CAHIERS GUTenberg

FORMATONS LES FORMATS DE FONTE !

Luc DEVROYE

Cahiers GUTenberg, nº 46-47 (2006), p. 149-166.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_2006___46-47_149_0>

FORMATONS LES FORMATS DE FONTE !

Luc DEVROYE

RÉSUMÉ. — Les formats de fonte ont depuis toujours été un tir de barrage entre les artistes (graphistes et dessinateurs de fontes), les programmeurs (informaticiens), le monde des affaires et les utilisateurs. Ces quatre groupes ont influencé l'itinéraire historique que les formats de fonte ont suivi ces vingt dernières années.

Nous allons, dans cette présentation, rappeler les succès et les échecs des formats de fonte avant de présenter une liste de vœux pour les propriétés que nous attendons d'un bon format de fonte.

ABSTRACT. — Font formats are a tug of war between artists (designers and drawers), programmers (computer scientists), the business world, and users. Each of these four groups has had an influence on the path that font formats have followed. We review the successes and failures, and present a wish list of properties that a good font format should have.

NOTE. — L'auteur a reçu une bourse du NSERC (nº A3456) et du FCAR (nº 90-ER-0291) pour mener à bien ses recherches. Cet article est paru dans *TUGBOAT*, volume 24 (2003), nº 3 — *EuroTEX 2003 Proceedings* (Yannis Haralambous éd.), sous le titre « Formatting Font Formats », p. 588-596. Version française par Thierry Bouche. Il est reproduit ici avec l'autorisation de l'auteur, de Karl Berry (président de TUG) et de Barbara Beeton *(TUGBOAT editor)*.

1. INTRODUCTION

Essayons d'imaginer dans quel format les fontes existeront d'ici quelques décennies. Cette question est très pertinente en 2003, car le monde de la typographie est prêt pour une refonte de plus… Cet article commence par une présentation rapide de la situation actuelle, où les formats TrueType et PostScript sont dominants, tandis que le format OpenType, défini depuis plus de huit ans, est en cours de promotion.

Nous passons ensuite à une vision plus large et à plus long terme, ce qui nous permettra de discuter quelques aspects des formats de fonte informatisées.

Avant de nous plonger dans les aspects les plus techniques des formats de fonte électronique, il peut être utile de bien connaître les forces en présence qui participent à la définition de ces formats.

D'abord et avant tout, les *utilisateurs* aimeraient disposer de formats simples et utiles, faciles à manipuler et modifier. Ils veulent à la fois mettre les mains sur les œuvres d'art créées par de grands typographes et les raffinements techniques fournis par les experts ès fontes numériques. En outre, les utilisateurs professionnels réclament un certain degré de flexibilité des fontes, de façon à pouvoir y inclure des choix personnels.

Les *artistes et typographes* avaient une influence considérable sur les formats de fonte à l'ère pré-électronique. Les premiers typographes étaient à peu près tous des artisans. Au vingtième siècle, diverses avancées technologiques furent réalisées par des compagnies comme Linotype ou Monotype, à la demande des dessinateurs de caractères ; et nous avons assisté à une diminution croissante du délai entre la préparation des dessins sur le papier et la production industrielle des glyphes. À l'âge électronique, les artistes et typographes ont été écartés des décisions sur les formats de fonte, ce qui a conduit à une scission fâcheuse dans la grande famille des typographes : d'une part ceux qui ne se sont jamais adaptés à la souris et à l'écran, et qui ont continué à dessiner des caractères sur papier. Il est probable qu'ils ont été dissuadés par le média lui-même, ou par l'excès de mathématiques dans la nature des formats, voire dans les éditeurs. D'autre part, nous avons assisté à l'émergence d'artistes numériques qui dessinent des glyphes directement à l'écran, et cela avec une très grande efficacité. On peut placer dans cette catégorie des artistes prolifiques comme Lucas De Groot, Jean-François Porchez et David Berlow. Plus rarement, quelques uns ont même maîtrisé le format *bitmap*, et sont devenus des techniciens numériques aboutis. Verdana de Matthew Carter, une police vectorielle conçue et optimisée pour l'affichage sur écran, est un magnifique exemple de la production d'un maître artisan de l'ère numérique. Pour plus d'informations sur le point de vue d'un dessinateur, on pourra consulter le livre qu'Hermann Zapf a publié en 1991 [53]. Quoi qu'il en soit, pour ces deux catégories de créateurs, le format de fonte préexistait, et ils durent s'adapter à la

technologie. On peut se demander s'il ne serait pas judicieux de prendre leur avis à l'avenir, de façon à inventer un média leur permettant d'exprimer librement leur inventivité.

Les *ingénieurs* ont leur mot à dire dans cette affaire car ils peuvent rendre compte des limitations de certains médias. Les capacités des écrans, des imprimantes, et d'autres particularités physiques limitent le format dans lequel une fonte peut être présentée sur ces périphériques. Il y a une frontière mal définie entre les responsabilités de l'ordinateur ou du périphérique en matière de traitements : l'ordinateur peut envoyer les données brutes d'une police et attendre de la part du périphérique un traitement adéquat, mais certains périphériques peuvent nécessiter un format spécifique précalculé pour eux sur l'ordinateur.

Les *informaticiens* sont des acteurs chaque jour plus importants. Les créateurs de PostScript, Geschke et Warnock, qui l'ont développé à partir du langage de description de page (PDL) de John Gaffney en 1976 [1], ainsi que les formats de fonte dits Type 1 et Type 3 [2, 3], étaient des informaticiens doués d'une culture graphique. Le succès de PostScript est dû à sa simplicité et à sa flexibilité. L'influence d'Adobe est encore due aujourd'hui à l'invention de PostScript. Les informaticiens, même théoriciens, sont trop souvent préoccupés par des questions d'organisation des fichiers, et de standardisation, ce qui les tient à distance des utilisateurs. Ils devraient rester à l'écoute de ces derniers, et non le contraire.

En remontant la chaîne, nous trouvons les *vendeurs, fonderies et compagnies*, dont les intérêts sont en général commerciaux, et qui sont, par définition, concernés par l'image de la société, le volume des ventes, les parts de marché, les formats propriétaires et les stratégies logicielles. Les fontes sont souvent développées comme sous-ensemble d'une suite logicielle, ou comme éléments d'un système d'exploitation. Les enjeux couvrent aussi des brevets, marques déposées ou copyright : les diverses protections utilisées pour les logiciels comme pour les fontes. Le format de fonte effectivement soutenu par ce groupe est souvent le résultat de stratégies commerciales, l'exemple canonique étant fourni par l'histoire de PostScript, TrueType ou OpenType que nous retrouverons plus bas.

La dernière force à l'œuvre dans la création des formats de fonte est l'*inertie,* induite par les traditions et modèles historiques. Un format de fonte électronique est en général obtenu par modification d'un format préalable. La compatibilité ascendante est souvent considérée comme

une obligation pour un format nouveau, mais ceci est contredit par l'histoire réelle, qui a dû faire face à des sauts technologiques irréversibles.

Le répertoire des polices est riche, la plupart d'entre elles n'existant que dans l'un des nombreux formats ayant existé. De nombreuses polices historiques n'existent que sous forme imprimée (dans des spécimens, ou simplement dans un tirage), tandis que des centaines, voire des milliers, sont des types de métal ou de bois. À l'âge de la photocomposition (les années 1950-1980), les polices étaient stockées sous une forme photographique et, pour finir, plusieurs formats électroniques ont été introduits depuis les années 1980. Cela a commencé par des *bitmaps* (« fon », « bdf », « fnt »…). Dès 1982, Jim Warnock et Charles Geschke introduisirent le PostScript [1] et l'idée de stocker les glyphes sous forme de contours analytiques (courbes de Bézier), ce qui a conduit aux formats de fonte dits Type 3 et Type 1. Knuth s'est aussi appuyé sur les courbes de Bézier pour tracer des contours, mais il a eu l'idée de décrire les glyphes par des programmes dans son langage Metafont [30], qu'il a développé au cours des années 1977-1985. Sampo Kaasila a inventé le format TrueType chez Apple pendant les années 1987-1989, suite à une décision stratégique de contrer l'hégémonie d'Adobe sur la chaîne graphique due à PostScript. Finalement, Microsoft et Adobe se sont unis pour définir OpenType dans l'espoir de réconcilier TrueType et PostScript Nous verrons ci-dessous qu'il s'agit d'une évolution mineure du point de vue technologique. Quand on s'interroge sur le futur, il ne faut pas perdre de vue cette histoire assez contrastée. L'ère de l'électronique est la seule où les formats de fonte soient devenus propriétaires — ils sont conçus et contrôlés par une entité commerciale. C'est un écueil à éviter lorsque nous concevrons les formats à venir.

On peut raisonnablement supposer que le modèle des données informatisées actuel (composé d'atomes binaires, stockés sous une forme physique) va exister encore pendant quelques décennies. Nous utiliserons donc des mots comme fichiers et octets dans la suite, même si un hypothétique lecteur de l'avenir nous trouvera vieux-jeu. C'est pourtant à lui que nous nous adressons désormais, en développant le « modèle réparti ».

2. Le modèle réparti

Une fonte est l'implémentation d'une police : elle contient idéalement la description intégrale de cette police [1]. C'est comme un livre entier — lisible par tous, aucun détail ne fait défaut, l'auteur est précisément identifié, etc. De même, une fonte devrait être implémentée comme un « livre ouvert » déchiffrable par un humain. Aucun des formats antérieurs ne permettait cela : à l'époque du plomb, de nombreuses informations sur le dessin manquaient, et seules les fonderies possédaient les poinçons. Les formats TrueType, Type 1 et OpenType ne sont lisibles que par des programmes. Metafont et Type 3 ne peuvent être déchiffrés que par des informaticiens. De fait, à cause de la concurrence des formats, nous avons souvent la même police dans plusieurs formats, avec de nombreuses différences de détail dues aux incompatibilités entre les formats. On peut dire qu'aujourd'hui une police est incarnée dans une foule de déclinaisons, c'est le chaos !

Il devrait exister une fois pour toutes une fonte-mère lisible par tous, qui serait immuable, comme pour les versions d'un logiciel. Une modification engendrerait une nouvelle fonte. On pourrait dériver des données de la fonte-mère les implémentations dans les formats populaires à l'aide de filtres, et seulement dans ce sens-là.

Les éditeurs de fontes du moment sont nombreux : Fontographer (possédé par Macromedia à l'époque où cet article fut rédigé, aujourd'hui par FontLab, décrit par Moye [39]), FontLab (de Yuri Yarmola), Font Studio (de Letraset), Ikarus (de Peter Karow pour URW), FontForge (de George Williams), FontCreator (d'Erwin Denissen), Softy (de Dave Emmett), Manutius (d'Andreas Gebert) et Noah (de Yeah Noah). Chacun gère quelques formats sur quelques plateformes. Les éditeurs du futur devraient être conçus pour créer et manipuler la fonte-mère, ce qui serait plus logique. Les artistes devraient pouvoir définir l'aspect de la police directement dans cette fonte-mère. Les périphériques se contenteraient de formats électroniques simplifiés déduits de cette fonte-mère. On doit réaliser que tous les éditeurs sérieux ont un format interne, lisible par un humain, pour représenter les fontes, qui pourraient servir de modèle pour la fonte-mère. Mais la plupart ne vont pas au-delà d'une

1. Nous distinguons ici *fonte* en tant que réalisation matérielle d'une *police* : le dessin d'un caractère. [N.d.T.]

traduction exacte du format binaire correspondant. Pour un survol des technologies typographiques, nous nous référons aux livres d'André [6], Karow [26, 27] et Knuth [33], et aux articles de Gonczarowski [17, 18] et André & Hersch [7].

Chacune des sections ci-dessous traite de façon plus approfondie de l'un des aspects de la fonte-mère.

3. Contour et précontour

L'une des contributions les plus importantes à la géométrie calculatoire et au dessin géométrique assisté par ordinateur est le développement des courbes de Bézier par James Ferguson, concepteur d'aéronefs, Pierre Bézier, ingénieur chez Renault, et Paul de Casteljau, ingénieur chez Citroën. On peut décrire ou approcher des objets à deux ou trois dimensions de façon simple à partir de courbes. C'est en fait une méthode pour représenter un objet physique à l'aide de *bits*, donc une méthode de compression de données : le tracé d'une courbe peut être numérisé et produire un *bitmap* de grande taille, tandis que sa représentation analytique n'occupera qu'une toute petite place en mémoire, pour une précision comparable, car la formule qui fournit le tracé d'une courbe de Bézier d'ordre n ne repose que sur la donnée de $n+1$ points de contrôle $x_0, x_1, \ldots, x_n$ dans le plan :

$$x(t) = \sum_{i=0}^{n} \mathrm{C}_n^i t^i (1-t)^{n-i} \cdot x_i, \quad 0 \leqslant t \leqslant 1.$$

Ici, $x(t)$ est une courbe paramétrée, une moyenne pondérée à coefficients continus des points de contrôle (donc la courbe de Bézier reste à l'intérieur de l'enveloppe convexe de ces points), qui va du point x_0 à x_n. Les propriétés mathématiques des courbes de Bézier et des splines en général sont décrites par Farin [15] et Su & Liu [45].

Il est assez naturel que PostScript et Metafont aient adopté les courbes de Bézier : leurs créateurs se sont fondés sur les cubiques ($n = 3$). TrueType repose sur des courbes quadratiques ($n = 2$), ce qui n'est pas malin, vu que l'on peut évidemment représenter une quadratique par une cubique (étant donnés x_0, x_1, x_2, poser $y_0 = x_0$, $y_1 = (2x_1 + x_0)/3$, $y_2 = (2x_1 + x_2)/3$, $y_3 = x_2$, pour obtenir les points de contrôle cubiques y_i), mais pas l'inverse. Donc Type 1 peut se déduire de TrueType, mais les

approximations cubiques sont en général préférées aux quadratiques, en particulier par les artistes.

Une courbe de Bézier ne peut pas représenter un cercle parfait, quel que soit l'ordre n (c'est un exercice sans difficulté pour un lecteur féru de mathématiques). Par exemple, la meilleure approximation cubique d'un quart de cercle est obtenue à l'aide des points de contrôle $(0,1)$, $(a,1)$, $(1,a)$, $(1,0)$, où $a = (4/3)(\sqrt{2}-1) = 0{,}552\,284\,749\ldots$ Ce défaut aurait pu être circonvenu si Bézier avait autorisé des fonctions trigonométriques ou racines dans ses développements paramétriques.

Les dessinateurs de caractères qui travaillent sur écran sont en fait des artistes en placement de points de Bézier. Leur instrument est la souris. C'est assez peu intuitif, car beaucoup de points de contrôle ne sont pas sur les courbes qu'ils déterminent, et la continuité des dérivées entre segments de Bézier adjacents ne saute pas à l'œil nu. Certains dessinateurs continuent de concevoir sur papier, et numérisent leurs dessins pour poursuivre le travail sur un ordinateur. D'autres encore, habitués à des logiciels dédiés, sont passés maîtres dans l'art de placer des points liés indirectement aux points de Bézier, comme peuvent l'être les splines de Böhm [10], où des propriétés comme la continuité des dérivées sont assurées.

Hobby [23] et Knuth [30] ont développé un algorithme pour construire une suite de courbes de Bézier forcées de passer par certains points fixés par l'artiste. Cet algorithme fait partie des routines standard de Metafont [30, p. 131], il peut être un outil de production puissant. Nous appelons de telles façons de décrire les contours des « précontours ».

En résumé, la fonte-mère devrait être ouverte et ne pas imposer de choix parmi les innombrables formats de contour ou précontour, tant que chaque format définit une courbe mathématique de façon unique. Des concaténations de courbes de Bézier de degré arbitraire devraient être autorisées, mais aussi une palette de formats de précontour pour satisfaire le plus grand nombre de typographes. Au minimum, l'un des modèles de spline supportés devrait permettre de définir des arcs de cercles, de telle sorte que nous puissions enfin disposer d'une représentation exacte de ces dessins fantastiques réalisés à la règle et au compas par des maîtres comme Philippe Grandjean, dessinateur du Romain du roi (1693-1745).

4. Modèles d'encrage

Les deux formats principaux, TrueType et Type 1, et leur cousin OpenType, sont tous basés sur un même modèle d'encrage primitif, qui repose sur l'idée qu'un glyphe est défini par un certain nombre de contours fermés, qui sont remplis d'encre d'après la règle du nombre de tours non nul. Ce qui signifie que, si un point ou un pixel se trouve dans une région, sa couleur (blanc ou noir) sera déterminée en traçant une demi-droite allant vers l'infini (dans n'importe quelle direction !) et en calculant la somme algébrique des contours croisés. La valeur 1 est affectée à un contour qui tourne dans le sens des aiguilles d'une montre, et -1 dans le cas contraire. Mais ça n'est pas exactement la façon dont nous nous y prenons pour peindre à la maison, où les actions élémentaires seraient plutôt : surimprimer et effacer. On devrait aussi pouvoir travailler avec plusieurs images monochromes, peut-être des calques, et définir l'image définitive comme le résultat d'opérations logiques à partir de ses composantes, comme « ou » (exclusif ou non), « et », la négation, etc. On devrait aussi pouvoir marquer une région comme noire ou blanche en la déclarant telle, sans se préoccuper d'arithmétique.

Les fontes composées à partir de tracés se distinguent des contours par leur modèle d'encrage : un tracé est défini par une collection de splines de Bézier, et l'encre est appliquée en suivant le tracé avec une brosse ou un pinceau ou tout autre outil. Les idéogrammes semblent être le territoire naturel de telles fontes, mais elles sont également le format naturel dans lequel capturer un tracé sur une tablette.

La recommandation d'accepter des modèles d'encrage nombreux semble être une requête d'extension du modèle graphique dominant, en particulier de PostScript, mais ça n'est pas nécessaire car on pourra toujours exporter dans ce format à partir de la fonte-mère. L'extension suggérée a pour seul but de rendre le dessin de caractères plus facile, plus universel, plus abordable.

5. Complexité des contours

Les possibilités en terme de points ou de contours ne sont pas illimitées dans les formats électroniques actuels. Par exemple, un chemin ne peut excéder 750 points de contrôle en PostScript, donc en Type 1. De telles limitations empêchent de conserver certains caractères complexes comme des fleurons, capitales décoratives ou simplement obtenus par

traçage de scans de haute définition. Les limites de TrueType sont plus élevées, mais la fonte-mère ne devrait en principe imposer aucune limite. On peut accepter des limites dues aux formats d'exploitation déduits de la fonte-mère, mais pas à la source.

6. Précision

Toutes les formes de contour nécessitent une codification mathématique. Comme les points doivent avoir une représentation unique indépendamment des plateformes, il est impératif que les descriptions mathématiques soient toutes en termes d'entiers. Par exemple, un point peut avoir des coordonnées entières, mais aussi rationnelles. À l'heure actuelle, les coordonnées typiques d'une fonte Type 1 sont des millièmes de cadratin. En TrueType la définition de 1000×1000 est remplacée par 2048×2048. La différence des systèmes de coordonnées montre qu'il n'y a pas de conversion possible sans perte de précision.

Il est incompréhensible que personne n'ait seulement envisagé de repousser ces limites de précision. Imaginons un caractère complexe composé de cinquante lignes et colonnes de cercles adjacents. Si les coordonnées sont entières sur une grille de 1000×1000, le rayon de chaque cercle serait obligatoirement de 5. Une représentation cubique de Bézier devrait alors utiliser des points de contrôle en $(0,5)$, $(a,5)$, $(5,a)$, $(5,0)$, aucune valeur potable pour a n'étant disponible sur la grille (la valeur optimale étant de l'ordre de 2,75 comme vu précédemment).

Signalons également une autre motivation pour augmenter la précision : l'archivage du passé. Plus la précision est grande au moment de la conversion vers un format numérique, moins ce format altérera l'original.

Des méthodes adaptatives devraient permettre de stocker tous les éléments d'une fonte dans une précision suffisante pour chaque élément, de façon à conserver des tailles de fichier raisonnables sans limiter la précision.

On peut d'ailleurs mentionner que c'est en théorie une possibilité du format PostScript, donc Type 3, et même Type 1 puisque la FontMatrix peut être modifiée à tout moment, mais, dans la pratique, la plupart des programmes attendent une grille de 1000×1000 unités.

7. Fontes et programmation

À l'heure actuelle, les formats de fonte électronique ne sont guère que des tables. Tout comme leurs contreparties de métal, ce sont des objets inertes qui attendent d'être manipulés par un compositeur humain ou mécanique. Même si certaines firmes prétendent que leurs fontes sont des programmes, c'est faux à l'exception des formats Metafont et Type 3, qui furent tous deux des avancées importantes dans le domaine de la typographie numérique. Quelques fontes TrueType ont un peu de code dans les instructions d'adaptation aux basses définition, mais cela peut aussi être considéré comme des tables plutôt que comme des programmes.

Le format Type 3 autorise toutes les constructions du langage PostScript : il y a des paramètres, des variables, des boucles et des instructions conditionnelles. Il est possible d'avoir des fontes aléatoires, par exemple pour simuler l'écriture mansuscrite, et de produire des glyphes dépendant du contexte, par exemple pour adapter les liaisons de lettres adjacentes. On peut programmer les glyphes à partir d'un ensemble de paramètres ajustables. Les fontes les plus simples qui permettent de tels ajustements sont probablement les fontes multimaster qu'Adobe a lancées dans les années 1990, où les paramètres permettent d'interpoler entre des dessins extrêmaux. Cela peut bien sûr être programmé en Type 3. Metafont permet aussi ce genre de choses et Knuth l'a amplement démontré avec la famille Computer Modern [32] où un programme par glyphe permet de définir 72 fontes différentes, dont les styles machine à écrire, bâton et mécane de transition (voir aussi [19]). D'autres tentatives de paramétrisation, comme Infinifont (McQueen & Beausoleil [38]) et LiveType (Shamir & Rappoport [42, 43]), n'ont pas vécu longtemps.

L'inconvénient de fontes véritablement programmables est la nécessité de disposer dans les applications et périphériques d'interpréteurs puissants ou de convertisseurs vers des formats plus simples — sans parler des risques de virus : exécuter une « fonte » Type 3 peut avoir pour conséquence la destruction de quelques fichiers… Enfin, les interpréteurs associés à des langages puissants ont tendance à être protégés par des licences ou des brevets, et leur utilisation impose souvent des dépenses extraordinaires. En s'en tenant sagement à des fonctions uniquement mathématiques et graphiques, on devrait pouvoir limiter

ces inconvénients, et marquer les fontes qui contiennent un code actif comme c'est le cas pour les multimaster.

Les bénéfices de fontes programmables sont énormes dans le cas de la typographie mathématique. Le modèle de Knuth (Metafont + TEX [31]) a vingt ans et a de nombreuses limitations qui demandent une mise à jour. Il devrait y avoir un continuum de grands symboles, comme les parenthèses ou les intégrales, adaptés à toutes les situtations, notamment en terme de graisse et d'étendue. Aujourd'hui, nous devons nous satisfaire d'un nombre fini de possibilités, ce qui aboutit bien souvent à des aberrations esthétiques.

Évidemment, les symboles ajustables sont seulement une petite partie de mes vœux pour une typographie mathématique perfectionnée. Il devrait y avoir une symbiose entre figures, formules et texte, produisant des pages aussi équilibrées que ce que peut faire un mathématicien habile au tableau. Ces questions nécessiteront non seulement une extension des formats de fonte, mais également un nouveau paradigme pour toute la composition !

Pour ce qui est de l'utilisation de l'aléatoire pour la simulation de l'écriture manuscrite, nous nous référons à Devroye & McDougall [13] du point de vue théorique, Desruisseaux [12] pour une fonte savamment étudiée appelée MetamorFont, et à André & Borghi [5], Dooijes [14] ou encore van Blokland & van Rossum [51] pour des tentatives plus anciennes dans cette direction. Tous ces travaux reposent sur la capacité de programmer vraiment en Type 3 pour obtenir des caractères partiellement aléatoires qui sont tracés, soit à partir d'échantillons d'une écriture réelle comme le premier cité, soit en programmant les aléas dans les contours. Ce serait honteux d'interdire un générateur de nombres aléatoires dans la spécification de la fonte-mère. Il faut évidemment prévoir pour des tests de pouvoir reproduire les nombres aléatoires générés.

8. Ligatures et formes contextuelles

Les ligatures sont des combinaisons de deux caractères ou plus. Les caractères contextuels sont des caractères uniques dont la forme dépend de la place où ils se trouvent. L'activation d'un changement de contexte devrait toujours être de la responsabilité de l'application : la fonte elle-même devrait uniquement fournir les variantes contextuelles sans chercher à interpréter le contexte.

Cette séparation entre forme et application devrait être aussi respectée dans le cas des ligatures. Les fontes ne sont qu'un réservoir de formes. Cette distinction est rigoureusement à l'œuvre dans la combinaison Metafont + TEX. OpenType a introduit une table dite GSUB qui gère les ligatures en collaboration avec un logiciel comme InDesign. La séparation est moins claire et les dessinateurs de caractères sont désormais supposés inclure les règles sous une forme absconse dans leurs polices.

L'arabe nécessite un grand nombre de ligatures pour être composé de façon satisfaisante (voir p. ex. Smitshuijzen AbiFarès [44]). Mais c'est aussi le cas de l'écriture latine manuscrite. L'auteur a fait des expériences dans ce sens : inclure à l'aide d'une tablette dans une fonte Type 3 de l'ordre de 1600 ligatures pour les paires de lettres les plus fréquentes, avec des variantes initiales, médianes ou finales, mais aussi des combinaisons de capitales suivies d'une ou deux lettres minuscules. Sur un texte, un petit programme décidait de la meilleure forme d'un mot à l'aide d'heuristiques. Quiconque pourra améliorer les résultats obtenus en retouchant le programme, sans devoir toucher à la fonte.

9. Images et *bitmaps*

L'archivage des polices anciennes, s'il est fait à l'aide d'un format de type contour, nécessite une précision accrue. Quoiqu'il en soit, la conversion fondamentale de *bitmaps* vers des contours intervient à un moment donné. Cette étape s'appelle traçage, ou autotraçage. Les algorithmes dédiés sont légion (voir Avrahami & Pratt [9], Plass & Stone [40], Itoh & Ohno [24], Gonczarowski [16], Schneider [41], Lejun, Hao & Wah [34], ou Mazzucato [37]). Les perfectionnistes voudront cependant conserver aussi l'image originale en sus des contours dérivés, de plus ou moins bonne qualité. Il est possible que la quantité de données à stocker soit énorme, mais il existe des méthodes efficaces de compression sans perte de détail. La fonte-mère devrait donc autoriser la conservation de *bitmaps* extrêmement précis.

Il n'est pas totalement absurde d'imaginer qu'un jour ou l'autre toutes les fontes seront au format *bitmap*, étant donnée l'évolution conjointe des capacités de stockage et des méthodes de compression. L'avantage d'un tel format est que le dessin est beaucoup plus facile et abordable pour un typographe. D'une certaine façon, le papier et l'électronique pourraient converger à nouveau.

10. Codes et standards

L'effort de standardisation du nommage des symboles et de l'affectation d'un nombre permanent à chaque symbole (un codage) devrait se poursuivre. Unicode a modifié le paysage, mais il n'est pas envisageable d'attendre que chaque fonte soit « complète », quelle que soit la définition de « complète » que l'on préfère. Des nouveaux symboles apparaissent chaque jour : les organisations de standard ne peuvent qu'être à la traîne. En outre, l'existence de glyphes spécifiques est une valeur ajoutée pour certaines polices, d'autant plus grande que lesdits glyphes sont rares. C'est la nature humaine que d'inventer toujours, par conséquent la fonte-mère ne doit pas être intimement liée à un système de codage particulier. Une fonte pourrait être marquée comme fournissant les caractères d'Unicode, ou d'un sous-ensemble, mais on ne saurait prédire l'avenir en matière de noms de glyphes ou de codage — qui se serait risqué à prévoir l'euro pendant les années 1970 ?

Le nombre de glyphes dans une fonte ne devrait pas non plus être limité a priori. Chaque glyphe devrait se voir affecter un nom et une position numérique, dont la borne doit rester arbitraire.

11. Métadonnées

Une composante essentielle de la fonte-mère est l'ensemble des informations relatives à l'histoire ou au contexte de la police comme de la fonte qui l'implémente. Chaque fonte a un généalogie, un contexte. On peut imaginer que chaque fonte disposera d'un arbre généalogique (sous la forme d'un graphe acyclique dont chaque nœud contiendrait les informations datées relatives à ce proche « parent »). Il ne faut pas compter sur un archivage tiers de ces informations, il n'y a donc pas de meilleure manière de la conserver que d'en faire une composante en soi de la fonte-mère. De la sorte, les crédits de l'ensemble des créateurs associés au développement d'une fonte survivront aussi longtemps qu'elle, ce qui est loin d'être la situation actuelle.

Les noms de fontes devraient rester uniques, peut-être par l'ajout d'identifiants de fonderie et de version.

12. Format lisible

La fonte-mère ne saurait exister sous forme de programme cryptique à l'intention des seuls programmeurs ou logiciels. Pour ce qui est de

TrueType et OpenType, van Blokland & van Rossum [52] ont développé l'outil TTX qui traduit sans perte le contenu du format binaire en XML lisible. Il existe d'autres exemples similaires pour d'autres formats. Les formats non commerciaux comme Metafont sont généralement lisibles dès leur conception.

Quiconque devrait pouvoir lire et interpréter le code d'une fonte, sans avoir besoin de logiciels spécifiques, dont la durée de vie n'est jamais bien longue. Ajouter un accent sur une lettre devrait être une opération banale. Il est important qu'il soit facile de modifier le code d'une fonte-mère pour réaliser des effets non prévus à l'origine. Pour toutes ces raisons, le format ne doit pas être binaire. Si l'on objecte, pour des raisons de dimension des fichiers, il faut se souvenir qu'il est toujours possible de préparer des versions comprimées ou simplifiées pour les transferts ou l'exploitation par un périphérique.

13. Opérations automatisées

Les données métriques (chasses, crénage…) sont essentielles pour toutes les polices. Kindersley [28] propose des idées intéressantes pour espacer les lettres. Chez URW [47, 48, 50], des programmes ont été développés pour automatiser intégralement l'espacement des caractères. Espacer proprement une police est un travail d'expert que bien peu de typographes maîtrisent. Par suite, la fonte-mère devrait permettre de signaler que l'espacement est produit par un algorithme, voire de choisir cet algorithme, qui pourrait avoir des options également indiquées là. De la sorte, un crénage manuel pourrait cohabiter avec un crénage automatisé, l'un prenant le pas sur l'autre le cas échéant.

Le problème des *hints*, ou suggestions d'adaptation des contours en *bitmaps* en fonction de la définition du périphérique, ne concerne que les fontes électroniques. Là encore, il doit être possible de déclarer la méthode préférée (automatique ou manuelle) et de la préciser, voire de la paramétrer. Des exemples d'algorithmes sont décrits par Karow [25], Andler [4], Hersch & Bitrisey [21] et Herz & Hersch [22]. On pourrait argumenter que ces questions d'adaptation au périphérique n'ont que faire dans la fonte, puisqu'elles sont de la responsabilité du périphérique de sortie.

14. Addendum

Après la rédaction de cet article en 2003, nous avons appris l'existence de tentatives similaires de définition d'un format textuel pour les fontes-mères. Signalons le format UFO (*unified font object*) introduit en octobre 2004 à la conférence ATypI de Prague par l'équipe de Letterror, Erik van Blokland et Just van Rossum. Par ailleurs, le format interne SFD (*spline font database font format*) défini par George Williams pour son éditeur de fontes libre FontForge rentre également dans cette catégorie.

Bibliographie

[1] Adobe Systems. — *PostScript language reference manual*, Addison-Wesley, Reading, MA, 1990.

[2] Adobe Systems. — *Adobe Type 1 font format*, Addison-Wesley, Reading, MA, 1990.

[3] Adobe Systems. — *Adobe font metric files specification, version 3.0*, Adobe, 1990.

[4] S. F. Andler. — « Automatic generation of gridfitting hints for rasterization of outline fonts », *Proceedings of the international conference on electronic publishing, document manipulation & typography, Gaithersburg, Maryland, September 1990* (edited by R. Furuta), p. 221–234, New York, 1990.

[5] J. André & B. Borghi. — « Dynamic fonts », *Raster imaging and digital typography* (edited by J. André and R. D. Hersch), p. 198–204, Cambridge university press, Cambridge, 1989.

[6] J. André. — « Création de fontes et typographie numérique », IRISA, campus de Beaulieu, Rennes, 1993.

[7] J. André. — « An introduction to digital type », *Visual and technical aspects of types* (edited by R. D. Hersch), p. 56–63, Cambridge university press, Cambridge, UK, 1993.

[8] J. André. — « Ligatures & informatique », *Cahiers GUTenberg*, nº 22, p. 61–86, 1995.

[9] G. Avrahami & V. Pratt. — « Sub-pixel edge detection in character digitization », *Raster imaging and digital typography II* (edited by R. A. Morris and J. André), p. 54–64, Cambridge university press, Cambridge, 1991.

[10] W. Böhm. — « Cubic B-Spline curves and surfaces in computer-aided geometric design », *Computing*, vol. 19, p. 29–34, 1977.

[11] W. Böhm, G. Farin & J. Kahmann. — « A survey of curve and surface methods in CAGD », *Computer-aided geometric design*, vol. 1, p. 1–60, 1984.

[12] B. Desruisseaux. — « Random dynamic fonts », M.Sc. thesis, School of computer science, McGill university, Montreal, Canada, October 1996.

[13] L. Devroye & M. McDougall. — « Random fonts for the simulation of handwriting », *Electronic publishing (EP-odd)*, vol. 8, p. 281–294, 1995.

[14] E. H. Dooijes. — « Rendition of quasi-calligraphic script defined by pen trajectory », *Raster imaging and digital typography, Raster imaging and digital typography : proceedings of the international conferences, École polytechnique fédérale, Lausanne, Switzerland, October 1989* (edited by J. André and R. D. Hersch), p. 251–260, Cambridge university press, Cambridge, 1989.

[15] G. Farin. — *Curves and Surfaces for CAGD, a practical guide*, Academic press, New York, 1993.

[16] J. Gonczarowski. — « A fast approach to auto-tracing (with parametric cubics) », *Raster imaging and digital typography* (edited by R. A. Morris and J. André), vol. 2, p. 1–15, Cambridge university press, Cambridge, 1991.

[17] J. Gonczarowski. — « Industry standard outline font formats », *Visual and technical aspects of types* (edited by R. D. Hersch), p. 110–125, Cambridge university press, Cambridge, UK, 1993.

[18] J. Gonczarowski. — « Curve techniques by autotracing », *Visual and technical aspects of types* (edited by R. D. Hersch), p. 126–147, Cambridge university press, Cambridge, UK, 1993.

[19] Y. Haralambous. — « Parametrization of PostScript fonts through Metafont—an alternative to Adobe multiple master fonts », *Electronic publishing (EP-odd)*, vol. 6, p. 145–157, 1993.

[20] Y. Haralambous. — « Tour du monde des ligatures », *Cahiers GUTenberg*, nº 22, p. 87–100, 1995.

[21] R. D. Hersch & C. Bitrisey. — « Model-based matching and hinting of fonts », *ACM computer graphics*, vol. 25, p. 71–80, 1991.

[22] J. Herz & R. D. Hersch. — « Towards a universal auto-hinting system for typographic shapes », *Electronic publishing (EP-odd)*, vol. 7, p. 251–260, Special issue on typography, John Wiley, 1994.

[23] J. D. Hobby. — « Smooth, easy to compute interpolating splines », *Discrete computational geometry*, vol. 1, p. 123–140, 1986.

[24] K. Itoh & Y. Ohno. — « A curve fitting algorithm for character fonts », *Electronic publishing (EP-odd)*, vol. 6, p. 195–205, 1993.

[25] P. Karow. — « Automatic hinting for intelligent font scaling », *Raster imaging and digital typography : proceedings of the international conferences, École polytechnique fédérale, Lausanne, Switzerland, October 1989* (edited by J. André and R. D. Hersch), p. 232–241, New York, 1989.

[26] P. Karow. — *Digital typefaces*, Springer-Verlag, Berlin, 1994.

[27] P. Karow. — *Font technology*, Springer-Verlag, Berlin, 1994.

[28] D. Kindersley. — *Optical letter spacing for new printing systems*, Wynkyn de Worde Society, distributed by Lund Humphries publishers Ltd., 26 Litchfield St. London WC2, 1976.

[29] D. Kindersley & N. Wiseman. — « Computer-aided letter design », *Printing world*, p. 12–17, 1979.

[30] D. E. Knuth. — *The Metafont book*, Addison-Wesley, Reading, Mass., 1986.

[31] D. E. Knuth. — *The TEXbook*, Addison-Wesley, Reading, Mass., 1986.

[32] D. E. Knuth. — *Computer Modern typefaces*, Addison-Wesley, Reading, Mass., 1986.

[33] D. E. Knuth. — *Digital typography*, Cambridge university press, 1999.

[34] S. Lejun, Z. Hao & C. K. Wah. — « FontScript—A Chinese font generation system », *Proceedings of the international conference on Chinese computing (ICC94)*, p. 1–9, 1994.

[35] C. W. Liao & J. S. Huang. — « Font generation by beta-spline curve », *Computers and graphics*, vol. 15, p. 527–534, 1991.

[36] J. R. Manning. — « Continuity conditions for spline curves », *The computer journal*, vol. 17, p. 181–186, 1974.

[37] S. Mazzucato. — « Optimization of Bézier outlines and automatic font generation », M.Sc. thesis, school of computer science, McGill university, Montreal, Canada, 1994.

[38] C. D. McQueen III & R. G. Beausoleil. — « Infinifont : a parametric font generation system », *Electronic publishing (EP-odd)*, vol. 6, p. 117–132, 1993.

[39] S. Moye. — *Fontographer : type by design*, MIS press, 1995.

[40] M. Plass & M. Stone. — « Curve-fitting with piecewise parametric cubics », *Computer graphics*, vol. 17, p. 229–239, 1983.

[41] P. J. Schneider. — « An algorithm for automatically fitting digitized curves », *Graphics gems* (edited by A. S. Glassner), p. 612–626, Academic press, San Diego, CA, 1990.

[42] A. Shamir & A. Rappoport. — « Extraction of typographic elements from outline representations of fonts », *Computer graphics forum*, vol. 15(3), p. 259–268, 1996.

[43] A. Shamir & A. Rappoport. — « LiveType : a parametric font model based on features and constraints », technical report TR-97-11, institute of computer science, The Hebrew university, 1997.

[44] H. Smitshuijzen AbiFarès. — *Arabic typography*, Saqi books, London, 2001.

[45] B.-Q. Su & D.-Y. Liu. — *Computational geometry–curve and surface modeling*, Academic press, Boston, 1989.

[46] Unicode Consortium. — « Unicode », 2003. `http://www.unicode.org`

[47] URW. — « Kerning on the fly », technical report, URW, 1991.

[48] URW. — « Phototypesetting with the URW hz-program », technical report, URW, 1991. Voir aussi [49].

[49] Peter Karow. — « Le programme *hz* : micro-typographie pour photocomposition de haut niveau », *Cahiers GUTenberg* nº 27, p. 34-70.

[50] URW. — « Phototypesetting with the URW Kq-program », technical report, URW, 1991.

[51] E. van Blokland & J. van Rossum. — « Different approaches to lively outlines », *Raster imaging and digital typography II* (edited by R. A. Morris and J. André), p. 28–33, Cambridge university press, Cambridge, 1991.

[52] E. van Blokland & J. van Rossum. — « TTX », 2002. `http://www.letterror.com/code/ttx`

[53] H. Zapf. — *Classical typography in the computer age*, Oak Knoll books, 1991.

◖ Luc Devroye
McGill University,
Montréal H3A 2K6 (Canada)
luc@cs.mcgill.ca