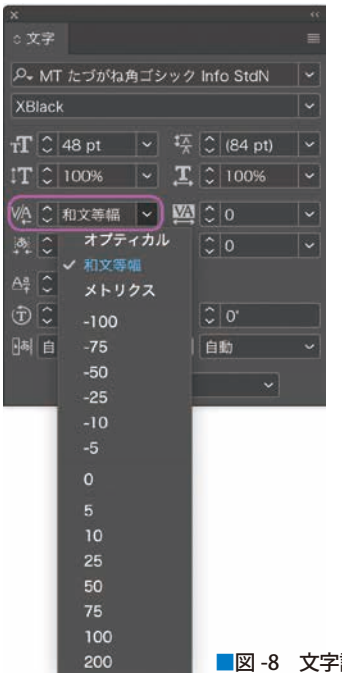
また，ここ数年でデスクトップ環境からモバイル環境への移行が急速に進んでいる．特に 2019 年 9 月にリリースされた iOS 13 では App Store 経由でのフォントのインストールに対応している．これによってモバイル環境で従来のデスクトップ環境に近いフォントの使用状況が生まれてくると想像される．

この状況とバリアブルフォントとが組み合わさることで，視認性や可読性といった観点からディスプレイにふさわしいパラメータを算出，もしくはあらかじめ設定しておくことでチューニングされた文字が表示されるような仕組み等が可能と考える．Monotype としては，130 年に及ぶフォント資産は変わらず提供するとともに，今後ますます変化するフォント使用環境にも積極的に対応していく．このバリアブルフォントを含む新しい変化は始まったばかりであるが，非常に期待できる未来であると確信している．

（2019 年 8 月 8 日受付）



■図 -7　OpenType 機能へのアクセス方法



■図 -8　文字詰め設定の変更方法

■図 -9　バリアブルフォントの概念図

■齊藤暁男　akio.saito@monotype.com

会津大学コンピュータ理工学部卒業．2012 年，Monotype 株式会社入社．同社初の日本語オリジナル書体，たづがね角ゴシックのエンジニアリングに携わる．