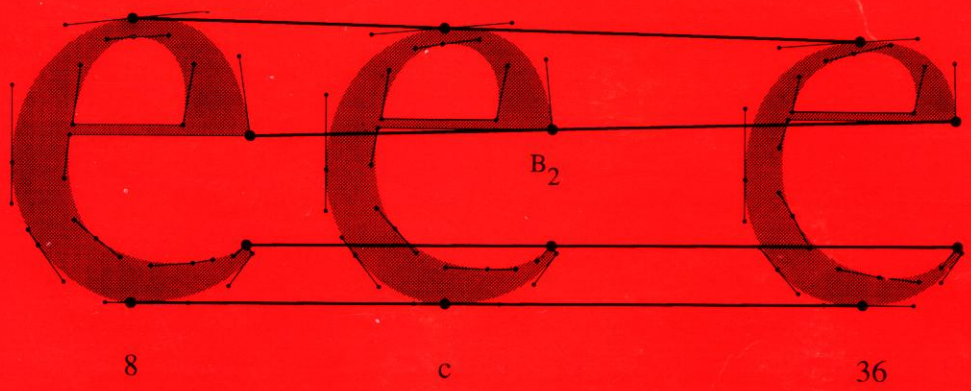
Création de fontes en typographie numérique

Jacques André

Documents d'habilitation



Création de fontes en typographie numérique

par Jacques ANDRÉ

IRISA + IFSIC

Page vide

HABILITATION À DIRIGER DES RECHERCHES

présentée devant

l'université de Rennes 1
Institut de Formation Supérieure
en Informatique et en Communication

par

Jacques ANDRÉ

Contribution à la création de fontes en typographie numérique

soutenue le 29 septembre 1993 devant le jury composé de

Jean-Pierre BANÂTRE	Président
Roger HERSCH	Rapporteur
Roger LAUFER	Rapporteur
Vincent QUINT	Rapporteur
Daniel HERMAN	Examinateur
Patrice QUINTON	Examinateur
Jean-Claude RAOULT	Examinateur

Page vide

Remerciements

Le présent mémoire traite de la partie des activités de recherche que j'ai menées depuis quelques années concernant la *typographie numérique*. Ces recherches ont été faites essentiellement à l'IRISA mais elles ont démarré en Suisse, lors de séjours au LIUF *(Laboratoire d'Informatique de l'Université de Fribourg)* puis à l'EPFL *(École Polytechnique Fédérale de Lausanne)* sur des invitations respectives du professeur Beat HIRSCHBRUNER et des professeurs Giovanni CORAY et Roger HERSCH : sans eux, ce travail n'aurait probablement jamais été initialisé et je tiens à les remercier encore pour ces invitations.

Je tiens à adresser mes remerciements à Jean-Pierre BANÂTRE, professeur à l'université de Rennes et directeur de l'IRISA, qui me fait l'honneur de présider ce jury d'habilitation à diriger les recherches et qui m'a témoigné sa confiance en m'incitant à rédiger ce mémoire. Je le remercie aussi pour l'environnement stimulant de l'IRISA qui lui doit beaucoup, même avant qu'il en soit le directeur.

Je remercie Roger LAUFER, professeur à l'université de Paris 8, Vincent QUINT, directeur de recherches à l'INRIA, et à nouveau Roger HERSCH d'avoir accepté la charge d'être les rapporteurs devant ce jury. Ces deux derniers ont partagé, depuis des années, mes préoccupations de recherche mais aussi de transmission de l'information scientifique et sont en quelque sorte mes maîtres. Je dois beaucoup à la clairvoyance de Roger LAUFER en matière de mélange « texte et informatique ».

Je remercie mes collègues rennais qui ont bien voulu participer à ce jury : les professeurs Daniel HERMAN, directeur de l'IFSIC – je lui dois notamment d'avoir été initié aux joies du Macintosh alors que le partageais son bureau en Suisse – et Jean-Claude RAOULT ainsi que Patrice QUINTON, directeur de recherche au CNRS, avec qui je partage encore le souvenir des « joies » de la photocomposeuse de l'IRISA et de sa chimie.

Mes activités en matière de typographie doivent beaucoup à plusieurs communautés, notamment les *Compagnons de Lure*, l'*Association GUTenberg*, ce groupe informel de personnes que l'on retrouve dans les comités de rédaction de la revue *EPODD-Electronic Publishing* et dans les comités de programmes des conférences *EP (Electronic Publishing)* et *RIDT (Raster Imaging and Digital Typography)* et surtout mes collègues du *Projet Didot*.

Par ailleurs, je voudrais remercier ici tous ceux qui m'ont aidé de diverses façons : mes amis de l'EPFL et, à l'IRISA, les membres (tant rennais

que grenoblois) du projet OPÉRA et de l'« atelier ». Je voudrais mentionner spécialement Pierre-Antoine ANGELINI, Éric PICHERAL et Bertrand DECOUTY pour leur aide et Philippe LOUARN pour son dévouement.

Plusieurs personnes ont bien voulu critiquer des versions provisoires de ce document et je tiens à citer notamment Marie CHARRIER, Bernard GAULLE, Christian ROLLAND et Emmanuel SAINT-JAMMES.

Enfin, je voudrais remercier les secrétaires de l'IRISA (et particulièrement Martine GLEVEO, Frédérique MOINET et Jacqueline ARNALDI) qui ont supporté, pendant des années, mes exigences typographiques.

Table des Matières

Page vide

Introduction

Faut-il donc tant de carrés pour former un O, qui est rond,
& tant de ronds pour former d'autres lettres qui sont carrées?
FOURNIER LE JEUNE, *Manuel typographique*, Paris 1764 (page xxiij)

MULTIMÉDIA, HYPERMÉDIA, ... de quoi faire retourner GUTENBERG et MAC-LUHAN dans leurs tombes, pour des raisons inverses d'ailleurs. Au delà d'un certain phénomène de mode, il est certain que – même si, du moins pour les multimédia, il s'agit plutôt de techniques – ces concepts vont sensiblement modifier la façon de communiquer des hommes. Si l'on élimine le toucher, l'odorat et le goût, nous n'avons finalement que deux sens permettant de recevoir des informations par ordinateur: l'ouïe et la vue. Or, même si la synthèse et la reconnaissance de la parole font d'énormes progrès, on ne conçoit pas une communication purement orale. Et si le papier a quelques bonnes raisons de sentir sa gloire sérieusement attaquée, les « écrans » quant à eux sont promis à un très bel avenir. Là encore, quelle que soit la qualité prochaine de la communication graphique ou iconique, on a quand même du mal à imaginer une communication d'où l'écrit serait absent. Les lettres et autres caractères de nos alphabets sont condamnés à survivre mais surtout à s'adapter au monde moderne et notamment aux écrans indissociables des hypermédia. Telle est la mission actuelle de la « typographie numérique ».

Nous voudrions, avant de présenter ce domaine, faire trois remarques.

(1) La quasi-totalité des caractères imprimés passe aujourd'hui par l'informatique. On ne peut plus parler de typographie « assistée » par ordinateur: les fontes sont des objets 100% informatiques, complètement dématérialisés.

(2) Si le passage du plomb aux pixels a pu se justifier, au départ, par des raisons économiques – c'est toujours la presse (la fabrication des journaux, en opposition au labeur, la production des livres) qui a provoqué les changements technologiques – aujourd'hui, ce sont d'autres critères que la rentabilité qui motivent les nouvelles recherches: lisibilité, qualité, confort et créativité.

(3) La typographie numérique est réputée être un domaine pointu, restreint. C'est au contraire un domaine très vaste. Le projet DIDOT[1] prépare un programme d'enseignement de la typographie numérique : aujourd'hui, pour être complet, un typographe devrait avoir des connaissances en mathématiques (courbes de Bézier, transformées de Fourier, etc.), en génie logiciel et programmation, en technologie des écrans et de la couleur, en ergonomie, en physiologie de la lecture, etc. sans oublier bien sûr l'histoire et le dessin. Ce même FOURNIER que nous citions en tête de cette introduction écrivait il y a plus de deux cents ans : *La Typographie ... est divisée en trois parties distinctes et essentielles ... mais il n'y a que celui qui réunit la science de ces trois parties que l'on puisse appeler un* TYPOGRAPHE [10]. Ce mélange de cultures est toujours indispensable aujourd'hui, sinon plus.

L'objet de ce mémoire est de montrer que désormais l'informatique permet de réaliser un vieux rêve : se débarrasser du carcan du plomb et créer de nouveaux caractères adaptés à de nouvelles technologies. Ce mémoire comprend deux parties. La première est une courte synthèse sur la typographie. Nous rappelons ce qu'est un caractère et comment on les fabriquait autrefois (chapitre 2) puis comment on les manipule aujourd'hui, c'est-à-dire ce qu'est une fonte informatisée (chapitre 3), et comment, depuis peu, on arrive à atteindre les exigences du passé (chapitre 4). La seconde partie montre plutôt des recherches auxquelles nous avons participé : nous montrerons d'abord (chapitre 5) une possibilité fondamentale en matière de créativité : on peut désormais calculer la forme d'un caractère lorsqu'on l'utilise (et non plus à l'avance) ; nous en donnerons une première application (le dessin des symboles mathématiques). Nous montrerons alors (chapitre 6) comment ce concept peut être utilisé pour créer de nouveaux caractères en fonction de contextes variés. Enfin, en guise de conclusion (chapitre 7), nous indiquerons plusieurs voies de recherche dans ce domaine méconnu.

Ce mémoire est rédigé par un informaticien et non par un artiste typographe. Notre propos n'est pas de créer de nouveaux caractères. Non pas qu'on n'en ait pas besoin (malgré les milliers de fontes existantes et les centaines proposées chaque année aux entreprises comme *Bitstream*, *ITC* ou *URW*) : à toute nouvelle technologie doivent correspondre de nouveaux caractères comme cela a toujours été le cas dans le passé [21] ; cependant, un dessin de caractère est une œuvre d'art et il faut avoir été formé à cette discipline (pendant une dizaine d'années dit Charles BIGELOW [206]) pour arriver à quelque résultat positif. Mais, c'est aux techniciens de proposer

1. Projet du programme COMETT II de la CEE. Ce programme vise à développer les formations professionnelles aux hautes technologies. Le projet DIDOT concerne la typographie numérique et a notamment pour but de préparer un *curriculum* dans ce domaine. Nous en avons publié, avec Roger HERSCH [100], une première partie pour les informaticiens. La seconde, pour les graphistes, vient de faire l'objet d'une première version [104, 106].

de nouveaux outils, voire de nouveaux concepts. C'est donc ce que nous faisons ici - proposer des outils pour une nouvelle idée: dessiner les caractères en fonction de leur environnement. Et ces outils ou idées sont justement basés sur des concepts d'informaticien: nous pensons en effet que la typographie numérique relève du même esprit que la compilation et que l'impression d'un caractère est équivalente à un appel de procédure avec tout ce que ce concept sous-entend en matière d'élaboration, de passages de paramètres ou de gestion mémoire.

On raconte que POINCARÉ, lorsqu'il interrogeait un élève sur quelque théorème, le laissait couvrir le tableau de formules puis lui disait « Effacez tout et racontez moi ça en français ». C'est un peu ce que nous avons essayé de faire ici en éliminant tout détail technique auquel nous renvoyons le lecteur par le biais de références bibliographiques. L'habitude, en informatique, est de donner la bibliographie *in fine* par ordre alphabétique et de faire des références sous la forme « américaine »: [AUTEURnn] où nn reprend les deux derniers chiffres de l'année de parution. Outre le fait que ceci pourrait poser des problèmes - [Fournier64] ne laisse guère penser qu'il s'agit du *Manuel* de FOURNIER qui date de 1764 - cette méthode ne s'applique pas à un domaine pluri-disciplinaire comme la typographie. Nous avons donc réparti cette bibliographie en divers sous-domaines et dans chacun nous reprenons le classement alphabétique puis chronologique. Mais nous numéroterons ces titres par ordre d'entrée en nous efforçant d'y faire appel en citant le ou les auteurs et parfois le titre de l'œuvre, par exemple nous parlerons du « *Manuel* de FOURNIER [10] ».

Enfin, en guise de colophon, signalons que ce texte a été composé en LaTeX (styles *habili* de Ph. LOUARN et *french* de B. GAULLE), que la majorité des figures (toujours originales si elles n'ont pas de référence) ont été programmées en PostScript natif et « collées » par `psfig` (pas toujours sans mal!). Le caractère choisi, *Lucida-bright*, a été dessiné par Charles BIGELOW et Kris HOLMES [126, 146]. Les sorties finales ont été faites sur imprimante à laser HP LJ 4M à 600 dpi. La reproduction de ce mémoire a été assurée par le service « imprimerie » de l'université de Rennes 1 sur Docutech de Xerox.

Page vide

Du plomb aux pixels

L'HISTOIRE de l'imprimerie, du livre et de l'édition a fait l'objet de très nombreux travaux[1], mais il y a peu d'études consacrées spécifiquement à l'histoire de la création de caractères. René PONOT [22] et Walter TRACY [23], par exemple, ne font pas allusion à la numérisation des caractères. Toutefois, on trouvera chez Lynn RUGGLE [116] et Richard SOUTHALL [57] les éléments essentiels.

Les principales dates de l'histoire de la production de caractères sont finalement peu nombreuses.

2.1 1435 : invention du caractère mobile en plomb

LA GRANDE INVENTION de GUTENBERG[2], vers 1435, n'est pas, comme on le croit généralement, celle de l'imprimerie (l'impression à partir de bois en relief est attestée en Chine dès la fin du premier millénaire), ni celle de la presse (en usage alors chez les vignerons), ni celle des caractères mobiles (des caractères en bois ont été utilisés dans des almanachs dès le XIV^{e} siècle). Sans nier ses compétences d'orfèvre (qu'il adapta aux techniques de gravure des poinçons), de chimiste et de métallurgiste (mise au point de l'alliage plomb-étain et plus tard antimoine, techniques de frappe, etc.), son invention est essentiellement celle du processus de fabrication, de production et de réutilisation de caractères.

Cette chaîne de production comprend les étapes suivantes (voir aussi figure 2.1) :

(1) Un poinçon est gravé à la main, à l'aide de gouges (figure 2.2). C'est une petite pièce d'acier d'environ 5 ou 6 cm de haut et de quelques millimètres carrés de surface. Ce poinçon représente exactement le caractère tel qu'il sera imprimé (figure 2.1.1).

1. Parmi les ouvrages français les plus récents, citons tous ceux donnés en bibliographie (section A.1) et notamment BLANCHARD [5], CHARTIER et MARTIN [6], DREYFUS [7, pages 184-214] et FEBVRE et MARTIN [9].

2. La littérature sur GUTENBERG est impressionnante ! On trouvera dans le très récent livre de BECHTEL [4] une synthèse sur l'histoire de GUTENBERG, mais aussi sur l'histoire de son histoire.

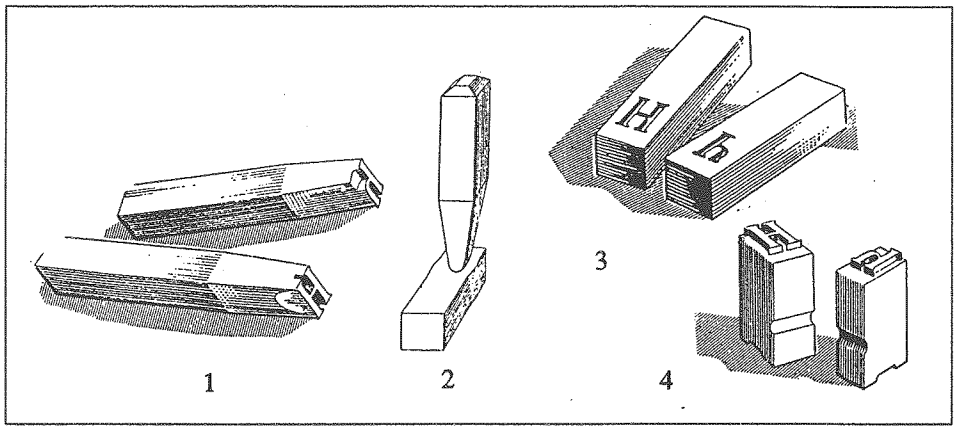


FIG. 2.1 - *Principales étapes de la fabrication de caractères en plomb (d'après* DREYFUS *[7, page187] et* MARSHALL *[18, page 44])*

(2) Ce poinçon, en relief, est alors frappé sur une matrice (figure 2.1.2) où il laisse son empreinte (figure 2.1.3).

(3) Cette matrice est ensuite introduite dans un moule (figure 2.3) où le caractère va être fondu (figure 2.1.4). Ce moule est probablement l'outil le plus génial de la chaîne: par un système de tirette, on va pouvoir donner au caractère une chasse dépendant de la lettre (un « m » est plus large qu'un «i ») tout en gardant corps et hauteur en papier constants (voir les détails techniques dans la description moderne de Stan NELSON [45].
Une fois les réglages faits pour un caractère (ou type), on peut en couler de nombreux identiques (BECHTEL [4, page 326] parle de 600 types par jour).

(4) Les caractères sont alors utilisés pour composer un texte.

(5) Une fois ce texte imprimé, les caractères sont démontés. Ils peuvent donc être réutilisés pour un autre texte.

(6) Après avoir servi plusieurs fois, les caractères finissent par s'émousser. On les refond et le plomb ainsi récupéré[3] permet de mouler de nouveaux caractères.

(7) Lorsque l'on refond ces caractères, on part de la matrice. Si vraiment celle-ci est en mauvais état, on peut toujours repartir du poinçon, mais ce cas semble avoir été très rare.

3. Le premier travail des apprentis typographes était de trier les caractères pour les remettre dans les casses mais aussi de ramasser, dans les ordures de l'atelier, le plomb qui pouvait y traîner (voir par exemple [19]).

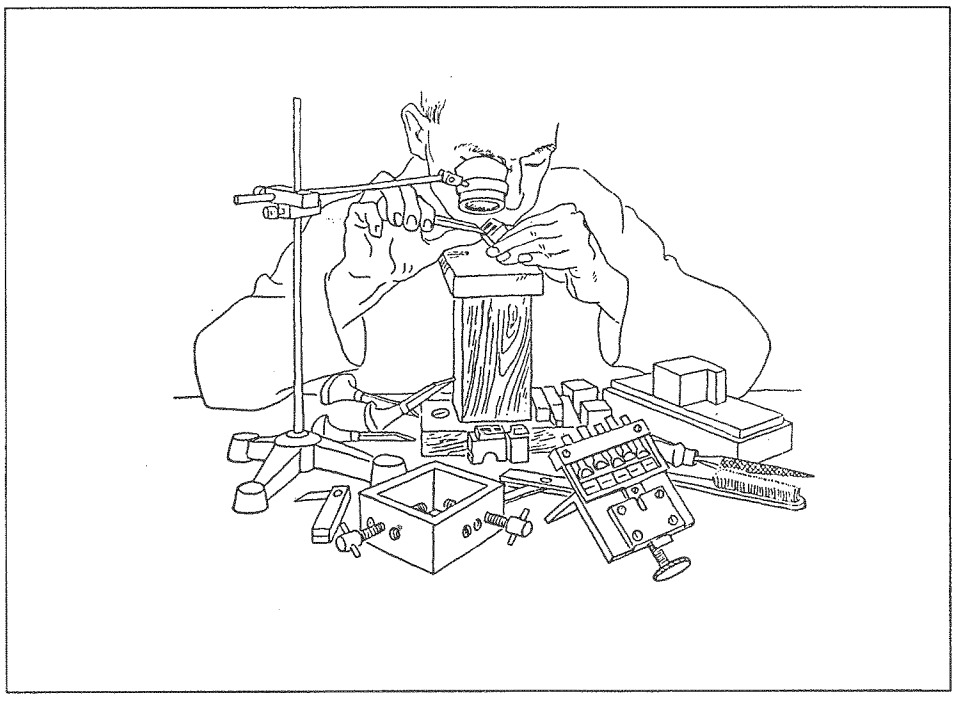


FIG. 2.2 - *Les poinçons étaient gravés à la main - d'après* TRACY *[23]*

C'est cette technique, à peine modifiée (les principales améliorations ont porté sur la composition de l'alliage et sur l'emploi de masselottes pour le démoulage), qui subsiste dans les rares ateliers où l'on grave encore des caractères (l'Imprimerie nationale à Paris a une équipe de trois personnes utilisant cette technique pour réparer les collections de poinçons conservés dans son *cabinet*). On trouvera des « films » de démonstration de cette technique « fossile » d'une part dans les actes du colloque *The computer and the hand in type design* [31] et, d'autre part, sous forme d'une vidéo réalisée, dans le cadre du projet Didot, à Reading [60].

2.2 1885 : mécanisation de la taille des poinçons

DÈS LE PREMIER quart du XIX^e siècle, de nombreuses recherches ont lieu pour fabriquer des machines à composer et débouchent sur la *Linotype*: l'idée est de couler non plus les caractères un par un puis de les composer manuellement, mais de composer, à l'aide d'un clavier comme celui d'une de ces nouvelles machines à écrire (la première Remington date de 1873), toute une ligne avec des matrices et de couler celle-ci d'un coup.

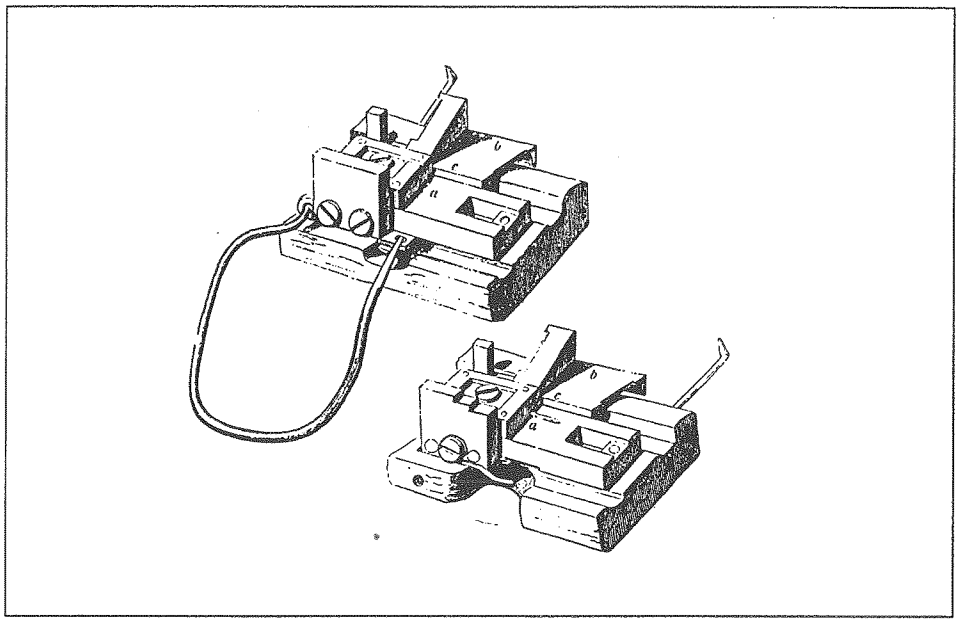


FIG. 2.3 - *Moule à caractères (d'après Bechtel [4, page 328])*

L'intérêt de mécaniser la fabrique des matrices amena un Américain, Lynn Boyd BENTON, à déposer en 1885 (c'est aussi l'année du brevet de la Linotype par MERGENTHALER) un brevet pour graver des poinçons : leur taille est un travail délicat et fastidieux. Pour un alphabet qui n'aurait que 80 signes, il faut non pas 80 poinçons mais $80 \times n$ poinçons où n est le nombre de corps désirés. Il faut en effet un poinçon pour le « a » en corps 6, un en corps 8, un en corps 9, etc. L'invention de BENTON consiste en l'emploi d'un pantographe (voir figure2.4) qui permet, à partir d'un dessin original, de produire tous les poinçons voulus.

2.3 1946 : photocomposition

DEUX INVENTIONS vont contribuer, au XX^{e} siècle, à la disparition du plomb : l'offset, procédé d'impression à plat basé sur la lithographie (W. RUBEL en 1904 à New York) et la photocomposition, en 1946 par les Français HIGONNET et MOYROUD (voir la thèse d'Alan MARSHALL [18] à ce sujet), cette nouvelle technique permettant de produire des films prêts pour les machines offset.

Le principe de la photocomposition est, *a priori*, simple (figure 2.5) : un faisceau de lumière passe à travers un disque où se trouve une lettre gravée. L'image est alors projetée sur un film photographique où elle sera

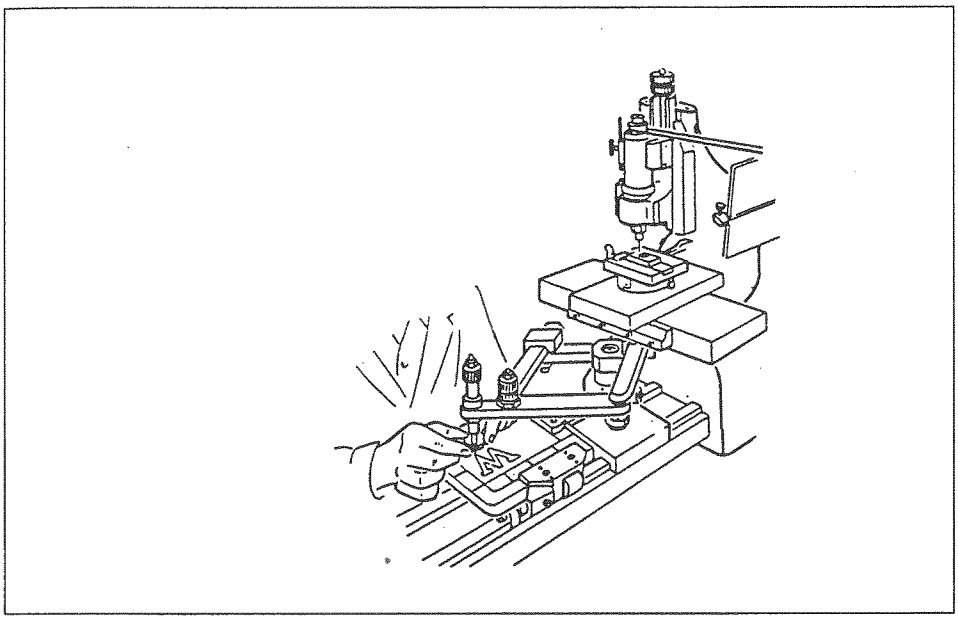


FIG. 2.4 - *Pantographe pour graver les poinçons (d'après* MARSHALL *[18, page 46])*

donc en positif. Un système de zoom permet d'avoir diverses forces de corps et de placer les lettres les unes après les autres dans une ligne. La réalisation, en revanche, est plus compliquée et il a fallu remplacer les techniques mécaniques par l'emploi d'ordinateurs puis de laser pour avoir des photocomposeuses vraiment efficaces et de très haute définition.

La fabrication des caractères s'est ramenée alors à la photographie de lettres et à leur gravure sur le disque (avec bien sûr divers problèmes techniques, comme le calage des lettres). Toutefois il a souvent fallu, pour des problèmes d'*aliasing*, de distorsion et de diffusion de la lumière, redessiner les caractères (voir figure 2.6).

2.4 1966 : numérisation des caractères

2.4.1 Convergences

Le passage de la photocomposition aux « fontes numériques » d'aujourd'hui n'est pas si immédiat qu'on pourrait le croire, surtout avec si peu de recul.

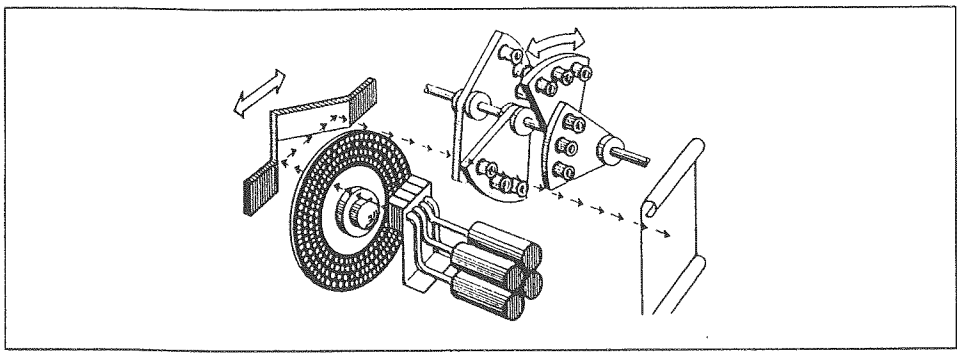


FIG. 2.5 - *Principe de la photocomposition (d'après* SEYBOLD *[117, page 74])*

Il y a d'abord eu trois voies parallèles.

(1) La voie royale de la composition au plomb qui a donc débouché sur la photocomposition des première puis seconde générations.

(2) Mais en parallèle, s'est aussi développée la branche (« honteuse » aux yeux des typographes) des chancelleries et des états-majors (plumes d'oie puis celles dites sergent-major) puis des secrétariats munis de machines à écrire.

(3) L'informatique enfin, depuis les années 1950 en gros, a eu aussi des besoins importants de sortie de documents, comptables au départ[4], ce qui a conduit au développement d'imprimantes de plus en plus rapides et de postes de saisie (sur cartes perforées ou rubans) de plus en plus proches des habitudes des secrétaires et comptables.

Il y a eu ensuite convergence de ces voies deux à deux (approximativement, vers 1970-80) :

(1) D'une part la photocomposition a suivi de très près ce qui commençait à se faire pour certains matériels informatiques (imprimantes mais surtout tubes CRT[5] utilisant des images tramées (*rasters*) c'est-à-dire basées sur des *bitmaps* ou plans de bits) : les photocomposeuses sont passées du film photo aux premières *bitmaps* (c'est sur la Digiset 50TI qu'apparurent les premiers caractères numérisés) pour devenir machines de troisième génération avec l'usage de rayons laser. Mais

4. D'où la présence sur nos claviers d'aujourd'hui des caractères « & @ $ # », ce qui ne s'explique, comme nous l'avons montré dans [121, 198], que parce qu'il s'agit de caractères utilisés en comptabilité américaine.

5. Selon Lynn RUGGLE [116] la première utilisation d'un tube cathodique pour le tracé de lettres remonterait à 1964.

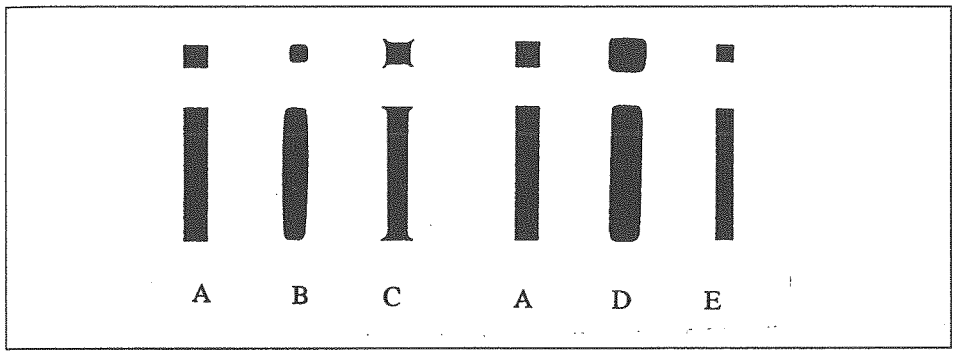


FIG. 2.6 - *Les caractères doivent être redessinés en fonction de la technologie employée: en A, le caractère théorique; en B ce qu'il donnerait en photocomposition; en C, la façon dont il doit être redessiné pour qu'une photocomposeuse donne le dessin A; en D, ce que donnerait une machine à écrire à partir de A; en E le caractère redessiné pour qu'une machine à écrire donne le caractère A (d'après* FRUTIGER *[33]).*

c'est la notion de matrice qui prévaut encore. Les lettres pour photocomposeuses sont alors dessinées directement sur la grille (agrandie), pixel par pixel. C'est la technique employée par les pionniers du dessin de caractères numérisés : aux États Unis Ch. BIGELOW [102] et en France Adrien FRUTIGER[32] avec son *Univers* et Ladislas MANDEL [43], auteur, par exemple, du *Galfa*, un caractère spécialement dessiné pour les tout petits corps des annuaires téléphoniques (voir figure 2.7).

(2) La seconde convergence (informatique et matériels de secrétariat) a donné naissance à ce qu'on appelle parfois la bureautique[6] : le début de la décentralisation des sites informatiques et des système distribués a amené les constructeurs d'ordinateurs à utiliser des machines à écrire comme périphériques ou terminaux, voire postes de saisie. IBM - qui était leader et en informatique et en matériel comptable ou de bureau - a alors sorti sa « machine à boule » et sa variante *Selectric* permettant de disposer de caractères à chasse variable[7]. Les deux voies, informatique et secrétariat, ont alors évolué ensemble avec les « systèmes de traitement de texte ».

6. Par bureautique, on entend non seulement le traitement des textes, mais aussi la communication, incluant donc ce que NORA et MINC ont appelé, en 1975, la *télématique* [208]. L'histoire moderne des jeux de caractères est très liée à celle des transmissions : les normes (du Telex au Fax, voire à SGML, Iso-Latin et Unicode) ont toujours été écrites pour « l'échange d'informations » et ce par des ingénieurs des télécommunications. Voir la synthèse *Télématique* du CCETT[202] et [121, 198].

7. Nous avons montré [119] qu'il s'agit non seulement d'une révolution technique mais aussi typographique : c'est ainsi que le fameux caractère *Courier* a été conçu pour cette nouvelle technologie (voir figure 2.6.)

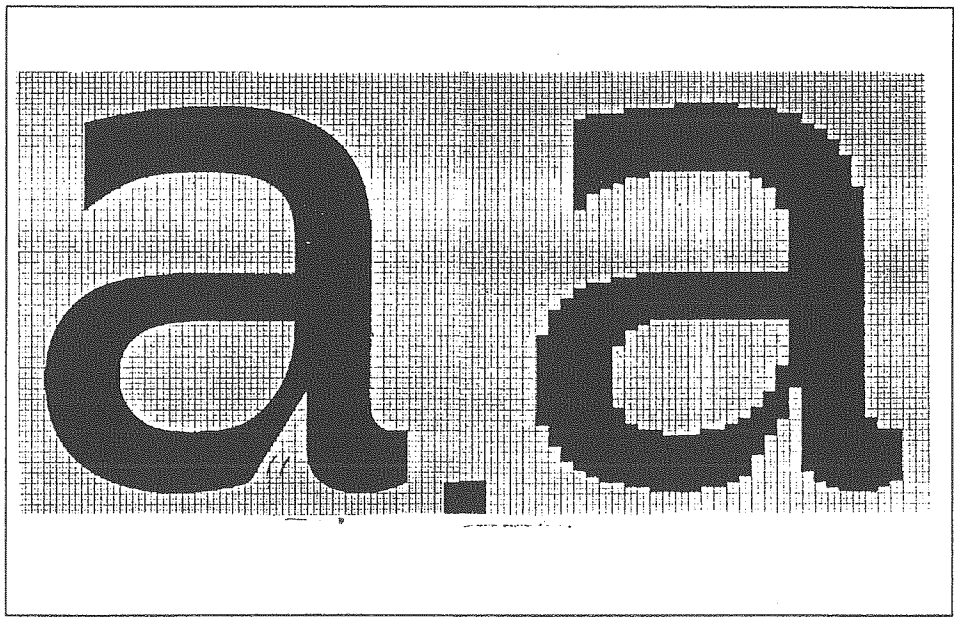


FIG. 2.7 - *Les pixels des caractères numérisés ont d'abord été dessinés « à la main ». Ici, un a de* Galfa *conçu par* LADISLAS MANDEL, *à gauche en continu, à droite « prédigitalisé » (extrait de [55, page 46])*

Si, pour les secrétariats, c'était un très gros progrès, pour les typographes, ces systèmes étaient tout simplement inadmissibles: non seulement la définition des caractères était très basse (les aiguilles des imprimantes faisaient quelques dizaines de millimètres de diamètre ce qui ne permettait guère de lissages des panses de lettres) mais en plus, pour des raisons d'économie de mémoire, les jeux de caractères étaient très limités, voire incomplets (pas ou peu de lettres accentuées) et souvent erronés (confusion entre l'italique et le penché par exemple). Il n'empêche que cette période, éphémère, a eu deux conséquences fondamentales et opposées:

- d'une part, les secrétariats ont découvert la typographie et ont perdu les mauvaises habitudes causées par les contraintes des machines à écrire (emploi du souligné, chasses fixes, non accentuation des capitales, etc.);
- d'autre part, les typographes ont vu leur tour d'ivoire s'effondrer mais, si certains se sont ouverts aux nouvelles technologies, d'autres (et c'est malheureusement particulièrement vrai en France) freinent de tout leur poids cette évolution[8].

8. On voit encore des gens reprocher « au Macintosh » de confondre italique et penché et de

2.4.2 Mutations

Enfin, on verra naître les caractères numérisés d'aujourd'hui définis non plus comme une image de points mais décrits par un contour que l'on remplit. Au delà du changement de méthode, il y a quelque chose de fondamental dans cette « mutation » : remplacer une image par la procédure qui la construit. Les premiers travaux remontent à la fin des années 60 (Mergler et Vargo aux USA). L'apport de français (et notamment de l'École des Mines de Saint-Étienne) aura été important : BLOCH [128], COUEIGNOUX [130] [105], GUEDJ [129], etc. Il faudra attendre 1973 pour voir le premier système professionnel de dessin : *Ikarus* des Allemands KAROW, RUBOW et WEBER [111]. METAFONT, de KNUTH [142], date de 1979, mais c'est bien plus qu'un système de dessin de fontes : c'est un système de meta-fontes[9]. C'est dans le début des années 1980 qu'apparaissent les premières normes en matière de « fontes » toutes liées à un « fondeur » : URW, Bitstream, etc.

En fait, la révolution commerciale viendra du laboratoire PARC *(Palo Alto Research Center)* de Xerox ou plutôt de ses transfuges qui sont à l'origine d'Apple et surtout d'Adobe qui ont inventé PostScript et la notion de « langage de description de page ». Ce langage, nous y reviendrons au chapitre 3, a non seulement été la première normalisation réussie de tous ces moyens de restitution (imprimantes, écrans, photocomposeuses, etc.), mais est vraiment à l'origine de la notion de « fonte » actuelle, c'est-à-dire d'un objet informatique que l'on manipule comme un simple fichier, que l'on copie (et co-pille[10]) à volonté, etc.

C'est avec cette notion de fonte à la PostScript que fonctionnent aujourd'hui tous les systèmes commerciaux de saisie (comme *Typo* et *Ikarus*), de dessin (*Fontographer* ou *Fontstudio*) et de modification de caractères (*Fontoshop* ou *Illustrator*) ; voir section 3.6.

Mais ce domaine n'en est pas pour autant stabilisé. À présent que les problèmes de « rendu » sont à peu près réglés, il faut s'attaquer vraiment à la typographie : espacement entre lettres, forme des lettres selon leur taille, etc. (voir chapitre 4). Il ne s'agit pas, par conservatisme, de copier un passé contraignant, mais bien au contraire utiliser les techniques d'aujourd'hui pour améliorer la lecture, sur papier mais aussi sur écran.

---

ne pas avoir de lettres accentuées, ce qui était en fait un défaut des imprimantes et non du Mac lui-même ; ce ne serait pas trop grave si ces personnes n'avaient un pouvoir de décision - voir à ce sujet le rapport PEIGNOT [52].

9. KNUTH s'est lui même longuement expliqué sur cette notion dans un article [142] qui a été largement commenté par Douglas HOFSTADTER dans *Ma thémagie* [207, pages 249-288].

10. L'expression a été créée par une cellule formée de personnes du ministère de l'éducation nationale et de syndicats de l'édition et fait allusion aux abus de l'emploi de la photocopie dans l'enseignement. Rappelons que les fontes font aussi, en tant qu'œuvre d'art, l'objet de réglementations. Mais elles sont assez inadaptées au monde informatisé d'aujourd'hui. Voir, au sujet du piratage des fontes, les articles de BERTRAND [205] et de BIGELOW [206].

2.5 Remarques

Avec un peu de recul, on remarque maintenant plusieurs choses.

(1) Le caractère, objet à trois dimensions (le « caractère » en plomb – avec la même ambiguïté que pour le mot gravure : quand on dit « caractère », s'agit-il de l'objet ou de sa marque imprimée ?), a été ramené à deux dimensions (le film des photocomposeuses) pour, finalement, être complètement dématérialisé : aujourd'hui, un caractère est un objet asbtrait comme peut l'être une procédure informatique. René Ponot dit d'ailleurs que *la composition, numérisée ou non, n'a conservé de l'expression « caractère d'imprimerie » que ce qui se rapporte à la trace imprimée dudit caractère* [22].

(2) Comme l'a fait remarquer Richard SOUTHALL [57], les rapports entre créateur et outils ont aussi évolué et sont loin d'être clairs aujourd'hui.

(3) Chaque nouvelle technologie a d'abord « copié » les caractères de la précédente : le plomb a commencé par copier l'écriture manuscrite, la photocomposition n'a fait que copier les caractères du plomb et les premières fontes numérisées ont été des copies de l'existant. Il faut toujours un certain temps pour voir des caractères adaptés aux nouvelles technologies (comme, récemment, *Univers* de FRUTIGER [32] pour les premiers caractères spécialement dessinés pour photocomposeuse ou *Lucida* de BIGELOW & HOLMES [126] pour les imprimantes à laser de basse définition).

Restent maintenant deux questions :

(1) la technique de dessin des caractères telle qu'elle est pratiquée aujourd'hui (description du caractère par son contour), se prête-t-elle à la création de caractères ? nous donnerons au chapitre 7 quelques unes des idées qui ont été émises à ce sujet par des théoriciens ;

(2) à part la qualité et la possibilité d'afficher des caractères sur des écrans, qu'y a-t-il de neuf par rapport ... au plomb ? ce sera l'objet de notre chapitre 5.

Fontes numériques

DANS CE CHAPITRE, nous faisons un rapide survol des techniques employées en typographie numérique. Le but est de situer les divers concepts qui seront utilisés dans les chapitres suivants.

L'édition électronique utilise trois types d'acteurs (produits logiciels ou matériels): des formateurs (éditeurs ou systèmes de traitement de texte), des organes de sortie (imprimantes, photocomposeuses, écrans) et l'ensemble des « fontes ». Derrière ces outils se trouvent trois types de personnes, respectivement l'auteur (ou le compositeur), le lecteur et le dessinateur de caractères.

Le problème de la typographie numérique est de faire cohabiter les fontes dans ce système, étant donné que ces acteurs peuvent intervenir à des moments différents et que les informations passées varient d'un système à l'autre. Nous avons signalé [122] que la notion de processus coopérants s'applique à ces acteurs.

Le chapitre précédent a montré que la numérisation des caractères était issue de la convergence d'intérêts divers mais confus. Sur un plan technique, cette évolution, encore en cours, est la suivante:

(1) utilisation directe de plans de bits (ou *bitmaps*);

(2) description des caractères par leurs contours;

(3) meilleure adaptation des contours à la grille en tenant compte des propriétés typographiques individuelles des caractères;
à ce niveau, on a la qualité de la photocomposition de seconde génération (films photo);

(4) prise en compte des propriétés typographiques des caractères entre eux (ajustement optique, espacement entre caractères);

(5) utilisation de niveaux de gris et non plus du seul rapport blanc/noir.

Certes, il y a une recherche « technologique» sous-jacente. Mais la finalité est essentiellement une qualité toujours meilleure pour les caractères. Et la difficulté est que si un typographe sait dire qu'un caractère est « beau», il est bien incapable de dire pourquoi, de quantifier, de formaliser. Les recherches en typographie numérique sont désormais indissociables de recherches

en typographie traditionnelle et sur la lisibilité. C'est donc cette dualité, technique et visuelle, qui va nous servir de base dans les pages à venir. Nous essayerons aussi de montrer comment l'analogie entre l'impression d'un caractère et un appel de procédure permet de voir, sous un jour nouveau, la typographie.

Mais, nous allons d'abord commencer par un rapide survol de PostScript : ce langage de description de page étant devenu une référence (voire une norme *de facto*) en matière d'impression de documents, il est normal que nous y fassions largement appel.

3.1 PostScript

PENDANT de nombreuses années, chaque photocomposeuse, chaque imprimante, chaque *driver* avait sa propre façon de recevoir les informations et de piloter des « moteurs d'impression » (voir notre article dans le *Traité d'informatique - techniques de l'ingénieur* à ce sujet [63]). Diverses normes ont été proposées, notamment dans le milieu des arts graphiques, à la recherche d'une sorte de balisage universel des photocomposeuses. Assez curieusement, cela a débouché sur SGML (*Standard General Markup Language* - on trouvera dans [200] une bonne introduction à ce langage) et non sur une norme de photocomposition. C'est la société Adobe qui, partant des travaux du Parc de Xerox, a vraiment lancé cette notion de « langage de description de page ».

Les ouvrages de base pour ce langage sont les manuels publiés (très intelligemmment, c'est suffisamment rare pour le signaler) par le vendeur/créateur, Adobe, plus connus sous le nom de « livres colorés » d'Addison Wesley (respectivement rouge pour le manuel de référence [86], bleu pour le manuel pédagogique [88], vert pour le programmeur avancé [89], noir pour les fontes [87], etc.). En français, il ne nous semble intéressant que de citer trois publications: la traduction française du livre bleu [88], un ouvrage d'EMINET [94] qui a la grande force de ne pas être un plagiat des ouvrages d'Adobe et la petite synthèse de Bruno BORGHI [93].

PostScript a pour principes de base :

- d'être un langage de programmation ;
- d'être indépendant du matériel (photocomposeuse, imprimante à laser, écran, etc.) du moment que c'est un « moteur d'impression » à trame ;
- de ne pas confondre les fonctions de composition (par exemple la justification ou la division des mots) et celles d'impression ;

- de considérer que le contenu d'une page est défini par l'algorithme qui la produit ;

- d'accepter les mélanges de textes, images et graphiques.

Du point de vue langage, PostScript est inspiré de Forth : c'est un langage procédural à notation préfixée. On y trouve donc la majorité des choses ce que l'on trouve classiquement dans les langages de programmation et d'aucuns se servent de PostScript comme d'autres de Pascal!

Le modèle graphique de PostScript s'appuie sur un petit nombre de concepts, simples, complets et cohérents ; pour le graphique, ils sont au nombre de trois : les dessins (au trait), les images (formées de points) et le texte. On peut même les ramener à deux car les textes sont faits de caractères qui sont eux-mêmes des dessins.

PostScript suit un modèle procédural : il n'y a pas (contrairement par exemple à GKS) d'objet graphique (ni cercle, ni carré, etc.) mais seulement des procédures (pour tracer un cercle, pour colorier le contenu d'un carré, etc.). Les caractères, qui ne sont que des dessins, sont donc des procédures.

Ces procédures-caractères sont définies dans des dictionnaires, appelés « fontes », et leur emploi se limite finalement à deux groupes d'opérations : la sélection d'une police (opérateurs `findfont`, `makefont` ou `scalefont` et `setfont`) et l'écriture d'un texte (`show` et ses quelques variantes). Cette simplicité d'utilisation cache cependant toute une machinerie sur laquelle nous reviendrons (section 3.4.1).

Une page PostScript correspond à l'espace normé de $-\infty$ à $+\infty$, les mesures étant en points picas arrondis[1] La page A4 d'une imprimante n'en représente qu'une partie, en gros pour x de 50 à 600 points et pour y de 50 à 850 points.

La figure 3.2 montre le programme PostScript très simple qui produit la figure 3.1.

PostScript est un langage de marquage (il trace des lettres) et non de composition (comme TEX, Word ou Grif) : c'est à ces formateurs de faire tous les calculs de justification, de décider des divisions de mots en fin de ligne (la ligne est d'ailleurs une notion inconnue de PostScript), etc. Pour cela, il faut que PostScript leur fournisse les informations métriques des caractères : c'est le rôle de l'AFM (voir section 3.4.2).

Depuis sa création, PostScript a essentiellement évolué dans deux directions :

(1) le développement des notions de couleur,

1. Un point PostScript = 1/72 de pouce, soit 0,3528 mm. Voir la section 3.4.3 au sujet des diverses valeurs des points typographiques.

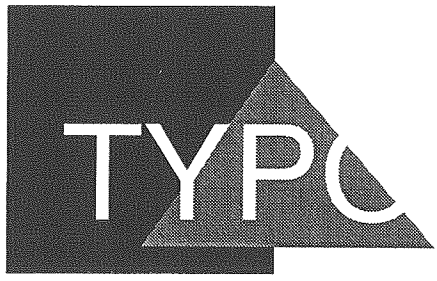


FIG. 3.1 - *Résultat du programme PostScript ci-dessous*

```
100 100 moveto % départ (bas gauche du carré en
                % x=100 points, y=100 points)
200 100 lineto % horizontale vers le point (200, 100)
200 200 lineto % verticale montante
100 200 lineto % horizontale vers la gauche
closepath      % retour au départ: on a tracé un carré
fill           % remplir ce carré de noir
150 110 moveto % nouveau pt de départ (coin gauche triangle)
260 110 lineto % horizontale
200 180 lineto % diagonale
closepath      % fermeture d'un triangle
.5 setgray     % prendre du gris
fill           % en remplir ce triangle
/Helvetica findfont 50 scalefont setfont
               % on vient d'appeler la police
               % Helvetica en corps 50
1 setgray      % prendre de la peinture blanche
120 120 moveto % point de départ
(TYPO) show    % écrire les lettres TYPO dans cette police
showpage       % imprimer la page
```

FIG. 3.2 - *Programme PostScript dessinant la figure ci-dessus*

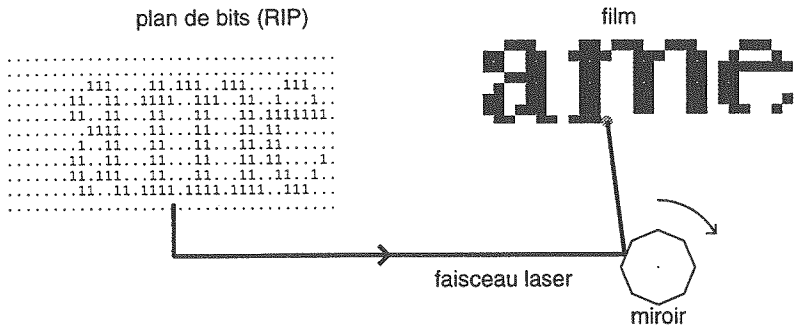


FIG. 3.3 - *Schéma de principe d'un « moteur d'impression à trame »*

(2) la mise à disposition de choses cachées aux utilisateurs, notamment en ce qui concerne les fontes (voir ci-dessous 3.3.2).

3.2 Notion de plan de bits

LES ÉCRANS, les imprimantes à laser et la quasi-totalité des photocomposeuses utilisent aujourd'hui des caractères numérisés, affichés, imprimés ou flashés sous forme d'une matrice de points. Ces moteurs d'impression sont tous basés sur le même principe: une photocomposeuse, par exemple, reçoit une matrice de points binaires (0-1, indiqués dans la figure 3.3 par un point ou le chiffre 1); une image de cette matrice est projetée par balayage d'un faisceau scanner (les valeurs binaires 0/1 devenant alors éclairé/non-éclairé). Les photocomposeuses ont souvent ce mécanisme d'impression dans une machine et l'interpréteur PostScript dans une autre (appelée RIP, *raster image processor*), tandis que, en général, les deux sont réunis dans les imprimantes à laser.

Les cases ou points de cette matrice s'appellent « pixels » (contraction de *picture elements*, éléments d'image) et les matrices sont les *bitmaps*, parfois appelées « grille de points discrets » et que nous appelons ici plans de bits. Les dimensions de ces pixels dépendent de la « définition » (traduction correcte de l'anglais *resolution*) de l'imprimante c'est-à-dire de la taille des grains utilisés: en photocomposition on utilise des films photographiques dont les sels d'argent sont très petits; en xérographie (le procédé de photocopie utilisé actuellement par pratiquement tous les copieurs et imprimantes à laser) on utilise des grains de sélénium qui sont beaucoup plus gros. Cette définition dépend aussi de la finesse du faisceau laser. On mesure cette définition en nombre de pixels noircis par centimètre mais l'origine américaine des matériels fait que l'on parle de dpi (*dots per inch*) ou de « ppp » (points par pouce). On a, par exemple, 300 dpi pour une

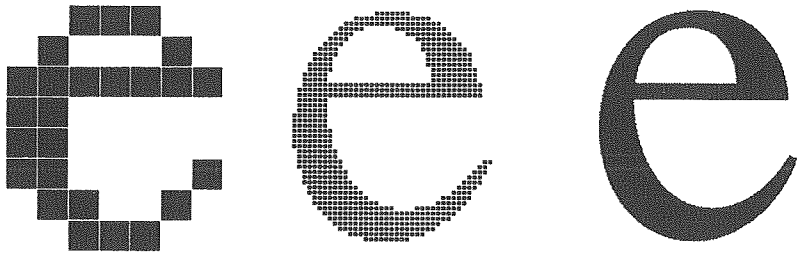


FIG. 3.4 - *La qualité de la restitution d'un caractère dépend de la définition du moteur d'impression. Ici le même caractère en corps 10 (agrandi environ 15 fois) avec une définition de 300, 1200 et 4800 dpi. Voir figure 3.12. Cette figure nous a été fournie par Claude* BETRISEY, *EPFL, Lausanne.*

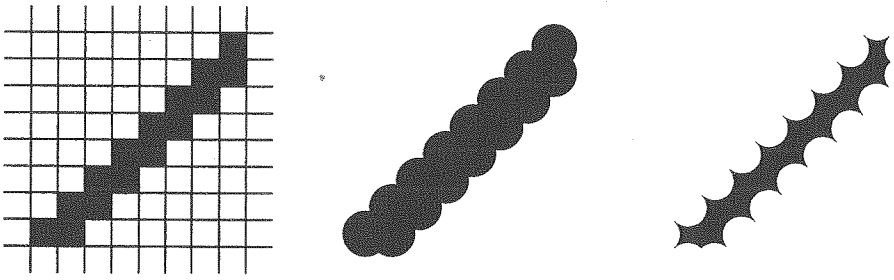


FIG. 3.5 - *Un plan de bits* (bitmap) *théorique (à gauche) et son aspect réel issu d'un moteur d'impression à insolation noir sur blanc (au centre) ou blanc sur noir (à droite)*

imprimante à laser, 2400 dpi pour une bonne photocomposeuse, etc. (figure 3.4).

Cette définition n'est pas obligatoirement la même horizontalement que verticalement (points rectangulaires de la figure 2.7).

On a l'habitude de représenter ces pixels comme des carrés ou des rectangles, mais en fait ce sont des taches plutôt circulaires ou ovales (la forme dépend des grains du tonner employé, du foulage de l'encre sur le papier, etc.). Par ailleurs, contrairement à ce que l'on croit, toutes les imprimantes ne mettent pas du noir sur une page blanche. Par exemple, certaines imprimantes (de chez Xerox ou HP) peignent complètement la page en noir puis détourent les lettres par un faisceau de lumière: les points ne sont alors plus des ronds mais les étoiles à quatre branches entre les rond blancs (voir figure 3.5). Les systèmes de création de fontes peuvent en tenir compte, mais alors il ne faut pas utiliser une fonte d'un type sur une imprimante prévue pour un autre type (MACKAY [148]).

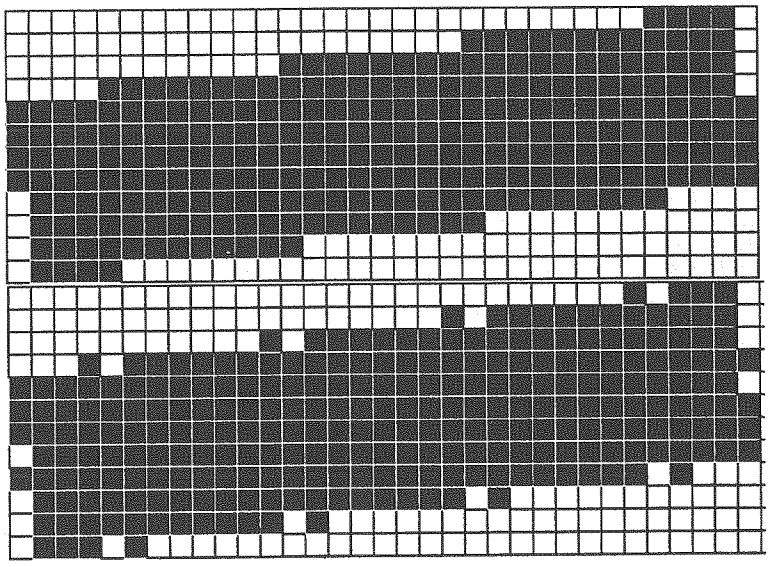


FIG. 3.6 - *L'effet d'escalier d'une diagonale peut être atténué en répartissant mieux les pixels* (half-bitting) *(d'après Beat* STAMM *[155])*

Défauts dûs aux plans de bits

Les plans de bits, matrices de points, relèvent du « discontinu » alors que les caractères relèvent du « continu ». En nous inspirant d'une métaphore de Richard SOUTHALL, tout le problème de la typographie numérique est de vouloir faire avec des briques de Lego (et ce sans les casser) une Ferrari de 10 cm de hauteur. Un problème bien connu, par exemple, est celui des diagonales qui prennent presque toujours une allure d'escalier *(aliasing).* L'art du dessinateur est alors de casser ces escaliers, par exemple en les atténuant (voir figure 3.6).

Un autre phénomène est celui des pointes qui ont tendance à s'émousser (ou du moins à paraîtres émoussées) et des creux qui ont tendance à se remplir (ou qui le paraissent). Là encore (voir figure 3.7), il faut ajouter ou supprimer des pixels, technique connue en anglais sous le nom de *half-bitting.*

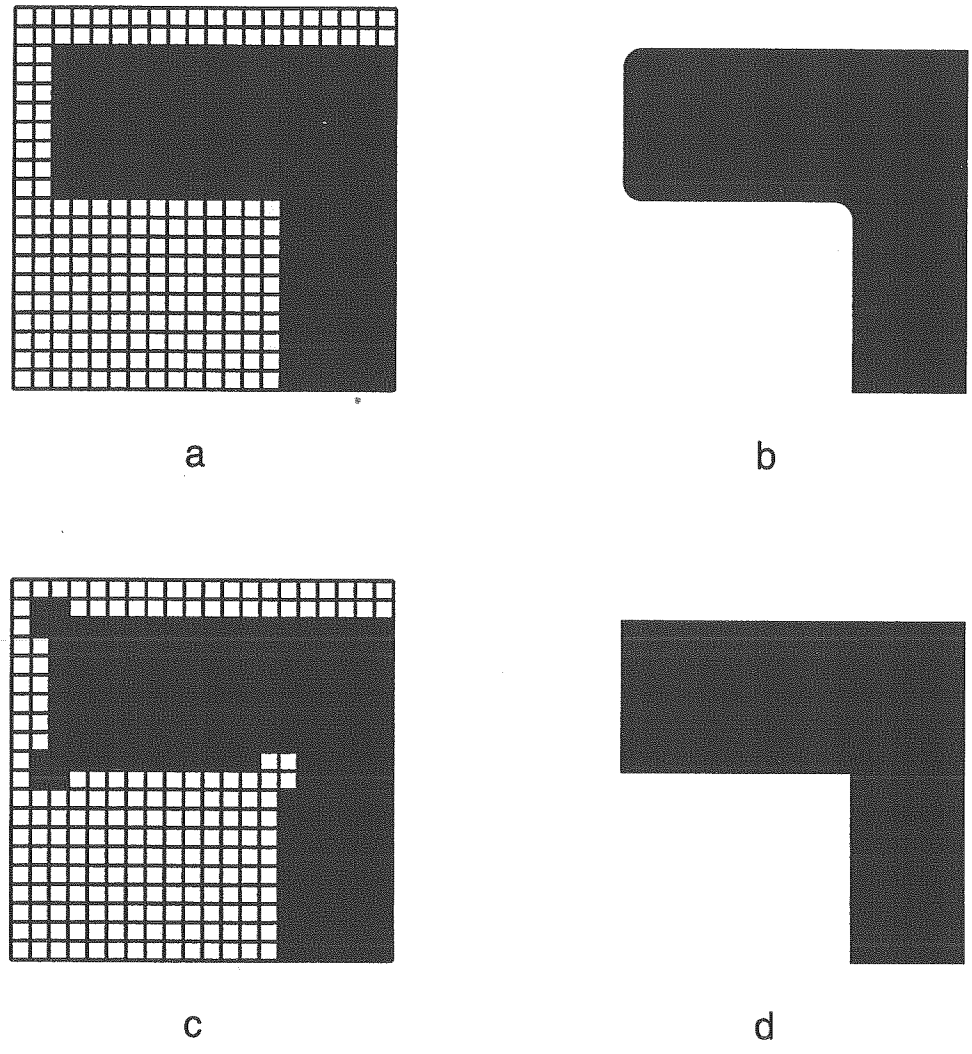


FIG. 3.7 - *Si on donne le plan de bits (a), on obtient l'image (b) ; pour avoir l'image (d), il faut tricher et donner le plan (c). Voir aussi la figure 2.6. D'après* RUBINSTEIN *[115, pages 79–80]*

Plans de bits et « variables » en programmation

Ces plans de bits, fournis « en entrée » aux moteurs d'impression, sont donc la « sortie » des programmes d'impression. Ce sont des « valeurs » qui peuvent être soit des constantes calculées une fois pour toutes, soit le résultat de l'élaboration d'un calcul. Par analogie, prenons l'exemple de matrices de Lagrange utilisées pour le calcul d'un polynôme de degré n. Il y a trois façons de faire :

(1) ou bien on calcule ces matrices à la main et on les entre ainsi dans le programme de calcul ;

(2) ou bien on écrit un programme qui va calculer ces coefficients de Lagrange, constantes que l'on pourra entrer ensuite dans le programme de calcul ;

(3) ou bien, enfin, on met dans le programme de calcul du polynôme une procédure qui calcule les matrices de Lagrange.

On a la même chose avec les plans de bits : ou bien on les construit à la main, ou bien on utilise des programmes pour les construire et ils sont alors chargés dans l'imprimante comme une constante, ou bien on donne à l'imprimante une procédure qui les calcule.

Pendant longtemps ces matrices de caractères ont été créées (à la main ou à l'aide d'une souris sur écran[2]) et ainsi chargées, sous forme de fichiers binaires, dans l'imprimante. Mais cette technique avait deux inconvénients majeurs.

(1) D'une part une fonte complète, même après compactage, prend une place énorme : si on ne compte qu'une centaine de caractères pour une dizaine de corps (6, 8, 9, 10, 12, etc.) pour une imprimante à seulement 300 dpi (chaque caractère occupant en moyenne 12 pixels de hauteur sur 6 en largeur) ça fait au bas mot $100 \times 10 \times 12 \times 6 = 72000$ pixels, 4 fois plus (soit 288000 pixels) si on prend un romain, un gras, un italique et un gras italique ; à 4800 dpi, ça fait dans les 28 millions de pixels. De plus ceci n'est valable que pour un type précis d'imprimante.

(2) Mais surtout, on ne peut pas faire subir à ces matrices de pixels des rotations (pour écrire en biais ou sur le pourtour d'un cercle par exemple) ou autres transformations géométriques sans risque de dégradation car la topologie de ces caractères est alors perdue (voir plus bas le §3.3.2 sur les *hints*). De même, les défauts montrés dans les figures 3.6 et 3.7 ne peuvent bien être corrigés qu'en fonction de cette topologie ; on a donc intérêt à la conserver le plus longtemps possible.

2. De nombreux utilitaires ont été créés dans ce but ; par exemple, à l'Inria-Sophia, par Vania JOLOBOFF [140].

Ces caractères matriciels ne sont donc plus toujours construits puis chargés une fois pour toute dans une imprimante. On les utilise toutefois encore dans certains cas.

(1) METAFONT peut fournir des fontes au sens de PostScript. Mais en général, on sort ces fontes sous forme de plans de bits qui sont chargés dans l'imprimante au moment de l'impression. Deux raisons à cela: d'une part, METAFONT fait toutes les adaptations à la grille (voir section 3.3.2) et sort donc des caractères de qualité et, d'autre part, il est rare que les textes produits par TEX aient besoin de faire des rotations sur des caractères, le nombre de fontes nécessaires restant alors raisonnable. L'inconvénient bien sûr est qu'un texte « formaté » pour une imprimante à 300 dpi reste à la définition de 300 dpi même sur une photocomposeuse à 4800 dpi; par ailleurs, il n'est pas possible de créer de fontes dynamiques (voir chapitre 5).

(2) Les plans de bits sont encore utilisés pour pratiquement tous les écrans: ils ont une définition assez grossière pour ne pas nécessiter trop de pixels par caractère. Et là non plus, on n'utilise guère de transformations affines sur les caractères. Par ailleurs, les techniques de *grey-scale* (caractères à niveaux de gris - voir ci-dessous §3.7.3) permettent de rendre ces caractères d'écran de plus en plus lisibles.

(3) Enfin, il est amusant de signaler que les images de synthèse utilisent souvent des caractères sous forme de plans de bits constants. Outre, probablement, l'absence d'interprète PostScript très efficace, la raison principale nous paraît beaucoup plus « culturelle »: les graphistes ont tendance à considérer que l'important c'est l'image, pas le texte qu'elle supporte.

3.3 Définition des caractères par contours

LA PRINCIPALE méthode utilisée aujourd'hui consiste non plus à donner le plan de bits à l'imprimante, mais à lui donner une description géométrique du caractère et à laisser l'interprète PostScript (qui est dans l'imprimante) calculer ce plan de bits en fonction du corps et de l'orientation du caractère, en fonction de la grille de cette imprimante, mais aussi en fonction des *hints* (indications sur les « propriétés » typographiques du caractère, voir §3.3.2).

Ces « contours » (figure 3.8), *outlines* en anglais, peuvent être des courbes quelconques: en général, un « o » n'est pas réductible à deux cercles emboîtés ni un « i » à un rectangle surmonté d'un point. Il faut alors les « approcher » au sens mathématique. Le problème est donc de trouver une approximation qui soit peu coûteuse en place, peu coûteuse en temps et

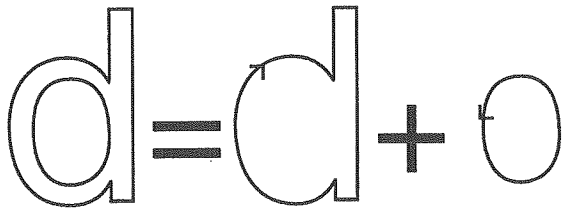


FIG. 3.8 - *Un caractère est défini par un ensemble non connexe de contours (orientés, ce qui définit extérieur/intérieur)*

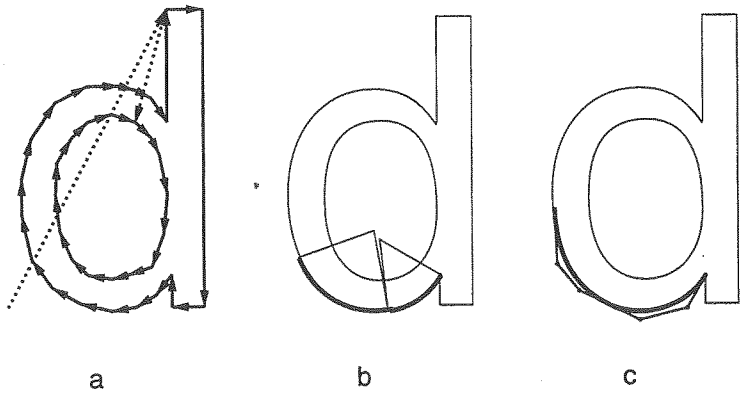


FIG. 3.9 - *Approximation d'un « d » à l'aide de vecteurs (a), d'arcs de cercles (b) et de courbes de Bézier (c)*

qui donne une bonne qualité pour le résultat final. Il faut aussi trouver des algorithmes qui, partant d'une courbe dont on connaît le contour, soient capables de colorier la surface correspondante.

On a d'abord utilisé, vers les années 1970, des vecteurs[3] (figure 3.9.a). Là encore, les tables prenaient trop de place et la méthode ne donnait pas des courbes très bien lissées.

On a ensuite remplacé ces vecteurs (qui sont des courbes du premier degré) par des morceaux de courbes décrites par des polynômes du second degré, notamment par des arcs de cercle (figure 3.9.b). Un arc peut être défini par son centre, le rayon et les angles des points d'extrémité et de départ. Des fontes entières ont été codées ainsi, notamment par *Bitstream* (SEYBOLD, [117, page57], en donne des exemples). URW puis *True Type* ont ensuite utilisé des quadratiques (coniques).

3. C'est de là que vient l'expression « police vectorielle » que l'on utilise encore (mais à tort puisqu'il n'y a plus de vecteurs dans la description actuelle des contours par courbes de Bézier).

Mais les raccords entre morceaux de courbes ne sont pas toujours fameux avec des courbes du second degré. Maintenant, les caractères sont définis par des morceaux de courbes qui se mettent bout à bout et qui doivent avoir une certaine continuité à leurs jonctions (continuité d'ordre 1, tangente commune, par exemple) et que l'on appelle des *splines*. Parmi ces courbes se trouvent les polynômes du troisième degré (cubiques) dont les courbes de Bézier sont un cas particulier (figure 3.9.c). D'autres courbes ont aussi pu être proposées (voir par exemple les colloques RIDT [99] [114] [98]), mais les courbes de Bézier sont encore les plus employées (notamment par Adobe) car finalement elles donnent des résultats très satisfaisants dans le rapport qualité/coût de calcul.

L'utilité de courbes de plus haut niveau (notamment de degré 5, des quintiques) est encore à étude. Il semble qu'elles puissent avoir quelque avenir, notamment en calligraphie et dans l'étude de la jonction des caractères manuscrits (travaux de KOKULA [185]).

Courbe de Bézier

Ces courbes du troisième degré ont été mises au point par le Français Pierre BÉZIER alors qu'il travaillait chez Renault sur des programmes de dessin assisté par ordinateur de capots de voitures. On trouvera dans de nombreux ouvrages des études complètes de ces courbes, par exemple [109] [183].

Les cubiques sont définies par les équations paramétriques

$$\begin{aligned} x(t) &= a_x + b_x \cdot t + c_x \cdot t^2 + d_x \cdot t^3 \\ y(t) &= a_y + b_y \cdot t + c_y \cdot t^2 + d_y \cdot t^3 \end{aligned}$$

celles de Bézier se ramenant au polynôme

$$P(t) = P_0 \cdot (1-t)^3 + P_1 \cdot 3 \cdot t(1-t)^2 + P_2 \cdot 3 \cdot t^2(1-t) + P_3 \cdot t^3$$

avec $t \in [0, 1]$.

Voici quelques propriétés de ces courbes.

- Étant donné 4 points P_0, P_1, P_2, P_3, il existe une seule courbe de Bézier passant en P_0 et P_3, ayant P_0-P_1 et P_2-P_3 comme directions de leurs tangentes ; P_0, P_1, P_2, P_3 sont les points de contrôle.

- Les points P_0, P_1, P_2 et P_3 donnent déjà une idée de la forme de la courbe dont ils sont une enveloppe (ceci n'est pas vrai pour un arc de cercle : son centre, le rayon et ses angles d'extrémités ne donnent pas une idée visuelle de la courbure).

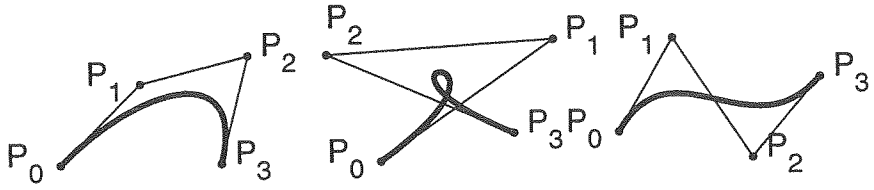


FIG. 3.10 - *Une courbe de Bézier est définie par 4 points P_i ; selon leurs positions relatives, on a diverses formes*

- Selon la position relative de ces points, on peut avoir des formes variées (figure 3.10). En déplaçant un seul point, on obtient des variations subtiles.
- Il existe une méthode simple de construction, dérivée du théorème de DE CASTELJOU [183]; elle est basée sur la décomposition de la courbe $P_0P_1P_2P_3$ en deux courbes $P_0P'_1P''_1M$ et $MP''_2P'_2P_3$ (voir figure 3.11) en calculant successivement les coordonnées des points

$$\begin{aligned} P'_1 &= (P_0 + P_1)/2 \\ A &= (P_1 + P_2)/2) \\ P'_2 &= (P_2 + P_3)/2) \\ P''_1 &= (P'_1 + A)/2 \\ P''_2 &= (A + P'_2)/2 \\ M &= (P''_1 + P''_2)/2 \end{aligned}$$

  et en continuant récursivement sur chacune des moitiés; basée sur des divisions par 2, cette méthode est très rapide, ce qui rend l'emploi des courbes de Bézier très efficace (voir ci-dessous 3.3.1).
- Ces courbes se raccordent facilement entre elles pour former des *splines*: elles ont les mêmes tangentes (continuité de la première dérivée) ce qui donne une impression de continu. Elles permettent de découper le contour d'un caractère en peu de morceaux (voir figure 3.12).
  Ce sont justement ces tangentes que proposent des produits commerciaux comme *Ikarus, Illustrator* ou *FontStudio*. Voir section 3.6 pour une description de ces utilitaires permettant d'entrer en ordinateur des contours de caractères et leurs propriétés typographiques.

Description d'un caractère

Voici (figure 3.13) comment est défini, en PostScript, le tracé des contours d'un « e » d'un *Times-Roman* pour un corps 1000. Ce n'est évidemment pas

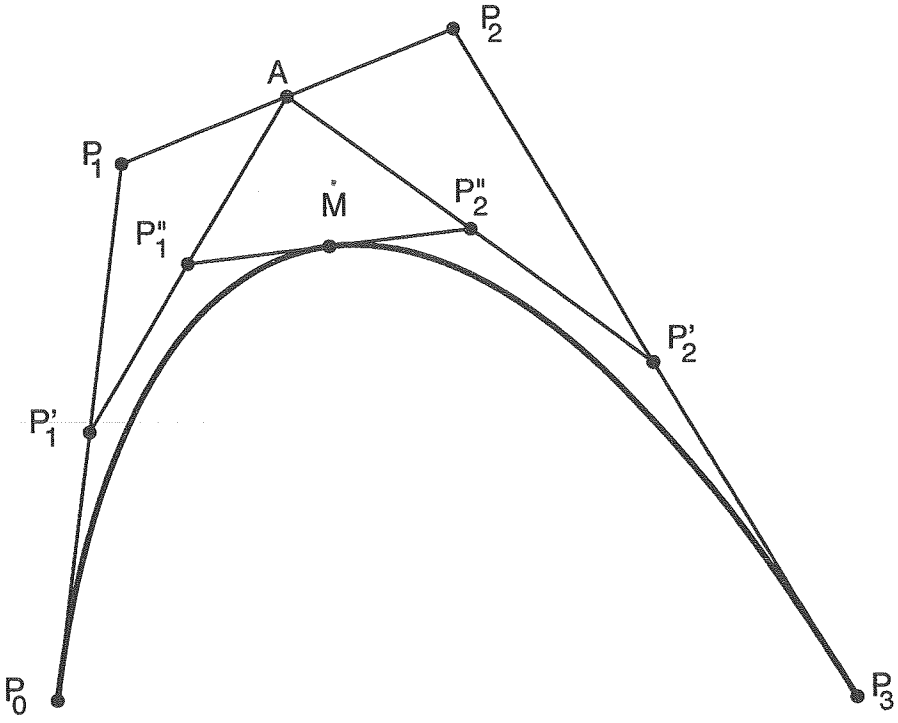


FIG. 3.11 - *Une courbe de Bézier se construit récursivement à partir de ses points de contrôle*

de cette façon qu'un caractère va être, en général, défini par un dessinateur typographe (voir section 3.6), mais c'est la forme finale interne des caractères.

La figure 3.12 montre, en grisé, le « e » correspondant et, en noir, les points de contrôle des courbes de Bézier associées.

3.3.1 Algorithme de conversion ponctuelle

Supposons, dans un premier temps (nous y reviendrons en 3.7), que les caractères soient tous identiques (proportionnellement), c'est-à-dire qu'un « e » en corps 10 soit identique à un « e » en corps 1000, mais 100 fois plus petit et que la définition de la grille n'ait pas d'influence sur le caractère définitif.

Le problème est alors : ayant la description d'un caractère par contours (comme celle correspondant à la figure 3.12), étant donné la grille physique de l'imprimante, déterminer quels en sont les points qu'il faut noircir (pour obtenir par exemple l'un des « e » de la figure 3.4). On emploie pour celà un programme de conversion ponctuelle (*scan conversion algorithm*) ce qui permet alors de convertir la surface (correspondant au caractère) en un ensemble de points à noircir. Ces algorithmes très complexes ont fait l'objet de beaucoup de travaux, mais relativement peu ont été publiés. On trouvera toutefois des descriptions de ces techniques dans les travaux de Roger HERSCH [109]. Voici quelques notions sur ces algorithmes.

Si on est capable de déterminer, pour chaque ligne de la grille physique, les points qui sont des frontières de changement de couleurs[4], alors un algorithme de *parity flag fill* permet de remplir correctement les pixels à noircir, y compris dans les cas aux limites comme le pixel unique en bas d'un V (figure 3.14).

Reste à déterminer ces *flags*, c'est-à-dire où passe la courbe sur la grille. Le principe est d'appliquer l'algorithme de De Casteljou (figure 3.11). Mais si cette méthode est simple en théorie, les conditions d'arrêt et de vitesse de convergence sont suffisamment compliquées pour avoir relevé longtemps pratiquement du secret commercial. Les points intermédiaires sont conservés dans une pile en mémoire-cache. Pour savoir quand arrêter la récursivité on emploie des méthodes basées sur la taille de l'enveloppe de chaque sous-courbe et la précision est choisie en fonction de la taille physique de la grille.

4. On choisit en général de mettre un pixel à l'intérieur ou à l'extérieur du bout de bande si le centre de ce pixel est lui même à l'intérieur ou à l'extérieur de la surface.

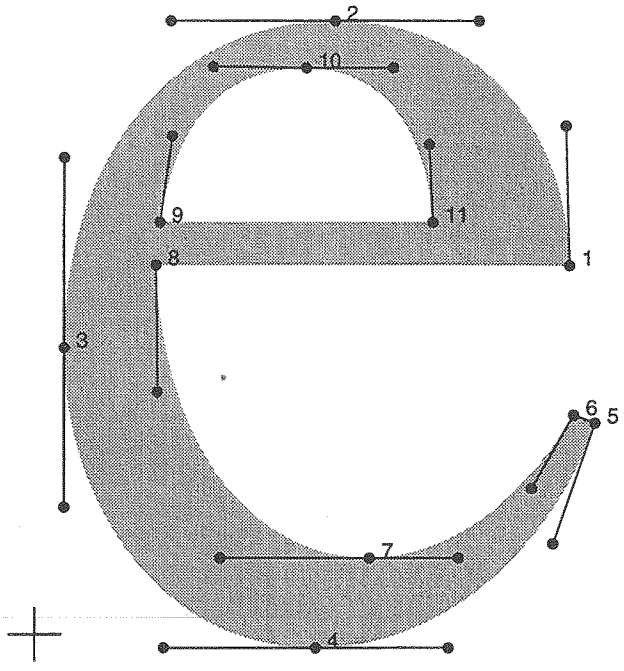


FIG. 3.12 - *Un « e » est défini par quelques points de contrôle et tangentes. Voir les figures 3.13 et 3.4*

```
\e{
402 276 moveto                    % départ point 1
399 380 334 458 226 458 curveto   % courbe de 1 à 2
102 458 22 356 22 214 curveto     % courbe de 2 à 3
22 95 97 -10 212 -10 curveto      % courbe de 3 à 4
312 -10 390,68 421 158 curveto    % courbe de 4 à 5
405 164 lineto                    % droite de 5 à 6
374 109 319 57 253 57 curveto     % courbe de 6 à 7
140 57 92 181 91 276 curveto      % courbe de 7 à 8
closepath                         % fin contour extérieur
94 308 moveto                     % départ au point 9
103 372 134 424 204 423 curveto   % courbe de 9 à 10
270 423 297 366 300 308 curveto   % courbe de 10 à 11
closepath} def                    % fin contour intérieur
```

FIG. 3.13 - *Programme PostScript donnant la définition du contour du « e » de la figure 3.12*

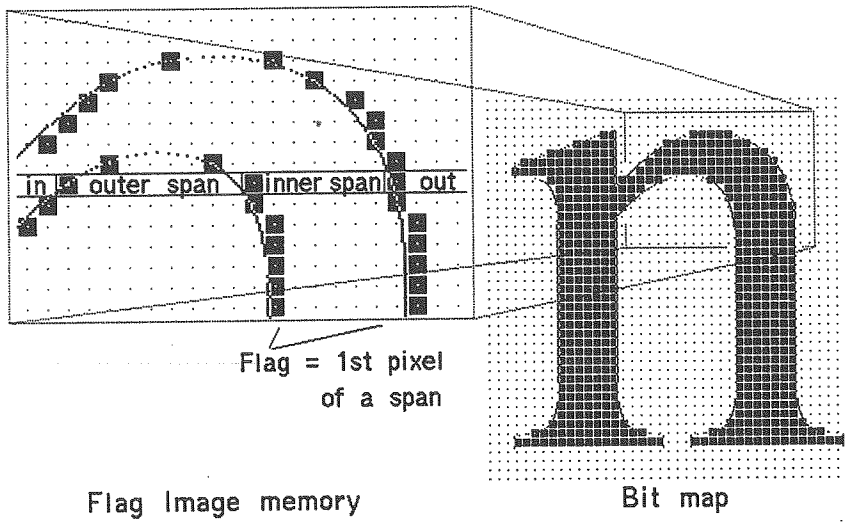


FIG. 3.14 - *Détermination des pixels à noircir à partir d'une image où sont marqués les bords de zones intérieures* (inner span) *et extérieures* (outer span) - *d'après R.* HERSCH *[109]*

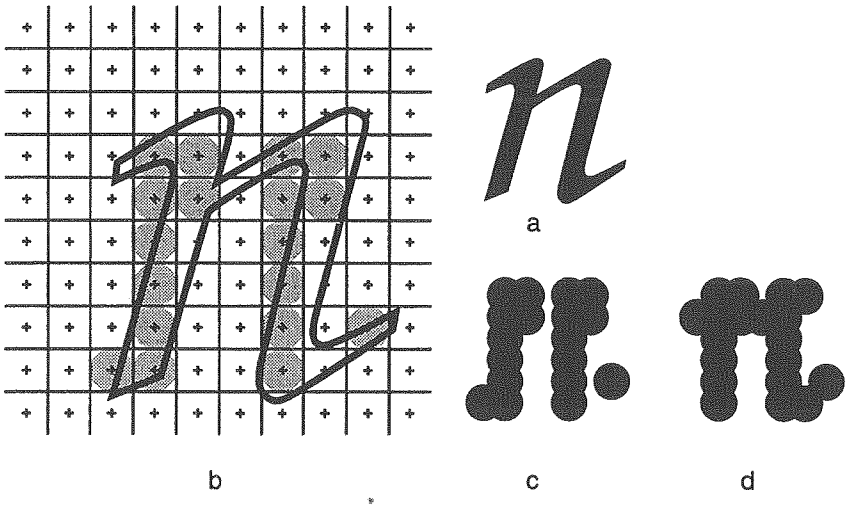


FIG. 3.15 - *Adaptation d'un très petit « n »* Zapf Chancellery *à la grille: a) le caractère désiré, b) la grille et la projection du contour théorique, c) le caractère obtenu sans correction (notez les trous) et d) après correction*

3.3.2 Adaptation des caractères à la grille

Avec une photocomposeuse à 4800 points par pouce et pour des corps usuels (par exemple un corps 12), les caractères ainsi tracés ont un très bon rendu (voir figure 3.4.c). Il n'en est pas de même pour de très petits corps (comme les corps 4 ou 5 des annuaires téléphoniques ou les corps 6 des contrats d'assurance) pour une photocomposeuse, ni pour des corps 10 sur une imprimante à laser (figure 3.4.a): dès que le nombre de pixels utilisés pour un caractère devient petit - par exemple lorsqu'un caractère a moins de 20 pixels de hauteur - les caractères risquent d'être dégradés.

Ces dégradations peuvent avoir des raisons diverses. En voici quelques-unes.

(1) Prenons un « n » qui devrait ressembler à celui de la figure 3.15.a. Compte tenu du corps désiré et de la résolution de cette grille, seuls vont être « allumés » les points marqués en gris (figure 3.15.b) ce qui donnera l'impression de la figure 3.15.c: le centre du pixel de l'arche du « n » n'étant pas à l'intérieur de la surface théorique reste blanc, ce qui donne un trou. De même à droite de la jambe de droite.

(2) Prenons maintenant le « n » de la figure 3.16.a. Cette fois, le contour théorique tombe de telle façon qu'un seul pixel se trouve (par ligne horizontale) à l'intérieur de la jambe de gauche alors qu'il y en aura

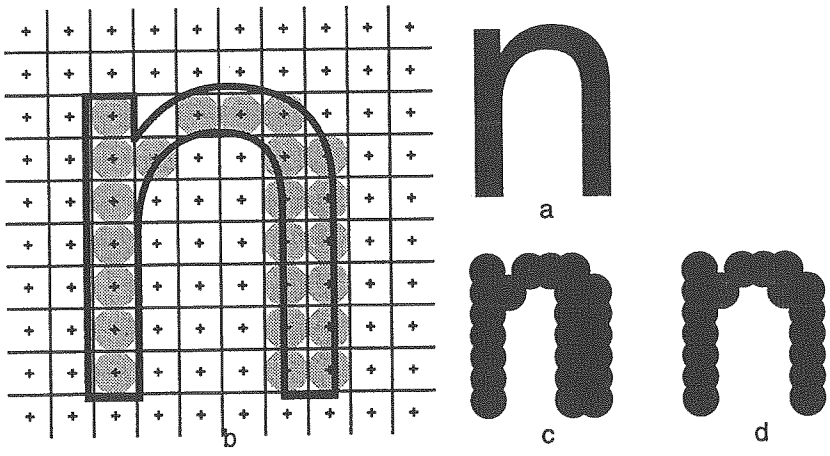


FIG. 3.16 - *Adaptation d'un très petit «n» Helvetica à la grille : a) le caractère désiré, b) la grille et la projection du contour théorique, c) le caractère obtenu sans correction (notez la différence d'épaisseur des deux jambes) et d) après correction*

deux dans celle de droite (figure 3.16.b). Il y aura désquilibre, le «n» ayant deux jambes inégales.

Comment corriger ces erreurs? En utilisant une «connaissance» de la topologie du caractère. On voit bien (figure 3.16-b) qu'il suffit de déplacer très légèrement la jambe de droite pourqu'elle ait la même graisse que celle de gauche (figure 3.16-d) ; de même en remontant très légèrement l'arche du «n» de la figure 3.15-b on obtient le «n» sans trou de la figure 3.15-d. C'est cette connaissance topologique qui est donnée par des *hints* (indications). Il existe plusieurs façons de donner ces indications - ce sont les standards *TrueType* [90], *Type1* [92], *URW* [111], etc. (on trouvera dans le rapport Seybold [131] et chez HERSCH [109, 137] quelques précisions et comparaisons). En fait, on trouve soit des méthodes déclaratives (par exemple *Type 1*), soit des méthodes impératives (comme *TrueType* et METAFONT dont il est issu). Cette dernière méthode considère qu'une fonte est un programme et que l'adaptation à la grille relève de ce programme, les instructions étant des instructions de déplacement des courbes avec toutes les instructions nécessaires de comparaison d'épaisseur de fûts, etc. Dans *TrueType* on a par exemple les instructions

```
17> MDRP[abcde]
23> ALIGNRP[]
33> SHPIX[]
```

qui s'appliquent aux points de contrôle numéro 17, 23 et 33 respectivement pour les déplacer *(MDRP=Move Direct Relative Point)* de façon à contrôler l'épaisseur ou la chasse d'une glyphe, aligner un point relativement à un autre ou déplacer un point selon un vecteur *(SHift point by a PIXel amount)*. De ce fait, il est très difficile pour un « artiste » de toucher à ces commandes.

En revanche, les commandes déclaratives sont séparées de la description du contour et viennent, en quelque sorte, en plus de la description des contours. Un système de création de fonte peut donc les inclure séparément dans la fonte à partir de commandes données par un typographe à l'aide de la souris. Ces commande sont du type

- pas de trou à tel endroit,
- la largeur ici doit être la même que là.

Elles concernent non plus les déformations des courbes mais de la grille elle-même. C'est pourquoi ces commandes sont liées aux *lignes bleues* qui servent de cadre à un caractère. Par exemple

```
/BlueValues[-16 0 320 340 ...] ND
```

précise que la ligne de correction optique des bas de casse est en $y = -16$, que la ligne de base est en $y = 0$, que la ligne des minuscules est en $y = 320$ que celle de correction optique des minuscules est en $y = 340$, etc. Pour le « e » de la figure 3.17, on définira des lignes verticales *vstem* ou horizontales *hstem* avec une largeur définissant des zones sans chevauchement (c'est-à-dire des épaisseurs à respecter), par exemple

```
20  60 vstem
250 60 vstem
```

Il s'agit bien sûr d'indications propres à un caractère donné pour une police donnée et non quelque chose de général (du style « tous les « n » ont leurs jambes égales ») : c'est donc un typographe qui doit les donner, caractère par caractère, tout comme, autrefois, seul un typographe corrigeait un par un les pixels d'un plan de bits.

De nombreux autres cas doivent ainsi être indiqués, par exemple pour éviter que deux capitales n'aient pas la même hauteur, pour qu'un « O » ne soit ni pointu ni aplati, pour traiter les diagonales ou les hampes d'italiques, etc.

Ce problème de l'adaptation des points à la grille est en fait un problème de conservation de propriétés géométriques, notamment des éléments de structure horizontaux et verticaux.

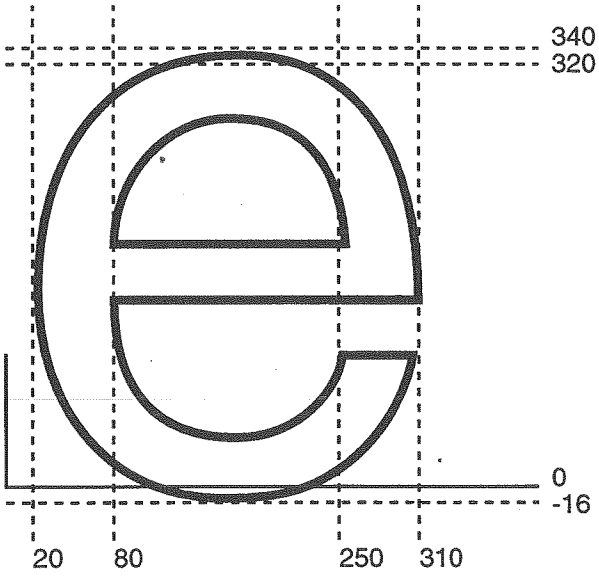


FIG. 3.17 - *Indications* (hints) *à l'aide de « lignes bleues » (horizontales et verticales) pour garantir la conservation des propriétés géométrique du caractère*

```
consultation de l'état graphique
prise des informations globales sur la fonte

si le plan de bits de (a) n'a pas été dejà calculé
alors début
        appeler l'algorithme de scan conversion pour (a)
        appliquer les calculs de hinting
        construire le plan de bits
        le sauver en mémoire cache
      fin
prendre le plan de bits en mémoire cache
le copier dans la page au point courant
mettre à jour le point courant
```

FIG. 3.18 - *Machinerie employée par l'instruction* (a) show *en PostScript*

Signalons que des « modèles topologiques de fontes » sont à l'étude, comme ceux de BETRISEY [125] [137]. Mais, il ne semble pas que, jusqu'à présent, les techniques employées en *cognition* aient été employées pour décrire les connaissances topologiques et les règles typographiques. Nous y reviendrons au chapitre 7.

3.4 Calcul des caractères PostScript

Voyons maintenant comment ces divers algorithmes fonctionnent en PostScript.

3.4.1 Machinerie des fontes

Les algorithmes de *scan conversion* et de *hinting* sont effectivement appelés pour chaque caractère à imprimer. Mais comme ce sont des algorithmes très coûteux en temps, PostScript a prévu un mécanisme de sauvetage des plans de bits: on ne calcule le plan de bits d'un caractère donné (par exemple un « a » de *Times-Roman* en corps 12 orienté à 30 degrés compte tenu de la grille de telle définition, etc.) que si ce plan de bit n'est pas déjà en mémoire « cache » du programme en cours. S'il y est, on se contente de le recopier.

L'instruction (a) show provoque donc l'exécution du programme de la figure 3.18.

Le gros avantage de cette machinerie est que même si elle est coûteuse en temps, elle évite d'avoir à précalculer tous les plans de bits possibles

et imaginables en fonction de la taille, de l'orientation des caractères et en fonction de la façon dont chaque imprimante est physiquement constituée.

3.4.2 Métrique des fontes

Les caractères en plomb avaient une largeur (appelée « chasse ») dépendant du caractère (un « m » est plus large qu'un « i ») et une hauteur (appelée « corps »). Voir figure 3.19, gauche. De part et d'autre de l'« œil » (la surface imprimable), se trouvaient des espaces de façon que les caractères imprimés ne se touchent pas, ni dans une même ligne (« approches » droite et gauche), ni d'une ligne à l'autre (« talus » de tête et de pied).

Ces notions « physiques », liées aux trois dimensions du caractère, n'ont plus de raison d'être maintenant. Mais, pour sauver le plan de bits d'un caractère, l'interpréteur PostScript a besoin de connaître la surface occupée : il demande alors qu'à chaque caractère soient associées les coordonnées du plus petit rectangle exinscrit à ce caractère. C'est ce que l'on appelle la *bounding box* ou *bbox* (figure 3.19).

Une autre information indispensable à PostScript est la valeur de la chasse du caractère, c'est-à-dire de combien il faut avancer la valeur du point courant pour placer le caractère suivant sans qu'il touche au premier. Nous avons donné dans [26] [91] tous les détails et les raisons de ces « métriques » qui sortent du cadre du présent texte. Disons simplement ici que toutes ces informations sont regroupées dans un fichier, appelé AFM chez Adobe (pour *Adobe Font Metric*), et que ce fichier est accessible par les formateurs (comme Word, FrameMaker ou LaTeX) qui doivent disposer de ces valeurs métriques pour faire tous les calculs de composition (notamment de justification et de division des mots) et de mise en page.

Enfin, PostScript utilise un mécanisme de codage *(encoding scheme)* des caractères : pour PostScript chaque caractère a un nom (par exemple *A* pour « A », *eacute* pour « é », *colon* pour « : »). Une table permet alors de passer des codes d'entrée (par exemple codes ASCII, ISO-LATIN1, EBCDIC, etc.) au caractère voulu (voir [198]). Ces tables de codage, assurant la portabilité, font aussi partie de la fonte.

3.4.3 Unités typographiques

Les typographes utilisent le « point typographique » comme unité de mesure. Le problème est qu'actuellement il y a au moins 4 définitions différentes de ce point. Avant de comparer ces valeurs, un peu d'histoire (on trouvera plus de détails dans les articles fondamentaux de MOSLEY [44] et TRACY [59]).

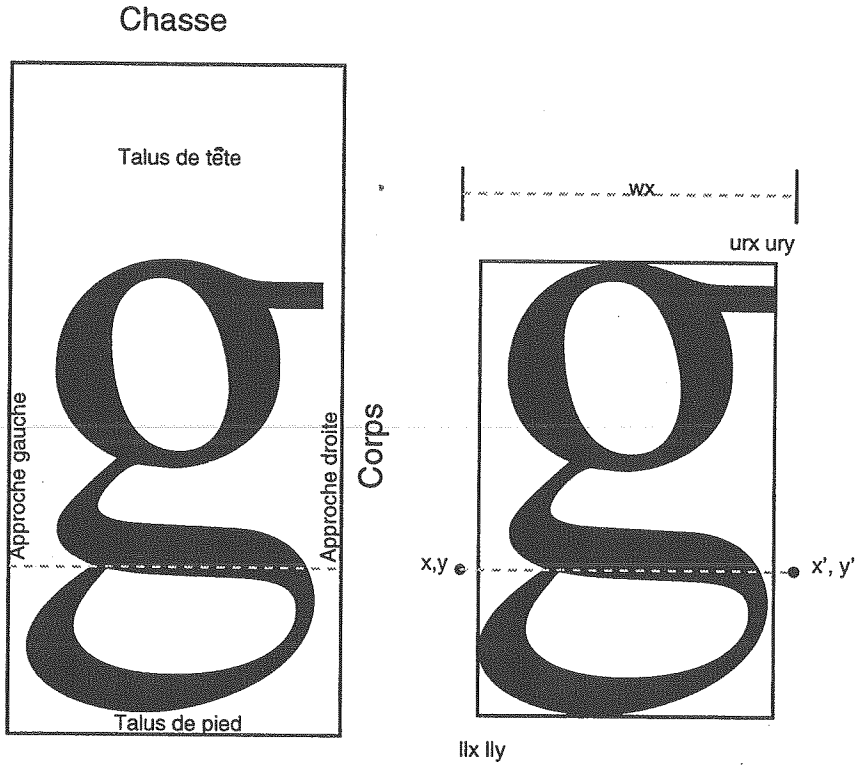


FIG. 3.19 - *Comparaison de la métrique d'un caractère en plomb (à gauche) avec celle d'un caractère en PostScript (à droite)*

TAB. 3.1 - *Les quatre points typographiques*

1 point didot	1/72 pouce français	0,3759 mm
1 point millimétrique		0,4000 mm
1 point pica	1/72,27 pouce	0,3515 mm
1 point DTP	1/72 pouce	0,3528 mm

Pendant longtemps, on a donné des noms[5] aux tailles de caractères. Ces noms étaient, en général, basés sur des titres de livres ayant été composés dans ce corps. Ils étaient souvent charmants (*nompareille, mignone, gaillarde, triple canon,* etc.), mais ne correspondaient pas toujours à la même taille; aussi, au début du XVIII^e siècle, on commença à essayer de normaliser ces valeurs. FOURNIER (voir son *Manuel* [10] et la récente étude de James MOSLEY [44]) proposa une première série de mesures, mais c'est François Ambroise DIDOT qui proposa, en 1783, la valeur encore en usage aujourd'hui sous le nom de « didot » : un point didot valait alors un sixième de la ligne de pied du roi, c'est-à-dire à 1/72 de pouce (français). Le point didot vaut aujourd'hui 0,3759 mm. Son multiple est le « douze » et comme il correspondait à la taille de caractère qui s'appelait *Cicero*, le mot « cicero » désigne encore ce multiple.

Peut-être parce que cette unité était trop récente, le point didot n'a pas été intégré dans le ystème métrique lors de sa création en 1801. Toutefois, l'Imprimerie royale, devenue nationale, a défini un « point millimétrique » , de 0,40 mm, qu'elle seule semble avoir jamais utilisé.

Le point didot a été adopté pratiquement partout en Europe et dans le monde, sauf en Grande Bretagne, aux USA et dans les pays anglophones, où le vieux système de noms prévalait (et prévaut encore parfois !). En 1866, l'association des fondeurs américains, se décida à adopter le système français, 1/72 de pouce, mais comme le pouce anglais était différent du pouce français de 1783, on obtint une unité légèrement différente: le *pica* (dont le nom vient d'un caractère équivalent au *Cicero* appelé ainsi à cause de sa « couleur » : *pica* signifie « la pie » en latin !). Le développement des matériels américains a bien sûr fait que ce point pica a commencé à supplanter le point didot.

Vers 1954, la valeur du pouce américain a été arrondie. Mais on a gardé la vieille taille du point pica, c'est-à-dire que sa définition est passée de 1/72 de vieux pouce à 1/72,27 de nouveau pouce...

Ce rapport 1/72,27 étant souvent difficile à calculer, les informaticiens ont pris, c'est notamment le cas de PostScript, l'habitude de l'arrondir à

5. On a aussi fait la même chose pour les papiers, par exemple *jesus*, et on fait encore la même chose en cette toute fin du XX^e siècle lorsque l'on parle de caractère « gras » ou « extra-maigre » !

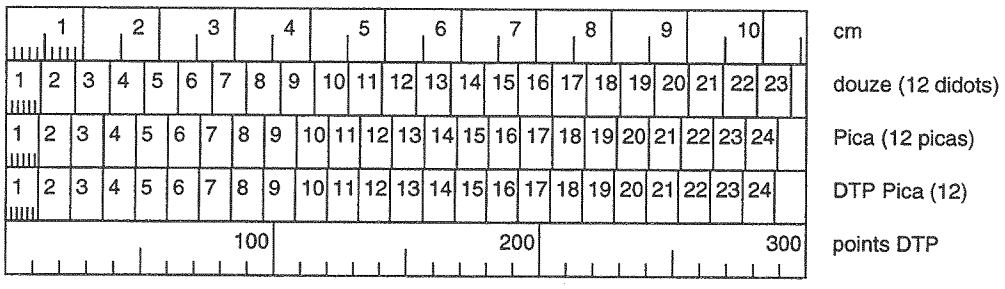


FIG. 3.20 - *Comparaison des principaux points typographiques (d'après [91])*

1/72. D'où un nouveau point pica, que l'on appelle parfois DTP *(DeskTop Point).*

En résumé, on a aujourd'hui 4 points typographiques (table 3.1).

Ces différences ne sont pas grandes, mais peuvent ne pas être négligeables quand on cumule ces valeurs (voir figure 3.20) ou, comme nous l'avons fait remarquer dans [91], pour placer des signes diacritiques.

3.5 Fonte numérique

FINALEMENT, une fonte PostScript (mais c'est pratiquement la même chose pour les autres formats) est une structure informatique complexe comprenant de nombreux champs ou « formats » (voir [91] et [134]) :

(1) pour l'interpréteur PostScript lui-même :

- des informations globales sur la fonte (hauteur des capitales, des descendantes, etc.),
- pour chaque caractère, un algorithme de dessin,
- pour chaque caractère, des indications sur ses propriétés topologiques *(hints),*
- des tables de codage,
- des tables de métrique ;

(2) pour les formateurs en amont :

- la métrique des caractères,
- des plans de bits précalculés (par exemple pour affichage sur écran dans les systèmes Wysiwyg).

Ce sont ces divers fichiers que l'on achète sous le nom de « fonte » (et non de « police » - voir section 3.9). Leur gestion relève du génie logiciel (mise à jour des versions, adaptation d'une fonte à divers matériels, etc.) et on commence seulement à prendre conscience de notions comme celle de « serveur de fonte », avec comme corollaire la nécessité de trouver des algorithmes pour remplacer une fonte manquante par une autre compatible (au niveau de la métrique notamment).

3.6 Saisie des fontes

DANS CETTE section, nous montrons comment sont numérisés aujourd'hui les caractères, c'est-à-dire comment trouver leurs courbes de Bézier ou comment donner des indications pour mieux les adapter à la grille spécifique où ils seront tracés.

3.6.1 Utilisation de tablette à numériser ou de scanner

C'est aujourd'hui la méthode la plus fréquente et en tout cas celle employée par les fondeurs d'aujourd'hui.

(1) Le caractère à numériser est dessiné à la main (ou reproduit photographiquement et retouché), avec une grande taille (par exemple sur une feuille A4, en gros un corps 1000).

(2) Un opérateur marque, à l'aide d'une tablette à numériser, les points des contours du caractère qui lui paraissent être les points importants (points d'inflexion, angles, etc.).

(3) Le système en déduit les points de contrôle des courbes de Bézier ou les extrémités de lignes.

(4) En déplaçant ces points à l'aide d'une souris, l'opérateur peut alors corriger le tracé final.

Le plus connu de ces produits est *Ikarus* de Peter KAROW [111]. Une variante est de ne plus utiliser de tablette mais directement un scanner et d'utiliser des procédures d'*autotracing* [133].

3.6.2 Création directe sur écran

Des produits commerciaux comme *FontoGrapher* ou *FontStudio* permettent, en cliquant avec une souris, de tracer directement des courbes sur un écran. De nombreuses procédures permettent ensuite, comme Ikarus, de modifier ou corriger ces courbes. Mais il est quasi impossible de définir

une fonte de la sorte, ces outils servant plutôt à modifier des fontes du commerce à des fins graphiques. Voir cependant le travail de Hans MEIER en Suisse [149].

3.6.3 Programmation

Une méthode beaucoup moins employée est de programmer directement en PostScript (c'est ce que nous avons fait avec le *Delorme*, voir section 6.2). Mais c'est aussi la façon dont on se sert de METAFONT, défini par D. KNUTH [144].

3.7 Recherches actuelles en typographie numérique

LA TYPOGRAPHIE numérique, nous venons de le voir, est suffisamment au point pour traiter les cas normaux et les produits sur le marché ont une qualité équivalente, sinon supérieure, à celle de la photocomposition. Mais, comme nous l'avons dit dès le chapitre 1, la photocomposition n'avait pas non plus la qualité du plomb. Les recherches actuelles en typographie numérique sont donc orientées sur la qualité des caractères non plus pris séparément (comment obtenir un bon rendu), mais dans leurs rapports entre eux: d'une part, lorsqu'ils varient d'une taille à l'autre (ce que nous appelons « ajustement optique » dans le chapitre 4) et, d'autre part, lorsqu'ils sont côte à côte (problèmes d'espacement).

Mais ces « règles » traditionnelles (relevant du « code typographique ») sont basées sur une connaissance du visuel typographique. Dès 1764, Fournier donnait dans son *Manuel typographique* [10, page 8] des instructions comme « *Il y a une chose essentielle à observer dans la taille du Calibre; c'est d'ouvrir un peu moins ... pour l'italique que pour le romain. Si ce Calibre étoit de la même grandeur, l'italique paroîtroit, à l'impression, plus grand que le romain ...* ». Depuis deux siècles, de nombreuses règles ont été émises, mais si un typographe sait dire « c'est trop grand ou c'est trop serré », il ne saura pas en général dire de combien ni pourquoi. Or les programmes de calcul de caractères doivent pouvoir quantifier ces variations. D'où l'intérêt de suivre les études sur la lisibilité.

3.7.1 Études sur la lisibilité des caractères

Depuis les travaux de JAVAL[6], de très nombreuses recherches ont été faites sur la physiologie de la lecture et sur la lisibilité des caractères. On

6. Émile JAVAL (1839 -1907) est le physiologue français qui a découvert les saccades de l'œil lors de la lecture; il est aussi connu pour ses études sur la lisibilité des lettres. Voir [7, page 321 sqq.] et [190].

trouvera dans THINKER [197] et, en français, dans divers livres édités par RICHAUDEAU [7, art. le processus de lecture et lisibilité] [196], un aperçu du très grand nombre de publications sur ce sujet.

Depuis quelques années, deux nouvelles voies de recherche en matière de lisibilité apparaissent, conjointement d'ailleurs :

(1) d'une part, celle basée sur la physiologie mais aussi sur la *cognition* et où l'on différencie bien les diverses étapes de la lecture (réception des images, intégration spatio-temporelle, reconnaissance des éléments, reconnaissance des lettres puis des mots avec sa propre culture, analyse du mouvement des yeux, etc.) ;

(2) d'autre part, celle basée d'avantage sur la recherche d'un modèle mathématique (transformées de Fourier et théories de la vision humaine).

Les travaux de LEGGE [192], MORRIS [193], [194] et, en France, de O'REGAN [195] sont les plus connus.

Mais il ne semble pas que ces travaux débouchent, aujourd'hui, sur des résultats « définitivement utilisables » en matière de règles de lisibilité de caractères ... Ainsi, par exemple, trouve-t-on des publications où l'on « prouve » que les capitales sont moins lisibles que les minuscules. Mais d'autres publications prouvent le contraire. Bien sûr, il ne s'agit pas du même contexte. Mais ça montre quand même qu'il n'y a probablement pas de règles absolues de ce type.

Néanmoins, si l'on veut procéder à des espacements automatiques de lettres (voir ci-dessous section 4.3), il faudra bien s'appuyer sur des considérations non seulement esthétiques mais aussi scientifiques et on est en droit d'espérer que ces résultats vont converger.

3.7.2 Des fontes statiques aux fontes analytiques

Le chapitre 4 sera consacré aux méthodes employées aujourd'hui pour satisfaire les besoins, empiriques, en matière de lisibilité. Disons dès à présent que le principe est de ne plus définir un contour en donnant des valeurs numériques (comme « `402 276 moveto` » en figure 3.13) pour les coordonnées des points de contrôle, mais des fonctions analytiques (par exemple remplacer `402` et `276` par des fonctions f et g dont la valeur est calculée en fonctions de divers paramètres, par exemple le corps du caractère).

En corollaire de ceci, de nouvelles recherches sont en cours pour considérer les « fontes » à un niveau encore plus élevé et notamment pour trouver une modélisation de fontes (voir chapitre 7).

FIG. 3.21 - *Caractères à niveaux de gris produits par Rastware [124]*

3.7.3 Caractère gris

Ces trois domaines de recherche (études sur la lisibilité, ajustement optique et calcul automatique des espaces) sont intimement liés à un autre domaine: l'étude de nouveaux caractères pour écrans utilisant les techniques de niveaux de gris. Nous n'en dirons ici que deux mots.

Pour une imprimante, un pixel n'a que deux couleurs: ou bien il est noir, ou bien il est blanc. Pour les écrans, ces valeurs noir/blanc sont en fait produites par un équilibrage des trois couleurs RVB et en jouant sur leurs rapports on peut avoir, pour certains écrans, des pixels plus ou moins gris. Selon l'échelle, on a plusieurs « niveaux de gris ».

Ces caractères sont voués à un très grand développement. En effet, ils permettent une bien meilleure lecture sur écran que les caractères traditionnels, ce qui est fondamental pour les développement des systèmes multimédia. La notion même de contour est remise en cause par certains, comme Richard SOUTHALL [57] qui pense que l'on va ainsi revenir à des techniques de spécification de l'apparence visuelle comme du temps des graveurs de poinçons (voir chapitre 7). Déjà, des typographes comme André GÜRTLER [138] s'y mettent très sérieusement et des produits commerciaux (FontStudio, PhotoShop utilisé avec un moniteur Mac Gray) permettent de créer des caractères à niveaux de gris.

Les lectures de base sur le sujet restent le livre de RUBINSTEIN [115, pages 111–115] et la thèse d'Avi NAIMAN [151].

3.8 À propos de fontes

Terminons ce survol de la typographie numérique par deux remarques.

3.8.1 Fonte ou police?

Ces deux expressions, police et fonte, ont été relativement malmenées ces temps derniers. Nous proposons de faire le point à ce sujet.

Du temps du plomb, une fonte correspondait à l'ensemble des caractères d'un « caractère » (par exemple le *Garamond*). Ça se vendait à la tonne, et on livrait avec une liste des caractères présents, par exemple « 1000 a romain en corps 12, 800 b romain en corps 12, etc. ». Cette liste s'appelait une police (ce mot vient de l'italien, *polizia* = liste ; c'est le mot que l'on retrouve dans « police d'assurance »). Lors de l'avènement de la Lumitype puis de la photocomposition, on a appelé « police » non plus la liste des caractères, mais les caractères eux-mêmes, leur support matériel, matrice ou disque[7]. Mais le *Times*, par exemple, était formé de plusieurs polices, non seulement pour différencier le gras, l'italique, etc., mais aussi pour avoir plusieurs games de corps (par exemple pour les petits corps, pour les moyens et pour les grands).

Lorsque les fontes numérisées sont apparues sur le marché des systèmes de traitement de textes, les commerciaux ne se sont pas fatigués et ont traduit *font* par *fonte* alors que, à l'époque, le mot police convenait mieux.

Mais avec la notion de *font* telle qu'elle se trouve en PostScript, voire en METAFONT, il faut distinguer deux choses :

- ce que l'on achète,
- ce que l'on utilise à un moment donné.

Nous proposons donc d'appeler

fonte ce que l'on achète et qui permet de créer un certain nombre de polices (concept important avec les Multiple Masters)

police ce que l'on utilise à un moment donné, c'est-à-dire le résultat de l'une des deux instructions PostScript

```
\FONT findfont NN scalefont setfont
\FONT findfont [ . . . . . .] makefont setfont
```

En particulier, nous associons à « fonte » tout ce qui est lié à la métrique des caractères, à leur topologie, aux *hints*, etc. Tandis que « police » correspond au seul aspect « visuel ».

C'est à peu près la même différence qu'entre « glyphe » et « caractère imprimé » (le premier étant l'entité théorique et le second la réalisation, ou l'instanciation, de cette entité - voir par exemple l'article de CH. BIGELOW sur Unicode [127]).

7. On a donc ici un bel exemple de glissement de sens, équivalent à celui du mot « bureau » qui a d'abord désigné une pièce d'étoffe, puis la table qui la portait, puis la pièce où se trouvait la table, puis l'immeuble, puis l'entité abstraite, etc.

3.8.2 Typographie numérique et images 2D

A priori, les caractères ne sont que des images 2D (dans le plan à deux dimensions). Effectivement, nombre d'algorithmes sont communs à ces deux disciplines. Mais la typographie numérique attache plus d'importance que l'imagerie 2D classique à un certains nombre de spécificités :

- les caractères sont en général de « petits » dessins, ce qui a obligé toute une série de recherches sur les problèmes d'adaptation à la grille ;
- la notion de fonte est indissociable de celle de famille, d'ensemble : une fonte doit gérer globalement certaines propriétés ; c'est d'ailleurs là, nous y reviendrons en conclusion, où les recherches sont les plus importantes ces temps-ci ;
- les caractères doivent être « lus », ce qui impose le respect d'un certain nombre de règles, même si elles ne sont pas encore bien formalisées aujourd'hui.

Toutes les recherches actuelles visent d'ailleurs à étudier ce qui fait qu'un caractère n'est pas qu'une simple image 2D noircie. La typographie numérique devient vraiment une discipline informatique en utilisant des techniques d'intelligence artificielle, de langages, d'objets, etc. (voir conclusion).

Fontes analytiques

DANS CE CHAPITRE, nous montrons que l'emploi de variables analytiques dans la description des contours des caractères permet de résoudre beaucoup des critiques faites aux caractères informatiques.

4.1 Introduction à la notion de fonte analytique

4.1.1 Problème

POUR BIEN situer les problèmes, prenons l'exemple de la figure 4.1. On y trouve plusieurs fois le même mot, avec des caractères originalement de corps différents (de 6 points à 40 points) mais agrandis photographiquement à la même taille[1].

Contrairement à ce que le non-typographe pense et contrairement à ce qui était fait dans les premiers systèmes (voir ci-dessus section 3.3.1), un caractère en corps 100 n'est pas identique au même caractère en corps 50 à une homothétie près. On voit dans cette figure 4.1, par exemple, que plus les caractères sont de petite taille, plus leurs traits sont épais (sinon ils seraient peu visibles) et plus leurs contre-poinçons (l'espace blanc dans la boucle du « e » par exemple) sont grands (sinon l'encre risquerait de les boucher).

Ne pas utiliser de fonctions linéaires globalement sur un caractère (comme le zoom des photocomposeuses) correspond à ce que les anglo-saxons appellent les règles d'*optical scaling* et que nous appelons « ajustement optique ». La figure 4.6 montre des « e » Garamond en divers corps, sans ajustement optique (en haut) et avec (en bas).

Nous nous trouvons face à deux problèmes :

(1) comment faire varier informatiquement le dessin d'un caractère en fonction du corps?

(2) quelles lois suivent ces variations?

1. On entend ici par « taille » la hauteur d'un caractère, plus exactement de la trace imprimée d'un caractère. Cette notion est liée à celle de corps ou mieux de *scaling factor* de PostScript, d'autant que nous l'utilisons parfois, justement, dans un repère qui n'est pas forcément le repère normal – la figure 4.6, par exemple, présente le même caractère dans des corps différents mais ramenés à la même taille.

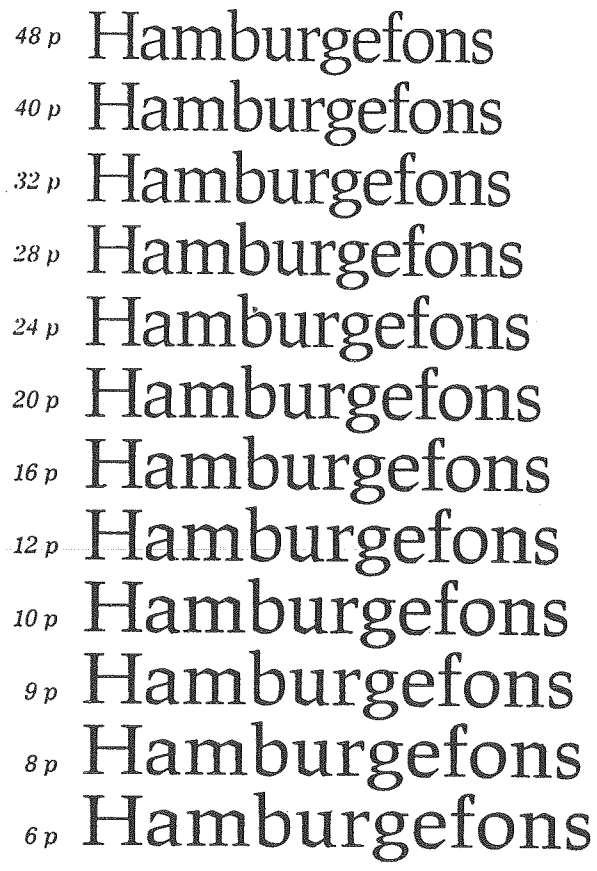


FIG. 4.1 - *Ajustement optique : le mot* Hamburgefons *a été composé (en* Leipziger Antiqua *par Peter* KAROW *[141]) dans divers corps et tous agrandis à la même taille. Plus le corps est petit, plus la lettre est, proportionnellement, large et aérée mais a des traits plus épais*

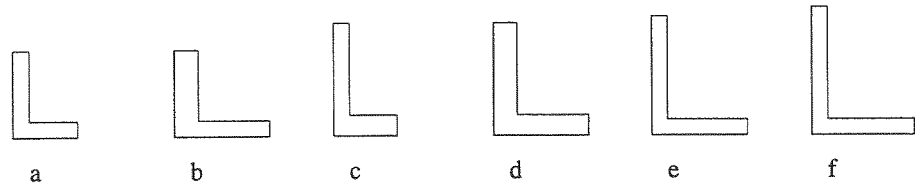


FIG. 4.2 - *Déformations d'un «L»*

```
...
/L{% description du caractère "L"
     0  270 moveto
     0    0 lineto
   200    0 lineto
   200   50 lineto
    50   50 lineto
    50  270 lineto
  closepath stroke } def
...
/BuildChar {
    ...
      wx wy llx lly urx ury setcachedevice
    ...
   } def
```

FIG. 4.3 - *Extrait de la définition d'une fonte*

Voyons d'abord le premier point.

4.1.2 Exemple simple

Considérons une fonte décrivant des caractères très simplistes (figure 4.3).

Supposons que nous voulions, à partir du «L» de la figure 4.2.a passer à la figure 4.2.e, c'est-à-dire allonger la barre horizontale de 50% et celle verticale de 30% sans pour autant les épaissir. Les instructions consistant à faire des homothéties

```
... 1.5 1 scale (L) show   % figure b
... 1 1.3 scale (L) show   % figure c
... 1.5 1.3 scale (L) show % figure d
```

donneraient respectivement les figures 4.2.b à 4.2.d.

Aucune n'est la bonne solution car la graisse est modifiée : l'épaisseur de la barre verticale est aussi augmentée de 50% en b et d, tandis que celle de la barre horizontale l'est de 30% en c et d. Il faut donc utiliser des variables analytiques, soit en pseudo-PostScript :

```
/L{% description analytique de "L"
     0   270+dy moveto
     0      0    lineto
   200+dx   0    lineto
   200+dx  50    lineto
    50     50    lineto
    50    270+dy lineto
  closepath stroke } def
```

et appeler ce caractère comme suit :

```
/dx 100 def % 200x1.5-200
/dy  90 def % 270x1.3-250
(L) show    % figure e
```

Il est possible depuis quelques années déjà, notamment avec METAFONT, de paramétrer des caractères. D'ailleurs depuis près de 20 ans, les fontes *computer modern*, redessinées par D. KNUTH [143] pour TEX, utilisent un dessin variant avec le corps et, ce qui est encore rare dans des fontes commerciales, offrent de vraies petites capitales. C'est la méthode employée aussi, par exemple, pour dessiner des ligatures arabes [135] ou des fontes aléatoires comme *Punk* de KNUTH [145]. Mais, comme nous l'avons dit au chapitre précédent, les plans de bits sont calculés avant d'être chargés dans l'imprimante. Les instructions suivantes ne donneraient pas, avec une telle fonte, la figure f mais redonneraient la figure e :

```
/dx 120 def
/dy 130 def
(L) show % on attend la figure f mais on a encore e
```

Il en est rigoureusement de même avec PostScript : on a la même chose que lors du premier appel ; en effet le mécanisme de cache de PostScript (section 3.4.1) fait que, au premier appel e, le plan de bits de ce « L » est calculé avec les valeurs présentes dans *dx* et *dy* (donc, ici, 100 et 90). Au second appel, on se ressert directement de ce plan de bit, quelles que soient les nouvelles valeurs de *dx* et *dy*.

Nous verrons au chapitre 5 que PostScript permet de désactiver ce mécanisme de cache et qu'on pourra alors avoir la figure f après la figure e dans une même page. Ceci nous permettra de définir la notion de caractère dynamique.

FIG. 4.4 - *Les deux caractères* Didot *de gauche sont en corps 100. Ceux de droite sont des corps 1 agrandis 100 fois*

4.2 Ajustement optique

4.2.1 Modification de la forme d'un caractère

Nous avons dit (section 3.4.1) que, lors de l'exécution du programme de show, on commençait par appeler le dictionnaire de la fonte courante. Celui-ci contient la force du corps courant. Rien n'interdit donc de paramétrer les instructions donnant le contour en fonction du corps actuel (nous donnons dans [161] la façon pratique de faire ceci en PostScript, tant pour les fontes de type 1 que celles de type 3). Nous venons de voir que l'emploi de fonctions analytiques est possible. Rappelons que, dans ce cas, le plan de bits est sauvé et réutilisé pour chaque caractère d'un corps donné : il n'y a pas, contrairement aux fontes dynamiques, de recalcul du caractère.

Prenons un exemple simple : le *Didot* est un caractère dessiné par Firmin DIDOT au début du XIX^{e} siècle et qui a la particularité d'être formé de filets très fins. Il fallait donc que les poinçons soient adaptés de façon que les petits corps soient quand même imprimables : proportionnellement, ils devaient avoir des filets beaucoup plus épais (figure 4.4).

Nous avons montré [161] que ceci pouvait s'obtenir en utilisant, pour la taille du filet, la fonction suivante

$$\text{épaisseur du filet} = 10+20 \times \log(100/\text{corps})$$

4.2.2 Comment varient les caractères ?

Notre solution pour le *Didot* est basée sur le fait qu'il fallait trouver une fonction logarithmique, les coefficients ayant été choisis expérimentalement. De même peut-on, par déplacement des points A_1, A_2, B_1 et B_2, augmenter le contre-poinçon du « e » de la figure 4.5.

Le gros problème, alors, est de savoir comment *doivent* varier ces caractères.

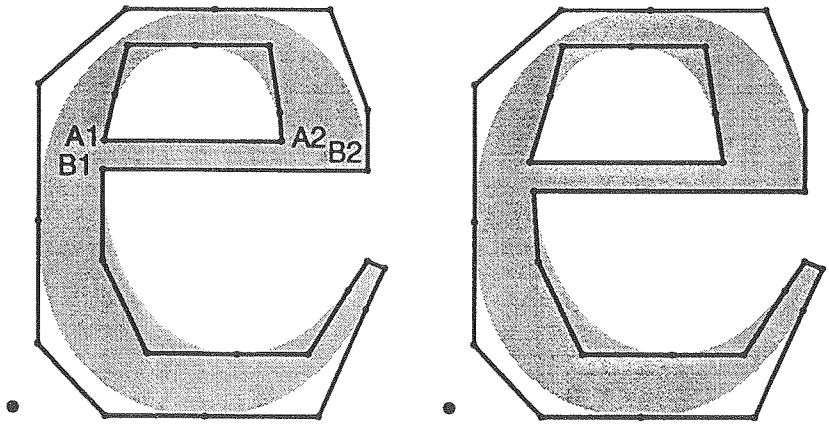


FIG. 4.5 - *Par déplacement des points de contrôle A_1, A_2, B_1 et B_2, on peut augmenter le contre-poinçon d'un « e »*

Les études sur la lisibilité n'ayant pas encore donné vraiment de résultat définitif, force est de nous baser sur la connaissance des typographes et d'utiliser l'axiome « il faut faire par ordinateur ce que faisaient à la main les graveurs de poinçons ». Le problème est que peu d'études ont été faites, ou du moins publiées, sur la façon dont les caractères varient d'un corps à l'autre.

Toutefois, la notion de *Multiple Master* d'Adobe [156] [179] et celle de *MultiType* de KAROW [141] ont ouvert à la voie à quelques publications de caractères à diverses tailles comme ceux de la figure 4.1. Si on étudie cette figure, on remarque que les diverses occurrences du mot *hamburgefons* sont alignées. De son côté, André GÜRTLER, à Bâle, a aussi commencé des études sur ce sujet et *a priori* il obtient aussi des variations linéaires par morceaux.

Mark ARGETSINGER a récemment étudié [123] le *Garamond* d'Adobe : dans sa version normale (sans ajustement optique, figure 4.6-haut) tous les caractères sont homothétiques ; du temps du plomb, ils n'étaient pas identiques (figure 4.6-bas). En étudiant (par scanner) ces caractères déja agrandis, nous avons pu reconstituer la façon dont ces caractères variaient : les points de contrôle des courbes de Bézier sont sur des droites (non parallèles comme le seraient des caractères homothétiques). Partant de deux caractères, par exemple en corps 8 et 40, on obtient les points de contrôle d'un corps intermédiaire *c* par interpolation linéaire (figure 4.7).

La définition générique du « e » *Garamond* consiste alors à remplacer les coordonnées constantes par des fonctions d'interpolation. Considérons par exemple le point B_2 de la figure 4.5 que nous appelons 1 dans

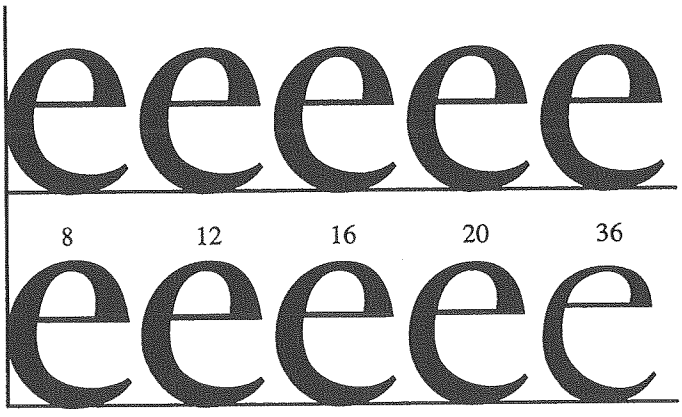


FIG. 4.6 - *Caractères* Garamond *du corps 8 au corps 36, agrandis à la même taille ; en haut, sans ajustement optique, tous les caractères sont identiques ; en bas, avec ajustement optique*

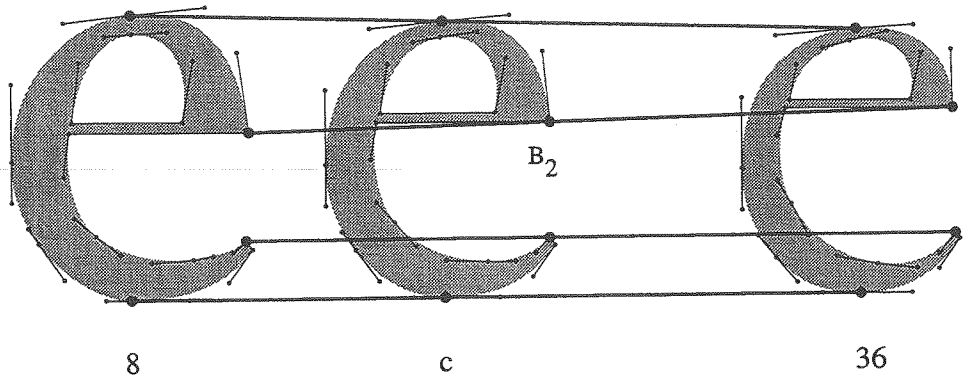


FIG. 4.7 - *Les points de contrôle correspondants des courbes de Bézier sont obtenus par interpolation linéaire - voir figure 4.6*

```
/e{
c := corps actuel
B2,x c := (c × (B2,x 36 − B2,x 8) + 36 × B2,x 8 − 8 × −B2,x 36)/(36 − 8)
B2,y c := (c × (B2,y 36 − B2,y 8) + 36 × B2,y 8 − 8 × −B2,y 36)/(36 − 8)
B2,x c B2,y c moveto % point de départ en corps c ...
} def
```

FIG. 4.8 - *Programme pseudo-PostScript montrant comment le point B_2 de la figure 4.7 en corps c est calculé à partir des coordonnées de B_2 en corps 8 et en corps 32*

la figure 3.12. Les valeurs 402 276 associées à ce point (programme de la figure 3.13) sont donc des valeurs moyennes pour tous les corps dans une fonte normale. Pour faire de l'ajustement optique, il faut connaître ces valeurs numériques pour deux forces de corps (par exemple 8 et 36 de la figure 4.7), soit $B^{8}_{2,x}$, $B^{8}_{2,y}$, $B^{36}_{2,x}$ et $B^{36}_{2,y}$.

Le programme 3.13 devient alors celui de la figure 4.8. Il faut bien insister : cette linéarité ne se produit que pour des points correspondants (il faut d'ailleurs que les caractères aient les mêmes points de contrôle, ce qui n'est pas toujours vrai) et non globalement pour tout le caractère.

Toutefois, il semble moins vrai que l'on puisse encore utiliser des interpolations linéaires pour les extrèmités (très petits corps notamment). KNUTH [143, page vi] a étudié la fonte *Monotype Modern 8A* et semble avoir utilisé des interpolations quadratiques. En tout cas, Yannis HARALAMBOUS [136] s'en inspire et propose aussi de telles interpolations pour les petites valeurs.

Nous avons par ailleurs fait quelques mesures, cette fois sur de très grands symboles, en l'occurrence des accolades (voir ci-dessous la figure 5.8) et avons obtenu le même genre de loi : les points de contrôle des caractères suivent à peu près des fonctions linéaires. Ceci est bien sûr à vérifier et nous espérons mener prochainement une série de mesures à ce sujet.

En fait, la seule étude publiée que nous connaissons, grâce à Richard SOUTHALL (qui a contribué aux études de KNUTH [143, page vi]), est celle de Bridget JOHNSON à Rochester [139] : elle a étudié de nombreux caractères imprimés et a, statistiquement, trouvé que le modèle

$$d = A(h)\, D + B(h)$$

avec

d	=	nouvelle chasse
A(h)	=	facteur d'échelle (corps)
D	=	chasse de référence
B(h)	=	facteur d'épaississement
h	=	nouvelle hauteur

s'appliquait relativement bien à ses propres mesures[2]. Appliqué aux divers facteurs d'un caractère, ce modèle donne, en gros, des fonctions assez proches de fonctions linéaires pour les valeurs « normales » et de fonctions quadratiques aux limites.

2. On lui a toutefois reproché de partir de caractères imprimés et non de « noirs de fumée », c'est-à-dire des empreintes des poinçons noircis par de la fumée de bougie sur un vergé : c'est ça que voit un graveur.

4.3 Espacement entre caractères

LA GRANDE difficulté en typographie classique n'est pas de gérer le noir dans une page, mais le blanc. Tandis que dans la section précédente la forme exacte des lettres est l'affaire de quelques rares personnes (les dessinateurs de caractères et maintenant les informaticiens qui gèrent ces créations de fontes), l'emploi du blanc de la page reste l'affaire des compositeurs. Mais les servitudes du plomb ont amené certaines contraintes à tel point que l'on confond un peu ce qui est règles « visuelles » de ce qui est contrainte technique. Signalons à ce propos que les études les plus remarquables sur l'espacement des lettres ont été justement faites dans un contexte autre que celui du plomb : enseignes (travaux de KINDERSLEY [40]), panneaux d'aéroports (travaux de FRUTIGER pour Roissy [32]) ou panneaux routiers (Michel OLYFF en Belgique [47]). Enfin, le contexte culturel est important : un typographe français trouve que l'absence d'une espace fine avant un deux-points est une atteinte à la lisibilité tandis que l'œil d'un Anglais ou d'un Allemand s'en accomodera fort bien[3].

Les problèmes d'espacement ont donc été liés aux caractères utilisés. GUTENBERG lui-même utilisait une casse très importante et la justification ne se faisait pas dans les blancs (qui avaient donc tous la même chasse), mais en utilisant des abréviations et ligatures diverses (voir le BECHTEL [4] et un récent article du conservateur du Musée Gutenberg à Mayence [62] à ce sujet). Par la suite, on a condensé cette casse à un nombre plus limité de caractères, ligatures et abréviations disparaissant peu à peu.

Un caractère en plomb avait alors la métrique de la figure 3.19 : à gauche, un espace (l'« approche gauche ») et à droite un autre espace (« approche droite »), ces espaces ayant des valeurs telles que l'espacement entre deux lettres successives était bon. En fait ces approches étaient des valeurs moyennes : les séquences $c_i c_j$ $i, j \in [1, n]$ utilisent en effet $2 \times n$ valeurs alors que l'on pourrait imaginer qu'il y en a $n \times n$. Tout l'art du typographe dessinateur de caractère était alors de trouver ces valeurs moyennes. TRACY [23] donne la méthode à suivre.

Ces valeurs moyennes marchaient quand même assez bien, sauf dans des cas spéciaux, comme « A» suivi de « V », l'espace entre ces lettres étant si grand que l'on ne savait plus si « AVIS » contenait un ou deux mots. La

3. Pour la petite histoire, ces règles d'espacement ont beaucoup évolué dans le temps : le *Manuel* de FOURNIER [10] a été composé en 1764 avec une espace devant chaque virgule ; le *Manuel de typographie* de FREY, [11, art. approche], en 1857 signale bien qu'il ne faut pas mettre d'espace avant le point d'interrogation, mais aujourd'hui (sans doute à la suite de la règle mnémonique apprise dans les écoles de secrétariat : « on met un espace avant les signes de ponctuation formés de deux parties : ; ! ? ») il et de coutume d'en mettre. Fernand BAUDIN montre [27] que la règle de ne pas mettre plusieurs coupures de mots à la suite n'est qu'une invention très récente, etc.

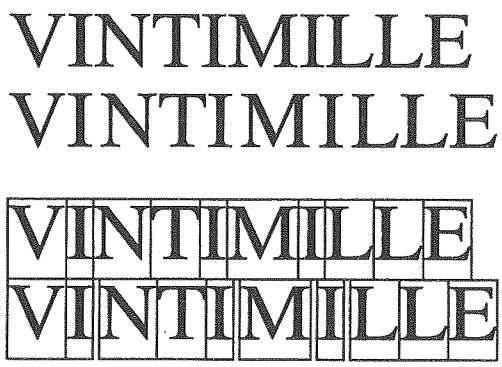


FIG. 4.9 - *Le mot « VINTIMILLE » a des blancs irrégulièrement répartis. Pour compenser les creux de certaines lettres, il faut rajouter des espaces*

solution alors était de mortaiser ces caractères en plomb, c'est-à-dire de leur faire un « cran », de « créner ». C'est le crénage *(pair kerning).*

Toutefois, dans de nombreux cas, la solution n'était pas de rapprocher les lettres, mais au contraire de les écarter de façon que les mots aient une impression de gris uniforme. En 1857, FREY [11], prenait comme exemple le mot « VINTIMILLE » où on ne peut pas créner après le « V » (car le « I » serait alors contre le haut du « V » ni de part et d'autre du « T ». Par ailleurs les « L » ont aussi un grand creux. Pour homogénéiser tous ces espaces, la seule solution est alors de rajouter des espaces entre les lettres à bords droits, c'est-à-dire dans « IN » et dans « IMI » (figure 4.9). Michel OLYFF base tout son cours sur les modifications d'approche sur le seul mot « LAPIN » !

Avec la photocomposition, la dimension « verticale » des caractères a disparu et il a donc été possible de serrer ou éloigner les caractères à volonté. C'est alors que l'on a pris la « mauvaise » habitude de faire de l'interlettrage *(track kerning).*

C'est avec les machines à composer puis avec la photocomposition qu'est arrivée la notion de table de chasses que l'on retrouve en typographie numérique : chaque caractère a un point de référence. Lorsqu'un caractère est flashé, son image (le plan de bits en cas de fonte numérisée) est amenée de façon que son point de référence soit mis sur le « point courant » (marqué par le point x, y en figure 3.19) de la ligne en cours de flashage. Ce point courant est alors augmenté de la chasse w_x du caractère (et devient donc le point x', y'). Ces valeurs w_x doivent être connues de la photocomposeuse ou de l'imprimante et lui sont fournies dans ce qu'on appelle une table de chasses.

La même technique a été adoptée pour les fontes numérisées et la recherche de qualité a même fait qu'on y a ajouté des tables de crénages donnant la valeur recommandée de crénage entre deux caractères (on trouvera dans [91] la façon dont sont conçues ces tables en PostScript). Mais ce système est un peu « absurde » :

(1) les tables de chasses ont des moyennes puisqu'elles tiennent compte des approches qui sont elles mêmes des valeurs moyennes ;

(2) dans une fonte de n caractères, il y a pratiquement $n \times n$ valeurs de crénage si on veut corriger ces valeurs moyennes, ce qui fait finalement $n + n \times n$ valeurs si on tient compte des tables de chasses ;

(3) de toute façon ces tables sont données pour un corps moyen (12 en général) et des corrections doivent s'ajouter en fonction du corps.

Ce système d'approches doit donc être repensé et les solutions reposent sur deux méthodes.

(1) *Kerning on the fly* : les calculs de l'espace entre deux lettres se font au moment de s'en servir[4] : la distance entre les *bbox* de deux caractères $c_g c_d$ peut se trouver dans une table $K_{g,d}$. Pour 300 caractères, cette table devrait avoir 300×300 éléments ; mais, d'une part, on peut regrouper certaines lettres par affinités topologiques et, d'autre part, on a, de toute façon, des tables de crénages pratiquement aussi grosses.

C'est la solution proposée actuellement par certains fondeurs et notamment Peter KAROW dans le cas de son *hz-program* [141]. Le problème, maintenant, n'est plus technique mais commercial : il faudrait réécrire tous les formateurs pour qu'ils puissent travailler avec de tels formats de fontes !

(2) Calcul automatique de l'espace entre lettres : puisque l'espace entre les lettres dépend essentiellement de la forme de ces lettres, on peut concevoir une fonte qui calculerait automatiquement ces espaces. C'est ce que nous avons fait avec le *Delorme* (voir section 6.2 et figure 6.15). Mais il s'agit d'un cas très simple puisque les formes de ces lettres sont très limitées et connues.

Claude BETRISEY est allé beaucoup plus loin [124] [125] : grâce à son modèle de caractères, un programme est capable de déterminer si une face est convexe, si un caractère contient une cavité, etc. Il applique alors, comme en traitement d'image, de la lumière qui donne des zones d'ombre dont il tient alors compte pour calculer la surface « visuelle » du caractère.

4. D'ailleurs, PostScript fournit une instruction qui permet de faire cela : kshow ; voir [91].

4.4 Conclusion

JUSQU'À PRÉSENT, l'effort avait été fait pour que les caractères soient corrects compte-tenu des contraintes de la grille (section 3.3.2) et relèvent de ce que l'on peut appeler la qualité « technique ». Il fallait bien commencer par cela. Les nouvelles recherches sur la qualité « typographique » des caractères (calcul des formes, des espaces, etc.) nous semblent beaucoup plus importantes et prometteuses, bien qu'elles soient peut-être plus difficiles car elles font appel au « visuel », lequel reste encore « subjectif » !

Fontes dynamiques
Application aux symboles mathématiques

DANS LE CHAPITRE précédent, nous avons vu que l'on pouvait construire des caractères avec des variables analytiques et non uniquement avec de simples constantes. Ici, nous montrons l'intérêt, et la possibilité en PostScript, de créer des fontes que nous avons appelées dynamiques [157], faisant ainsi explicitement allusion aux propriétés dynamiques des langages de programmation, en opposition à leurs propriétés statiques. Nous aurons toujours en tête le parallèle:

élaboration d'une fonte	≡	compilation d'un ensemble de procédures;
impression d'un caractère	≡	exécution de l'une de ces procédures.

Dans une seconde partie, nous montrerons, à titre d'illustration, comment une fonte dynamique permet de construire des symboles mathématiques dont la taille est directement liée au terme qu'ils englobent.

5.1 Désactivation du mécanisme de cache en PostScript

LA SECTION 4.1 nous a montré que si l'on faisait varier la valeur d'une variable, par exemple une ordonnée d'un point de contrôle d'une courbe de Bézier décrivant un contour de caractère, un mécanisme de cache faisait que l'on ne modifie en rien la forme de ce caractère.

Or, il se trouve que ce mécanisme de cache des fontes PostScript peut être désactivé. C'est même en fait le processus « normal », le plus simple, pour appeler une fonte. Mais il ne présentait *a priori* aucun intérêt, sauf de pouvoir changer l'opacité de certaines couleurs comme l'indiquait le premier manuel de référence du langage [85, page 213].

Nous avons découvert ([157], [158]) que la désactivation de ce mécanisme de cache permettait justement d'utiliser des variables analytiques dans la description de fontes de façon vraiment dynamique. Depuis, le nouveau manuel de référence du langage en fait état [86, page 496] et, comme nous le verrons au chapitre 6, de nombreuses utilisations en ont été faites.

En désactivant ce mécanisme de cache, les instructions suivantes (voir section 4.1.2)

```
/dx 100 def
/dy  90 def
    (L) show
/dx 120 def
/dy 130 def
    (L) show
```

donnent bien successivement les figures 4.2-e et -f.

5.1.1 `setcachedevice` et `setcharwidth`

En décrivant une fonte de type 3, le mécanisme de cache est appelé lorsque l'on utilise l'opérateur `setcachedevice` depuis la procédure `BuildChar` comme nous l'avons fait dans le programme de la figure 4.3. Pour ne pas activer ce mécanisme de cache, il suffit d'employer l'opérateur `setcharwidth` qui n'a pas besoin de connaître la *bbox* du caractère (puisqu'il n'y pas à réserver de place mémoire pour le plan de bits) et ne demande donc que les deux valeurs utiles pour mettre à jour les coordonnées du point courant.

La procédure `BuildChar` du programme de la figure 4.3 doit alors être définie comme suit:

```
/BuildChar {
    ...
      wx wy setcharwidth
    ...
    } def
```

5.1.2 Fontes dynamiques et passage de paramètres

Sous ce titre, nous étudions deux choses différentes:

Comparaison avec le passage de paramètres des langages de programmation

Appeler un caractère *x* (par exemple `x show`) revient à appeler un paramètre *x* d'une procédure d'un langage de programmation un peu riche en passage de paramètres (par exemple Algol60, Simula67 ou Algol68 [204, page 195]): à une fonte statique (avec mécanisme de cache) correspond un

FIG. 5.1 - *Trois résultats du même appel* (AAAAA) show.
Les deux premières lignes après deux évocations de la même fonte aléatoire mais statique; la dernière après évocation de la même fonte mais dynamique.

passage d'un paramètre par valeur (le paramètre et le plan de bits du caractère ne sont calculés qu'une seule fois) tandis qu'à une fonte dynamique correspond un passage de paramètre par nom (le paramètre et le plan de bits du caractère sont réévalués à chaque instanciation).

Voici un exemple montrant la différence. Ajoutons à la fonte de la figure 4.3 la lettre «A» définie comme suit:

```
/A {40 setlinewidth
0 0 moveto
250 500 linetoa                    %sommet à +|- 50 près
500 0 lineto
125 250 movetoa
375 250 linetoa                    %  barre à +|- 50 près
stroke} def
```

où movetoa et linetoa sont des opérateurs équivalents, respectivement, à moveto et lineto mais modifiant de façon aléatoire les coordonnées de chaque point à ±50 points près et définies ainsi en pseudo-PostScript:

```
/nbal{%met sur pile un nombre aléatoire compris -50 et +50
 (rand modulo 100) - 50 } def
```

```
/movetoa {%x y
          moveto x+nbal y+nbal } def
/linetoa {%x y
          lineto x+nbal y+nbal } def
```

Appelons cette fonte `Static` si on utilise l'opérateur `setcachedevice` ou `Dynamic` si on utilise l'opérateur `setcharwidth`. Alors les appels suivants

```
/Static findfont 100 scalefont setfont
     0 140 moveto (AAAAA) show
/Static findfont 99 scalefont setfont
     0 70 moveto (AAAAA) show
/Dynamic findfont 100 scalefont setfont
     0 0 moveto (AAAAA) show
showpage
```

produisent les trois lignes de la figure 5.1. En effet, lors du premier appel de la fonte statique (en corps 100), le plan de bits du premier « A » est calculé, avec des valeurs aléatoires pour le sommet du A et pour celles des extrémités de la barre horizontale. Mais pour les 4 appels suivants, on reprend ce même plan de bits. La fonte en corps 99 n'est plus la même que celle en corps 100. On recalcule donc le plan de bits de « A » (avec de nouvelles valeurs aléatoires) qui est donc différent du précédent. Mais les 5 « A » de cette seconde ligne sont identiques. Par contre à chaque appel dynamique (ligne du bas), on recalcule le plan de bits de chaque « A » avec de nouvelles valeurs aléatoires : les 5 « A » du bas sont donc tous différents.

Passage de paramètres à une fonte

Les fontes PostScript sont des procédures, qui sont elles-mêmes décrites dans un « dictionnaire » *read only*, c'est-à-dire que l'on ne peut pas en modifier les valeurs. Qui plus est, PostScript n'offre aucun mécanisme implicite de passage de paramètres à une fonte.

Il faut donc programmer tout ceci en définissant un dictionnaire annexe où l'appelant (`show`) mettra les paramètres actuels et où la fonte appelée ira les lire et y mettre d'éventuels résultats intermédiaires.

Nous verrons (chapitre 7) que nous proposons que ce mécanisme soit rendu plus implicite en PostScript.

Comparaison avec les *MultipleMasters*

Certaines firmes commerciales ont mis sur le marché il y a à peine un an des fontes basées, *a priori*, sur le même principe, par exemple *MultipleMasters* d'Adobe ou *MultiType* d'URW. En fait, si ces fontes partent effectivement d'un dessin analytique unique (maître), ces *masters* produisent des fontes où les variables se voient affectées de leurs « valeurs » (comme les paramètres du même nom) et qu'il faut alors charger normalement. Le dynamisme n'a donc lieu qu'avant le chargement et non lors de l'évocation des dits caractères. Nous comparerons ces divers mécanismes en 6.4.

5.2 Application aux symboles mathématiques

5.2.1 Besoins

LES FORMULES MATHÉMATIQUES sont des éléments particuliers des textes scientifiques et leur composition a été très longtemps l'apanage de rares imprimeurs.

Jeu de caractères

Les mathématiques ont besoin de très nombreux caractères qui, du temps du plomb, faisaient partie de casses spéciales appelées « casseaux » et que seuls certains fondeurs offraient.

Les mathématiques utilisent

- des lettres de diverses polices, par exemple « a », « a », « α », « $\mathcal{A}$ », « $\aleph$ »,
- et toute une « ménagerie » (selon l'expression de Lamport [73]) de symboles (« $\times$ », « $\sqcap$ », « $\otimes$ », « ∞ », « $\sqsubset$ », « $\Leftrightarrow$ », etc.).

Notons aussi le besoin d'invention de caractères nouveaux à mesure que les mathématiques progressent.

Symboles à taille variable

Les mathématiques utilisent, par ailleurs, divers filets (par exemple barres horizontales de fractions ou celles verticales délimitant les déterminants) et tout un ensemble de symboles à taille variable (intégrales, signes de sommation ou de racine carrée, parenthèses, crochets, accolades, etc.). La taille de ces symboles est liée à leur « contenu » comme on peut le voir pour la hauteur totale et la longueur du filet des symboles de racine carrée

dans la formule suivante :

$$\sqrt{\frac{1}{1+\sqrt{1+\sqrt{x}}}}$$

Mais ces symboles doivent suivre les règles d'ajustement optique (*optical scaling*; voir 3.7) : si on agrandissait une intégrale de corps 50 par quelque transformation affine pour qu'elle ait la taille d'une intégrale de corps 100, cette transformation s'appliquerait aussi à son épaisseur et le signe serait beaucoup trop gras (figure 5.2). Par contre, ce dernier symbole a la bonne graisse si on veut l'appliquer sur un terme (par exemple $f(x)dx$) en corps 100 pour préparer un transparent de rétroprojection ou quelque affichage sur écran.

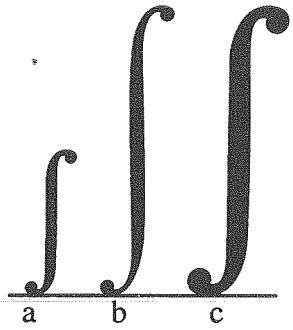


FIG. 5.2 - *Une intégrale ne peut être agrandie par transformation affine : a) une intégrale en corps 50, b) la même intégrale agrandie deux fois en hauteur, mais avec la même graisse , c) la même intégrale en corps 100.*

C'est pourquoi, autrefois, les catalogues de fontes mathématiques offraient toute une série de symboles de taille variable mais pour diverses graisses, celles-ci étant mesurées d'ailleurs en points car il s'agissait de la mesure de l'épaisseur des signes (figure 5.3).

Composition

Enfin, et c'est loin d'être le plus simple, ces symboles de taille fixe ou variable doivent être composés selon des règles rigoureuses (mais relativement peu décrites formellement) touchant aussi bien l'emploi des signes (par exemple les noms de variables se mettent, souvent, en italique, ceux des fonctions en romain), leur position relative (on centre un numérateur par rapport à une barre de fraction), leur taille (si dans a^{i+j}, a est, par exemple, en corps 12, alors i, j et + sont en corps 10). On doit aussi, comme toujours en typographie, respecter les usages des espaces pour des raisons de lisibilité.

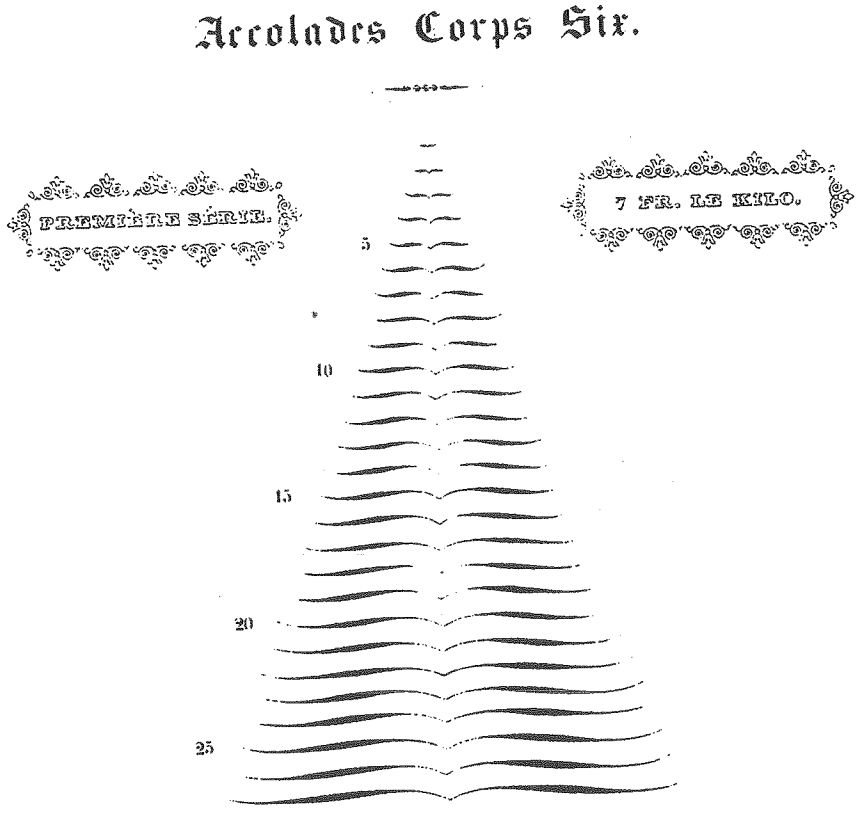


FIG. 5.3 - *Accolades d'après un catalogue de caractères en plomb de 1926 [51], musée de l'Imprimerie, Lyon*

Du temps du plomb et de la photocomposition de première ou seconde génération, beaucoup d'erreurs étaient produites dans les ateliers de composition et il y avait beaucoup plus de typographes affectés aux tâches de correction que de composition !

5.2.2 Fontes mathématiques informatisées

LA COMPOSITION des formules mathématiques par ordinateur a été assez longtemps mal conçue (les auteurs de systèmes les traitaient comme des images ce qui rendait la moindre correction très difficile). Ce sont Brian KERNIGHAN et L. CHERRY (avec EQN [69]) puis Donald KNUTH (avec TEX [70]) qui, les premiers, ont compris qu'une formule pouvait être décrite par une arborescence et que la composer revenait alors à emboîter des morceaux de formule, les tailles des boîtes se déduisant les unes des autres par héritage. C'est probablement Vincent QUINT qui a construit le premier système – Titus [77] – permettant la saisie et la correction interactives d'une formule.

La composition des formules mathématiques a donc fait de grands progrès. Il reste malgré tout des problèmes liés au tracé des symboles :

(1) Il existe très peu de fontes numérisées permettant de composer des mathématiques. Ces fontes se limitent en fait pratiquement aux suivantes :

- *Symbol* qui est, avec *Times*, *Courier* et *Helvetica*, l'une des quatre fontes fournies en standard sur toute imprimante PostScript depuis que ce langage existe ; voir [86, appendix E.11] ;
- l'extension mathématique *cmex*nn des fontes *cmex* créées par le logiciel METAFONT pour TEX [71, appendix F] ;
- la famille *Lucida* dessinée par BIGELOW & HOLMES [126] qui, outre *LucidaMath-Symbol* correspondant à peu près à *Symbol*, comprend *LucidaMath-Extension* où se trouvent les symboles propres à TEX de *cmex*nn ;
- enfin, *Euler* et un *Times* mathématiques offrent depuis peu ces symboles [68, pages, 183–185].

Parmi ces familles de fontes, deux seulement permettent aujourd'hui de composer un ouvrage mathématique de façon homogène, c'est-à-dire avec des caractères de même style tant pour le texte, les mathématiques que pour les éventuels programmes composés traditionnellement en caractères de type machine à écrire à chasse fixe. Il s'agit de *cmr* et de *Lucida*[1].

1. Hélas, cette fonte a un gros œil, c'est-à-dire des minuscules très hautes, ce qui n'est pas dans le goût français [43]. C'est néanmoins celle utilisée ici !

(2) La composition des symboles mathématiques est basée sur une décomposition en morceaux plus petits[2]. Mais leur assemblage ne fournit pas souvent la qualité désirée pour des travaux professionnels. Les principaux problèmes sont les suivants :

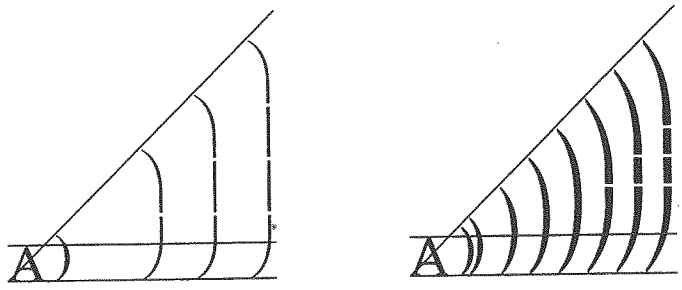


FIG. 5.4 - *Composition de symboles par assemblages de morceaux élémentaires. Les traits blancs horizontaux indiquent la séparation de ces morceaux. À gauche :* Symbol, *à droite :* Lucida. *Les « A » indiquent la force de corps du caractère employé.*

- les symboles de taille variable sont décomposés en éléments rectilignes (horizontaux ou verticaux), faciles à mettre bout à bout mais ne rendant pas, par exemple, le galbe espéré des parenthèses (voir figure 5.4) ;
- cette décomposition n'étant pas aisée pour les éléments non rectilignes, certains systèmes préfèrent ne pas permettre de grandes tailles plutôt que d'avoir des symboles peu esthétiques ; c'est ainsi que TEX n'offre qu'une intégrale en mode *display*, mais au moins est-elle oblique ;
- ces symboles étant définis de façon discrète, des tailles peuvent manquer dans certaines fontes et pas dans d'autres (comparer *Symbol* et *Lucida*, figure 5.4) ;

2. Ceci n'est pas nouveau : dans son *Manuel de typographie* de 1764, FOURNIER dit que c'est la façon de faire les grandes accolades [10].

- ces « trous » sont particulièrement sensibles pour les petites parenthèses ; ainsi n'est-il pas rare de voir des parenthèses assemblées comme suit en utilisant *Symbol*:

  « ⎛ ⎞ » ;

- on peut avoir des résultats inattendus avec LaTeX comme le montrent les deux formules suivantes :

$$\overbrace{a+b+c+d} \quad \Rightarrow \quad \overbrace{a+b+c+d}$$
$$\widehat{a+b+c+d} \quad \Rightarrow \quad a+\widehat{b+c+c+d}$$

- enfin, tant qu'à rechercher la perfection, il est dommage que (La)TeX, ne disposant que d'un jeu limité de parenthèses, compose la formule

```
\[\left(A^{A^B}\right)+
  \left(A^{A^{A^B}}\right)+
  \left(A^{A^{A^{A^B}}}\right)
\]
```

comme ceci

$$\left(A^{A^B}\right)+\left(A^{A^{A^B}}\right)+\left(A^{A^{A^{A^B}}}\right)$$

c'est-à-dire en utilisant deux paires de parenthèses de même taille pour les deux termes de droite ; ceci explique que le « B » du dernier soit plus haut que les parenthèses « englobantes ». Une bonne typographie voudrait en fait que toutes ces parenthèses aient la même taille, un peu plus grande que la dernière. Mais TeX n'en dispose pas.

5.2.3 Implémentation de symboles mathématiques dynamiques dans Grif

Il nous a donc paru utile de définire une fonte dynamique de symboles mathématiques dont le nom, *Math-Fly*, fait allusion à l'expression anglo-saxonne, *on the fly*, pour ces calculs qui sont faits « à la volée », c'est-à-dire à la demande et non une fois pour toutes. Nous sommes en train de l'installer, en collaboration avec Irène Vatton de Grenoble, dans le système de manipulation de documents Grif [166]. L'avantage de cette intégration dans Grif est évident : notre fonte a un mode d'appel non standard car il faut lui passer des paramètres (c'est-à-dire qu'elle ne peut être appelée directement par TeX, par exemple). Nous aurions évidemment pu modifier le formateur TeX pour cela (le code est dans le domaine public). Mais il se trouve que nous avons de bonnes relations avec l'équipe grenobloise

qui travaille sur Grif et c'est donc tout naturellement que nous avons pu collaborer.

L'éditeur Grif

Grif est un système de manipulation de documents conçu et développé par Vincent QUINT et Irène VATTON [78] [66], [79], [65]. C'est un éditeur qui, contrairement aux systèmes de PAO ordinaires, ne s'intéresse pas uniquement à la forme graphique des documents, mais aussi à la structure logique. Un langage, appelé S, permet de donner la description logique du document - il est équivalent aux DTD *(Document Type Definitions)* de SGML. Un autre langage, P, permet de spécifier les règles de présentation pour affichage sur écran (Grif est complètement interactif) ou pour une sortie sur imprimante. C'est l'utilisation de P, associé à la notion de boîte abstraite (à la TEX), qui a rendu possible la sortie de formules mathématiques utilisant dynamiquement notre fonte.

Dessin des caractères

Nous partons de la description PostScript du contour d'un caractère, obtenue soit en programmant directement à la main (cas des flèches et crochets par exemple), soit en partant de la description d'un caractère du domaine public, soit en partant d'un dessin au crayon et en le numérisant par un scanner et un produit commercial comme Fontographer ou Ikarus. Toutes ces descriptions fournissent des coordonnées numériques. Le problème est donc de passer aux coordonnées analytiques.

Le principe est de remplacer, dans cette description, toutes les instructions de dessin (`moveto`, `lineto`, `curveto`) par des instructions `movetox`, `movetoy`, `movetoxy`, `linetox`, `...` remplaçant chaque point x, y par $x + \{\delta x\}, y + \{\delta y\}$, la partie entre accolades étant optionnelle.

Vu de l'utilisateur (auteur ou formateur), il y a alors plusieurs cas:

(1) Les symboles dont l'extension reste une homothétie selon un seul axe. C'est le cas des symboles verticaux (crochets [ ], barres | ‖, flèches ↑ ⇓, etc.) et des symboles horizontaux (flèches ⟸⟹↦, barres $\underline{a + \overline{b + c} + d}$), etc. Ceci est aussi valable pour les symboles ayant des fioritures telles que des empattements (c'est par exemple le cas des crochets) qui sont simplement translatés. Mais c'est aussi le cas de double homothétie (selon l'axe des x et selon celui des y) comme pour les symboles ↗ et aussi $\sum$ ou $\prod$, voire $\surd$ que l'on peut indifféremment déformer en x ou en y sans donc changer la graisse. Voir figures 5.5 et 5.6.

(2) Les symboles qui sont translatés en x et/ou en y mais qui, pour des raisons d'esthétique, sont légèrement déformés en y et/ou en x. Par

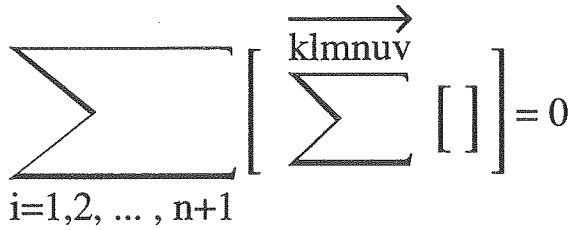


FIG. 5.5 - *Symboles mathématiques transformés linéairement en x et/ou en y sans modification de la graisse. Comparer avec la formule 2 page 75*

exemple une parenthèse gauche sera légèrement galbée sur la gauche, et un chapeau légèrement vers le haut. Cette déformation peut en fait être vue comme une simple fonction linéaire dont les paramètres peuvent être connus et du formateur et de la fonte. Voir figure 5.7.

(3) Enfin des symboles dont les déformations en x et en y ne sont pas linéaires (intégrale, véritables accolades, etc.).

Mais, vu de PostScript, il y a deux cas : ou bien on ne touche pas aux points de contrôle des courbes de Bézier (ce sont les deux premiers cas précédents) et alors il n'y a pas de problèmes majeurs ; ou bien on y touche et alors le problème est de trouver le lieu où vont se déplacer ces points de contrôle. C'est le problème de l'ajustement optique (*optical scaling* ; voir 3.7) non plus sur de tous petits corps (comme l'a étudié Bridget JOHNSON [139]) mais sur de grands caractères (ce qui est sans doute plus facile à étudier car leur tracé est moins sensible, proportionnellement, à la finesse des poinçons).

Considérons les accolades de la figure 5.3 et, après saisie par un scanner, agrandissons-en trois de la même graisse mais de tailles différentes (figure 5.8, gauche). On voit que ces accolades ont effectivement la même épaisseur mais des formes différentes. Mais si on regarde de plus près les points de contrôles des courbes de Bézier ramenées à la même échelle, on voit que les points correspondant sont en progression linéaire (du moins pour ceux qui ne sont pas aux extrémités, leur position étant très sensible aux coups de gouge[3]) : les points A_2 (sur une accolade de 20 points de haut), A_4 (accolade de 40 points) et A_5 (accolade de 50 points) sont alignés et le rapport A_2A_4/A_2A_5 est proportionnel au rapport $(20-40)/(20-50)$.

3. Ce sont de tels défauts qui apparaissent sûrement sur la partie inférieure des moyenne et grande accolades de la figure 5.8.

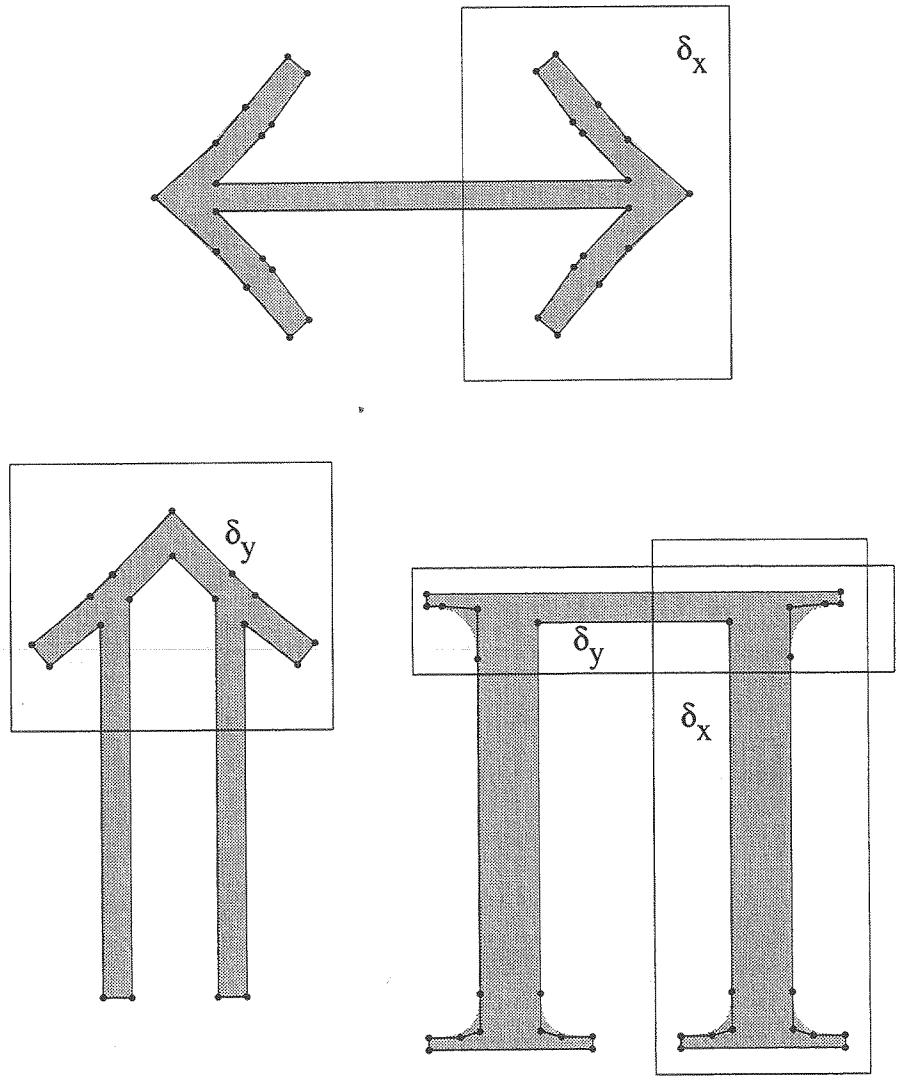


FIG. 5.6 - *L'extension des symboles ayant des segments horizontaux ou verticaux se fait par incrément de l'abscisse de δx pour les points dans le rectangle correspondant, ou de l'ordonnée de δy, ou les deux*

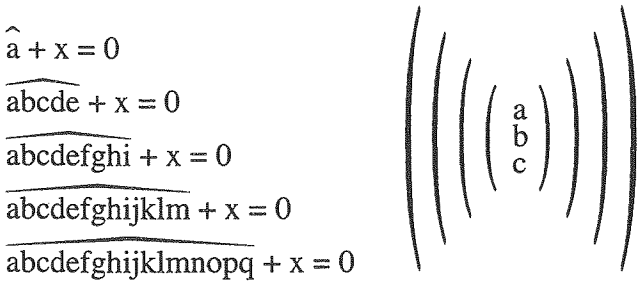


FIG. 5.7 - *Symboles mathématiques transformés par homothétie avec une légère déformation orthogonale à l'axe d'homothétie*

Ceci se confirme aussi sur les autres points de la panse de l'accolade et pour d'autres symboles ; nous pouvons alors employer la technique suivante :

(1) dessiner un caractère normal ;

(2) dessiner un caractère de même force de corps mais agrandi (tout en conservant donc la graisse) ;

(3) entre les deux, interpoler linéairement (mais ce n'est pas parce que chaque lieu des points de contrôle d'une courbe de Bézier est un segment de droite que toutes ces droites sont parallèles ; globalement la transformation n'est pas une homothétie).

C'est apparemment ce que font aussi des produits comme Multiple Masters d'Adobe et MultiType d'URW.

Passage de paramètres

Entre la fonte et le formateur, des informations doivent passer dans les deux sens :

(1) De la fonte vers le formateur : nous avons dû prévoir une modification de l'AFM (voir section 3.4.2) : en effet, il convient de passer au formateur les informations sur le type du symbole (extension en *x*, en *y*, les deux?), les valeurs minimales de la *bounding box* et surtout, en cas de déformation orthogonale d'un symbole (comme c'est le cas pour

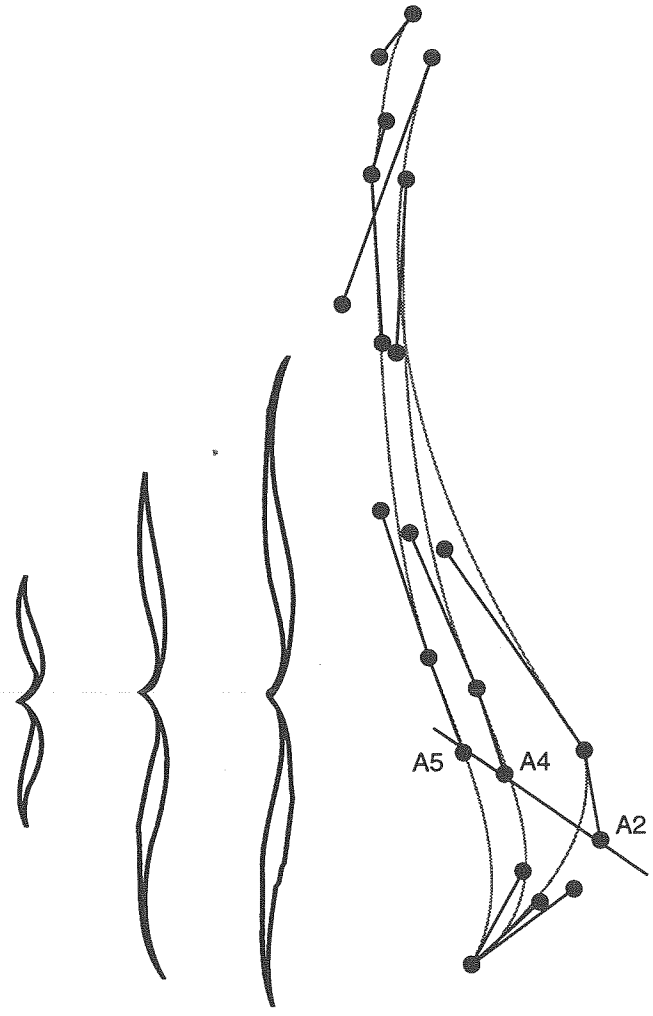


FIG. 5.8 - *Ajustement optique* (Optical scaling) *de symboles mathématiques : à gauche 3 accolades de même graisse (donc de même corps) à des tailles différentes (20, 40 et 50 points) ; à droite, détail des contours (partie supérieure gauche des 3 accolades ramenées à la même échelle) avec indication des points de contrôle des courbes de Bézier. Cette figure a été élargie de 3 fois en x.*

le chapeau ou les parenthèses), les coefficients de la fonction linéaire adoptée.

(2) Dans l'autre sens, c'est au formateur de décider de la taille voulue des symboles. Notre fonte demande donc de recevoir les coordonnées de la *bounding box* à utiliser. Pour cela, nous passons simplement les paramètres par le biais d'un dictionnaire ainsi que nous l'avons expliqué plus haut en 5.1.2.

Calcul de la taille des symboles

Le calcul des formules mathématiques par Grif repose sur son schéma général: un document est défini par une arborescence décrivant sa structure. Le langage P[4] permet d'attribuer à chaque élément de cette structure une «présentation» physique, en général par héritage des propriétés métriques des feuilles ou sous-arbres internes à un nœud donné.

Considérons la formule suivante:

$$m = \sum_{k=\min(1,i)}^{n} \sin^2 x_k \qquad (1)$$

Grif construit une image abstraite de cette formule, dans ce cas-ci (en fonction de la syntaxe associée) l'arbre de la figure 5.9. Il lui correspond la hiérarchie de boîtes abstraites de la figure 5.10, cette notion de boîte étant équivalente à celle de TEX. À chaque nœud de l'arbre est associée une série d'attributs positionnels comme

```
...
Sum:        HorizontalPosition:
                 Left = String.Right;
            VerticalPosition:
                 BaseLine = String.BaseLine;
Symbol: ,   HorizontalPosition:
                 Center = Lower_exp.Center;
            VerticalPosition:
                 Bottom = Lower_exp.Top;
Upper_exp:  HorizontalPosition:
                 Center = Lower_exp.Center;
            VerticalPosition:
                 Bottom = Symbol.Top;
Operand:    HorizontalPosition:
                 Left = Symbol.Right;
```

4. C'est un maquettiste qui s'en sert en fonction de la mise en page voulue.

```
            VerticalPosition:
                  BaseLine = Symbol.BaseLine;
...
```

Ces positions utilisent les coordonnées des quatre coins (*Top, Bottom, Left* et *Right*) d'une boîte, son centre ou sa ligne de base. Celle-ci peut être définie en fonction de ses descendants. Par exemple, ici nous aurons :

```
Sum:        BaseLine = Symbol.BaseLine;
Operand:    BaseLine = String.BaseLine;
```

L'application de ces différentes contraintes donne ici

(1) la chaîne `m=`, le symbole $\sum$ et la chaîne `sin` seront alignés sur la même ligne de base ;

(2) la borne supérieure `Upper_exp` sera centrée au dessus du symbole $\sum$;

(3) la borne inférieure `Lower_exp` sera centrée au dessous du symbole $\sum$.

Par ailleurs, les tailles des boîtes peuvent aussi êtres définies relativement ou de façon absolue. On écrit

```
Symbol:     Width =  Lower_exp.Width;
            Height = Operand.Height * 1.2;
```

le calcul de contraintes donnant alors ici :

(1) le symbole $\sum$ doit avoir une chasse égale à celle de la borne inférieure ;

(2) le symbole $\sum$ doit avoir une hauteur égale à 1.2 fois celle de son opérande.

Avec une fonte normale du type *Symbol*, il n'est pas possible de modifier la taille des symboles, aussi a-t-on des expressions comme la suivante :

$$\sum_{i=1,2,\ldots,n+1} \left[\overline{\sum^{klmnuv}} [\,] \right] = 0 \qquad (2)$$

Par contre, Grif peut ainsi passer à notre fonte *Math-Fly* les hauteur et largeur désirées pour le symbole $\sum$, ce qui donne la figure 5.5. Dans le cas de la formule 1, nous obtenons avec Grif la formule de la figure 5.11. On peut ne pas aimer cette forme un peu dégénérée du symbole somme. Aussi cherchons-nous, à l'aide de tests, jusqu'où ne pas aller trop loin dans la déformation des symboles pour ne pas choquer les lecteurs trop habitués aux symboles figés de *Symbol* !

Cette fonte est actuellement en phase de test. Les modifications dans le système Grif sont faites grâce aux mécanismes d'API *(application programming interface)* et d'ECF *(external call facility)* qui font de Grif un système de manipulation de documents actifs [76].

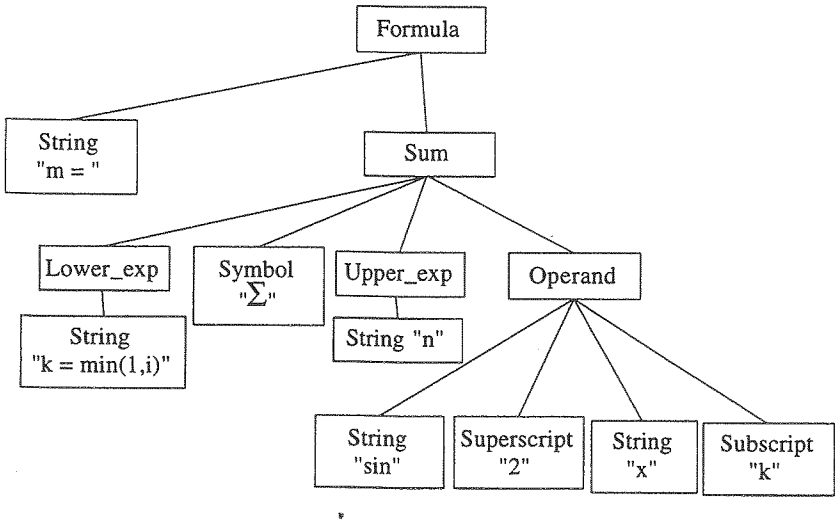


FIG. 5.9 - *Image abstraite de la formule 1*

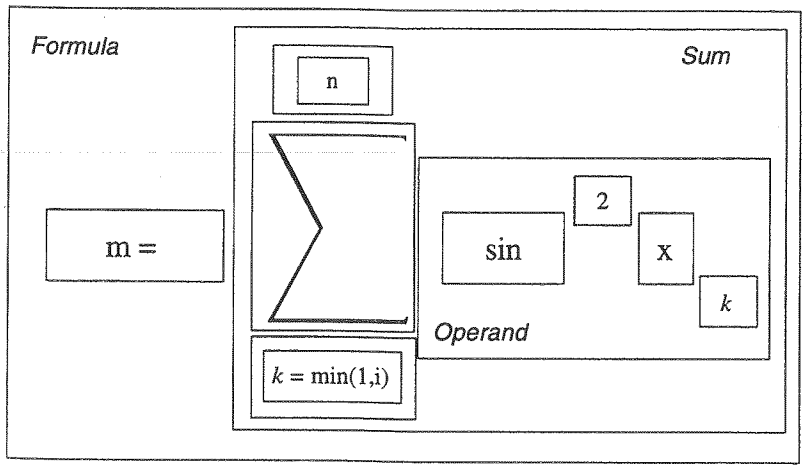


FIG. 5.10 - *Boîtes associées à la formule 1*

$$m = \sum_{k=\min(1,i)}^{n} \sin^2 x_k$$

FIG. 5.11 - *La formule 1 vue par Grif+ Math-Fly*

5.3 Conclusion

LA POSSIBILITÉ de donner des tailles quelconques aux symboles revient à faire ce que l'on fait à la main au tableau noir ou sur le papier : être libre de ses mouvements. Les contraintes physiques des caractères en plomb ne permettaient pas de donner cette liberté. Mais, pire, elles ont en quatre cents ans supprimé l'idée que cette liberté pouvait exister. Nous allons en voir dans le prochain chapitre quelques possibilités.

Page vide

Caractères aléatoires et caractères contextuels

Le chapitre précédent a montré que l'on pouvait dessiner des caractères « dynamiquement », c'est-à-dire lors de leur évocation (quand on les imprime) et non une fois pour toutes (au chargement, avant leur emploi). Cette possibilité, même si finalement elle ne revient qu'à changer un appel de procédure et à utiliser des variables analytiques au lieu de constantes numériques, est beaucoup plus importante qu'il ne paraît. Gutenberg est parti de l'écriture manuscrite - libre de toute contrainte - pour figer des caractères dans une métrique rigoureuse et dans un dessin unique. Les fontes dynamiques, sans éliminer la rigueur du plomb que l'on peut conserver, redonnent aux caractères leur liberté.

Il a fallu des années (avec Aldus, Caxton, Champfleury, Fournier, Didot, etc.), voire des siècles (avec les Excoffon, Lubalin, Tschichold, Zapf, etc.) pour qu'un nouvel art typographique se démarque de l'écriture manuscrite et de la calligraphie. Parallèlement, de nombreux poètes et écrivains ont essayé de créer de nouvelles formes d'écriture (avec le double sens de ce mot: forme et fond), par exemple Malarmé, Apollinaire et, plus près de nous, « le groupe Oulipo ».

Toutefois, leurs essais ont toujours souffert du carcan du plomb[1] mais ce phénomène a été rarement analysé[2]; les premières analyses des nouvelles possibilités de typographie par ordinateur se sont en fait limitées à celles du Minitel[3].

Il faudra sûrement des années pour qu'un nouvel art typographique « dynamique » atteigne quelque maturité. Mais il est passionnant de voir que déjà certains graphistes comme van Blokland ou di Sciulo - nous y reviendrons en 6.1.3 et 6.3.4 - sont attirés par de telles recherches et que la plus moderne des revues de typographie (aux USA: *Emigre*) leur offre ses

1. Nous avons montré, voir [120], la différence que pouvait avoir un calligramme d'Apollinaire très mal imprimé au plomb dans la collection de la Pléiade et le même programmé en PostScript.

2. Peu de publications ont été vraiment consacrées au blocage des artistes par le plomb; citons toutefois les colloques sur *Le texte et son inscription* et sur *Le texte en mouvement* de Roger Laufer [42] [41] ainsi qu'un numéro spécial de la revue *Visible Language* [56]; par ailleurs, le CNRS vient d'éditer un ouvrage sur les *Manuscrits des écrivains* [36].

3. Par exemple: *Réécriture de l'écriture* et *L'écriture télématique, année zéro* des Compagnons de Lure [54] [55].

pages. Et il n'est pas non plus étonnant que certains conservateurs crient au scandale contre ces recherches, par exemple, vers 1983, celles de Roger LAUFER [178] sur les caractères animés (voir ci-après 6.3.5).

Nous allons montrer dans ce chapitre certaines possibilités des fontes dynamiques. Nous avons dit que les caractères on retrouvé leur liberté. Ils vont donc pouvoir jouer avec le hasard. Ce seront ce que nous appelons les caractères aléatoires.

Cependant, l'aspect qui nous parait le plus important est de pouvoir dessiner un caractère en fonction de quelque chose. Nous appelons ce quelque chose un « contexte ». Nous proposons alors une classification des caractères contextuels et en donnons des exemples.

6.1 Caractères aléatoires

C'est probablement avec *Un coup de dé jamais n'abolira le Hasard* de MALLARMÉ (voir [49] [55]) que la poésie a pris conscience de son aspect visuel et que beaucoup d'auteurs ont joué, depuis, avec le hasard. Les premières œuvres d'art par ordinateur y ont largement fait appel [35]: en effet, les calculs de fonctions pseudo-aléatoires étaient déjà disponibles alors que le traitement d'images n'en étaient qu'à des balbutiements inutilisables en pratique.

La figure 6.1 a ainsi été créée avec un programme équivalent à celui de la figure 6.2.

Même avec un programme aussi simpliste, l'artiste garde un grand nombre de degrés de liberté: choix de N et B mais aussi des domaines de variations de X, Y et C, choix de la fonte (qui peut aussi être tirée aléatoirement), voire même choix du germe de départ des fonctions aléatoires.

Mais ceci concerne le choix aléatoire de caractères et non le dessin d'un caractère en utilisant des contours définis aléatoirement.

6.1.1 Coordonnées aléatoires

Le principe de telles fontes est de définir aléatoirement les coordonnées des points caractéristiques d'une lettre (figure 5.1).

Donald KNUTH a dessiné de cette façon son caractère *Punk* [145], mais il l'a fait en METAFONT qui n'a pas de propriétés dynamiques. Nous avons donc converti cette fonte en PostScript et l'avons rendue vraiment dynamique, en collaboration avec Victor OSTROMOUKHOV [163]. La figure 6.3 montre quelques exemples de ces caractères.

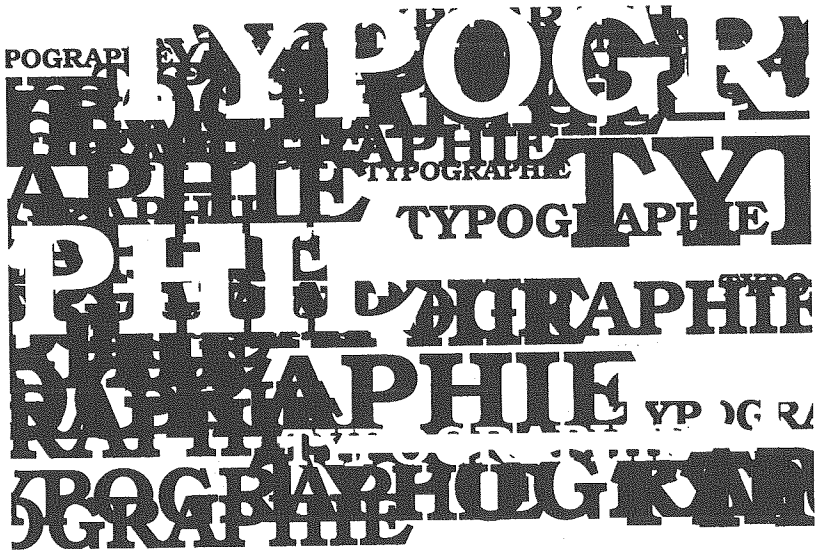


FIG. 6.1 - *Image produite par tirage aléatoire des coordonnées et tailles des caractères*

```
N := 100
B := 4
Prendre de l'encre noire
Faire N fois
      début tirer au hasard les valeurs de X, Y et C
            prendre une fonte de corps C
            imprimer le mot (TYPOGRAPHIE)
                  au point de coordonnées X,Y
      fin
Prendre de l'encre blanche
Faire B fois
      début tirer au hasard les valeurs de X, Y et C
            prendre une fonte de corps C
            imprimer le mot (TYPOGRAPHIE)
                  au point de coordonnées X,Y
      fin
```

FIG. 6.2 - *Programme réalisant l'image ci-dessus*

FIG. 6.3 - *Fonte* Punk *de* KNUTH, *rendue dynamique. Les deux appels ont été* `(Irisa Inria Rennes) show` *mais tous les caractères sont différents*

6.1.2 Variations sur les attributs typographiques

Une fonte n'est pas faite seulement d'algorithmes de dessin de caractères. Rentrent aussi en ligne de compte un certain nombre d'attributs typographiques comme la métrique des caractères (section 3.4.2).

La figure 6.4 a été composée avec la fonte JAVAL que nous venons de dessiner [164]. Cette fonte a, en fait, trois caractéristiques.

(1) Cette fonte (basée sur une première fonte aléatoire que nous avions écrite en 1989 [159]) permet de reproduire des parties de *Scrabble*; outre la forme des pièces une caractéristique de cette fonte est qu'elle est bi-directionnelle: elle écrit les caractères soit verticalement, soit horizontalement.

(2) Les caractères eux-mêmes (basés sur *Avant-Garde* d'Herb LUBALIN) ont été redessinés de façon à permettre la création de « palindromes typographiques »: on appelle ainsi un palindrome pour lequel on tient compte aussi de la symétrie planaire des lettres [48] [53] [164]. « LAVAL » est un palindrome mais n'est pas un palindrome typographique car « LA » et « AL » ne sont pas symétriques; par contre « JAVAL » (du nom du physiologue, voir 3.7.1) est un palindrome typographique à symétrie verticale, à condition de disposer d'un « J » qui ressemble à un « L » retourné. D'où la nécessité de redessiner ces lettres.

(3) Enfin, et c'est ce qui nous intéresse ici, au moment de tracer une lettre, une variable aléatoire est calculée et passée en paramètre à une fonction de rotation avant le tracé de la pièce.

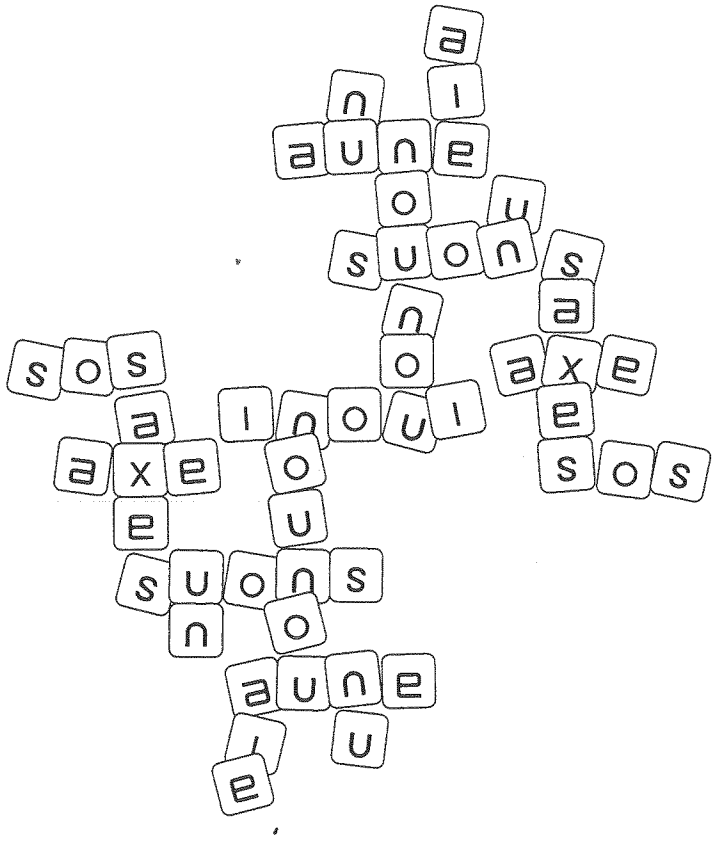


FIG. 6.4 - *Cette figure est un palindrome typographique : elle peut se lire à l'endroit ou tête-bèche ; elle a été composée en* JAVAL *qui secoue aléatoirement les pièces comme si on donnait un coup sur la table où se trouve cette partie de* Scrabble

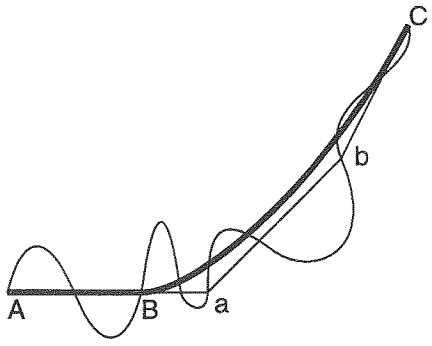


FIG. 6.5 - *Déformation du segment AB et de la courbe BC en éléments bruités*

6.1.3 Modification de contours

Une autre possibilité est de modifier non plus seulement les coordonnées des points de contrôle, mais de modifier les courbes entre ces points. La figure 6.5 montre le principe de ces déformations: ici, le segment de droite AB a été remplacé par une courbe de Bézier et la courbe de Bézier BC remplacée d'abord par son enveloppe Ba, ab et bC, chaque segment étant alors remplacé à son tour par une courbe de Bézier comme pour AB. Voici deux applications.

Simulation du foulage

Les caractères, tels qu'ils sont imprimés sur des photocomposeuses de très bonne définition, ne peuvent plus aujourd'hui prêter le flanc à la critique des typographes: leur qualité est sûrement bien meilleure que celle des caractères en plomb. Deux bonnes raisons à cela:

- les caractères en plomb étaient gravés à la main à l'aide de gouges diverses mais toujours minuscules; il était donc impossible pour l'œil humain de discerner les inévitables inégalités du tracé des formes (la figure 5.3 montre des aggrandissements d'accolades où l'on voit de telles déformations);
- lors de l'impression, l'encre débordait du caractère à cause de la pression et se répandait en fonction de la nature du support (papier, tissus, etc.): c'était le *foulage*; les anciens graveurs en tenaient compte, surtout pour les petits corps comme le rappelle Ladislas MANDEL [43].

Or, les typographes regrettent justement cette irrégularité des tracés, tout comme certains mélomanes reprochent aux instruments électroniques

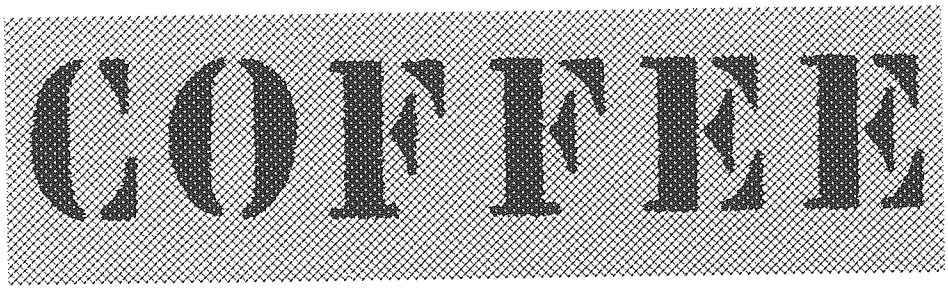


FIG. 6.6 - Pochoir *simule le foulage de l'encre, ici sur une toile de jute; à noter que chaque évocation de lettre produit une déformation originale*

l'absence de l'humain (bruits des doigts sur les cordes, souffle, etc.). Aussi, certains typographes ont-ils créé, même du temps de la photocomposition, des alphabets où ce phénomène est amplifié (il s'agit surtout de lettres dessinées à des fins plus graphiques que de caractères de labeur). C'est notamment le cas de *Chaillot*, dessiné il y a déjà quelques dizaines d'années par JACNO pour les affiches du TNP [61].

Il nous a donc semblé intéressant de simuler, aléatoirement, ces phénomènes de foulage. Dans notre version de *Punk*, nous avons dès 1989 introduit du bruit permettant de tracer notamment les gros points avec une certaine irrégularité paramétrable [163]. Ceci a depuis été introduit dans certaines fontes « antiquées » d'URW.

Nous avons repris cette possibilité de façon plus complète dans une fonte, *Pochoir*, inspirée justement de *Chaillot*: la recherche consiste essentiellement à trouver des approximations mathématiques (comme en synthèse d'image) permettant de paramétrer divers types de foulage (l'encre, ou la peinture, ne coule pas de la même façon selon sa propre nature, selon la matière du pochoir et selon la nature du support). La figure 6.6 montre quelques caractères d'une version provisoire (nous avons donné l'ossature des algorithmes dans [165]) en cours d'amélioration.

Des produits commerciaux comme *Fontoshop* permettent également de paramétrer des distorsions de contours ou de modifier la nature des aplats de caractères. Malhereusement, ce bruitage s'y fait sur une « image » entière: si on modifie le mot « COFFEE », les deux « E » et les deux « F » seront bien bruités de façon différente. Mais si on reprend ce mot « COFFEE », par exemple par *Illustrator*, toutes ses occurrences auront les mêmes déformations globales. C'est aussi, nous semble-t-il, la technique employée par Neville BRODY pour le catalogue de *Fontoshop*.

Broken Art

Sous ce nom a lieu, notamment aux États-Unis mais aussi en Europe du Nord, tout un mouvement artistique qui, comme son nom l'indique, revient à montrer la beauté de la dégradation d'objets, de tableaux et de textes. C'est ainsi que Erik VAN BLOKLAND et Just VAN ROSSUM, se référant explicitement à nos travaux dans [168], ont créé notamment une fonte, *Beowolf*, qui est très fréquemment citée (par exemple dans [175] [170] [169]) et dont on trouvera un exemplaire en figure 6.7.

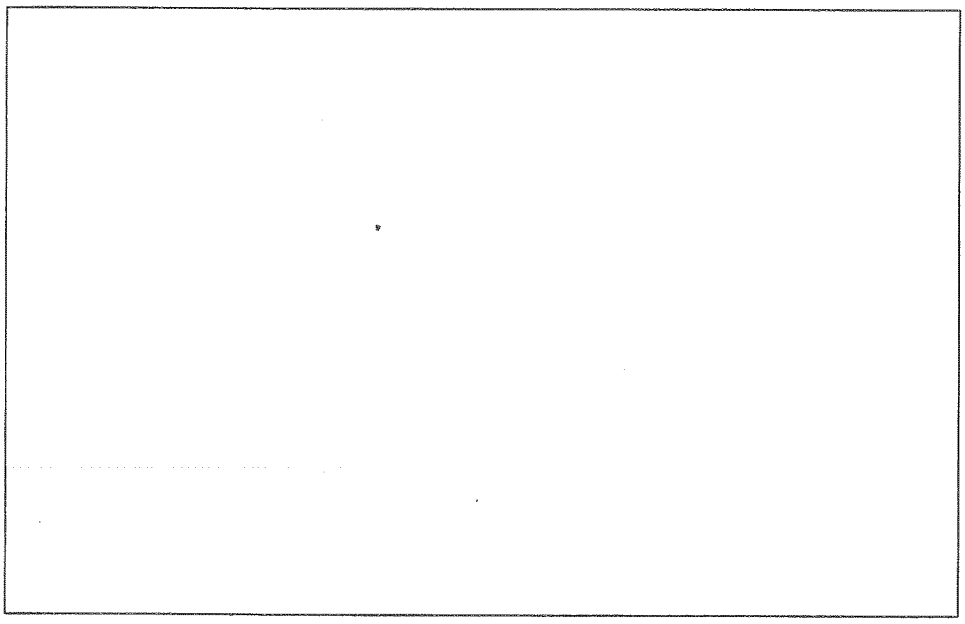


FIG. 6.7 - Beowolf, *une fonte aléatoirement dégradée de* VAN BLOKLAND *et* VAN ROSSUM, *d'après [175]*

6.1.4 Autres possibiliés

Enfin, signalons que du hasard, mais plus difficilement contrôlable, peut être introduit en utilisant des opérateurs dans un contexte de résultat indéfini, voire à la suite d'erreurs (involontaires au départ mais dont on se rend compte qu'elles donnent des résultats intéressants et reproductibles). Voici deux exemples basés sur le fait qu'un chemin en PostScript peut être complexe (ensemble connexes de chemins plus élémentaires).

- La détermination de l'intérieur ou de l'extérieur d'une surface est simple s'il y a peu de chemins qui se croisent. Elle peut devenir très

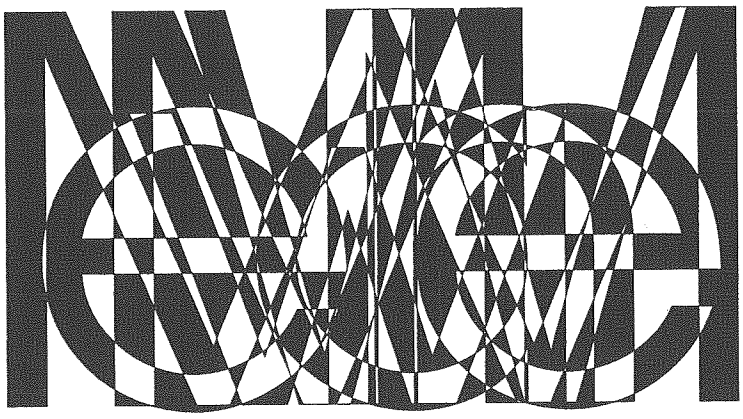


FIG. 6.8 - *Résultat « imprévisible » de* `eofill` *quand plusieurs caractères (M e M b W etc. avec des chasses quasi-nulles) se chevauchent*

complexe et donner des résultats surprenants si on superpose plusieurs contours de caractères et qu'on ne donne qu'une instruction `eofill` pour tous ces caractères ; voir figure 6.8.

- De même, l'opération `clip` de détourage d'un chemin peut aussi donner, comme en figure 6.9, des résultats curieux et esthétiques en rendant les chemins trop complexes.

Toutes ces fontes ont une caractéristique commune : si elles utilisent des fonctions aléatoires et sont donc bien dynamiques, elles sont complètement indépendantes. Elles n'ont donc pas besoin de passage de paramètres. Nous allons voir, section 6.3, ce que sont ces paramètres et décrire des fontes faisant largement appel au contexte. Mais auparavant, nous allons en décrire une qui nous servira de modèle pour les autres.

6.2 Le *Delorme*

Lorsque nous avons vu la possibilité de créer des fontes dynamiques, nous avons bien sûr voulu traiter ainsi une vraie fonte, complète. Mais n'étant pas dessinateur typographe, il a fallu que nous trouvions quelqu'un qui comprenne l'intérêt de la chose. C'est ainsi qu'il nous a été donné de rencontrer Christian DELORME et de travailler sur un caractère qu'il avait en tête. Cette collaboration a débouché sur la réalisation du *Delorme*, en fait sur deux versions : la première, développée essentiellement à l'EPFL à

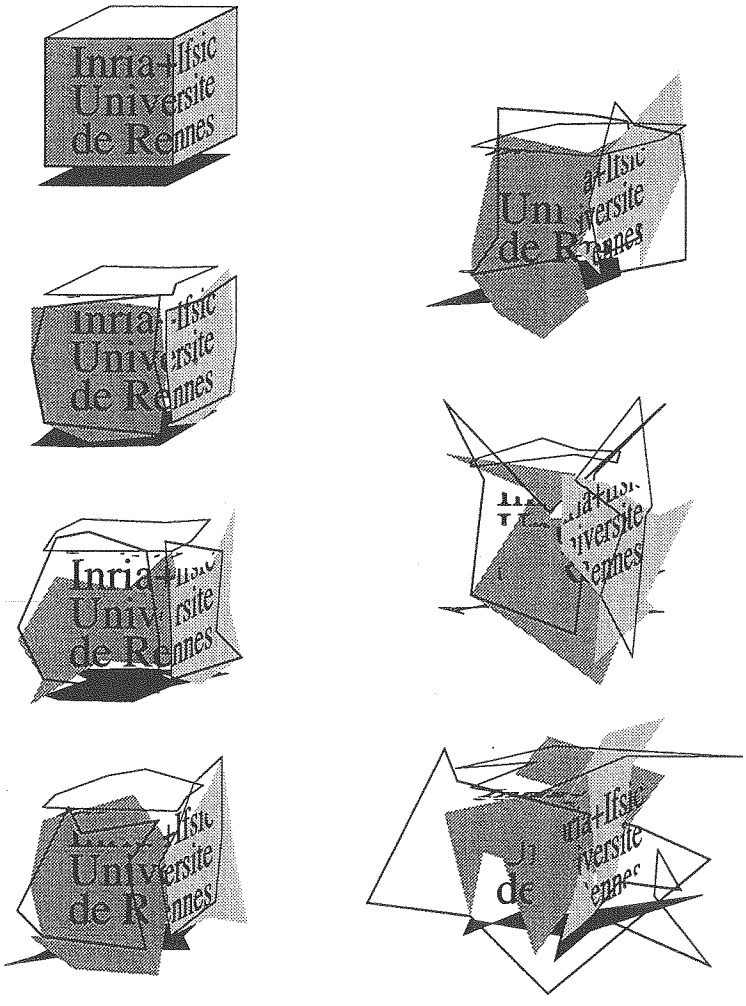


FIG. 6.9 - *Déformation d'un logo par modification aléatoire des chemins de découpe (dessin inspiré de [169])*

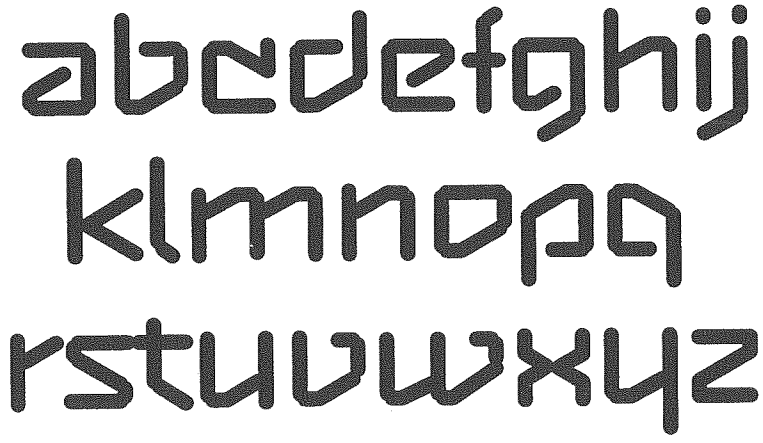


FIG. 6.10 - *Abécédaire du* Delorme

Lausanne en 1990, plutôt riche en matière de dynamisme[4] et la seconde où le graphisme a été très sensiblement amélioré. Nous parlerons indifféremment de ces deux versions sans les distinguer.

Ce caractère a été principalement publié dans *Communication et langage* [162]. Ce n'est pas un caractère de labeur et il a surtout servi a dessiner des logos (voir par exemple le livre de DELORME sur *le logo* [30]).

6.2.1 Dessin du *Delorme*

Le *Delorme* est un caractère « bâton», en fait une « linéale» (tracé d'épaisseur constante, sans empattements), conçu pour être modulaire, c'est-à-dire (aux yeux de son auteur) construit à partir d'éléments simples, en pratique des languettes rectangulaires à bouts arrondis (voir figure 6.10). La nouveauté, comparée aux nombreux alphabets bâtons (comme ceux illustrant le livre *Ma thémagie* de HOFSTADTER [207]), est que ces caractères ont à la fois une certaine épaisseur et une certaine illusion de courbe obtenue par la superposition de courts segments (voir figure 6.11).

6.2.2 Implémentation du *Delorme*

Le *Delorme* surprend en général (peut-être à cause de son allure assez carrée et un peu grasse?). Mais, ce qui nous a intéressé en lui, c'est que ce caractère est très simple sur le plan graphique: il n'y a aucune courbe

4. Mais elle péchait un peu par le graphisme des caractères et son imprécision dans les espacements.

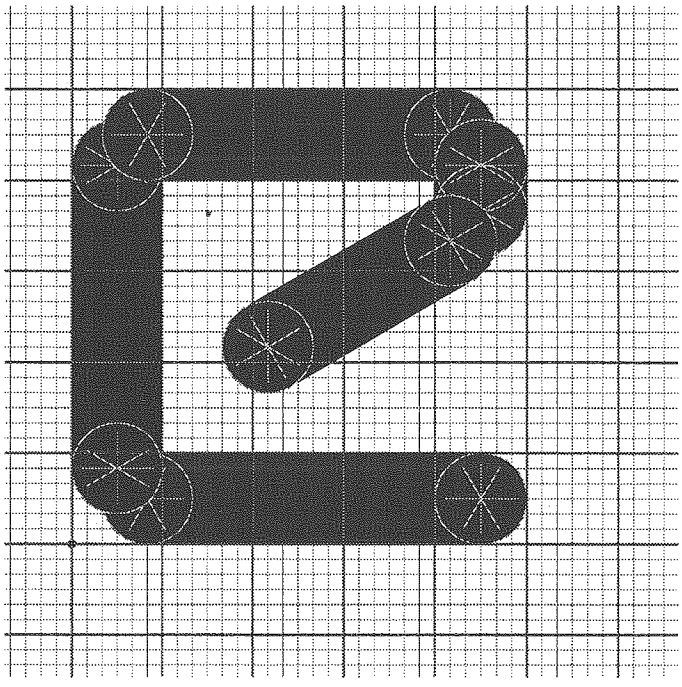


FIG. 6.11 - *Les caractères du* Delorme *sont des segments superposés donnant l'impression d'arrondi*

de Bézier, ce qui supprime tout problème de lieu géométrique (contrairement aux symboles mathématiques - voir figure5.3). Ce caractère a donc été construit comme une fonte dynamique (avec `setcharwidth`), mais il a nécessité une métrique très spéciale, les cartons du dessinateur étant en base 6 (voir les grilles de la figure 6.11) : chaque caractère entre dans un carré de $6/6 \times 6/6$, l'épaisseur des segments normaux est de 1/6, la chasse est de 1/6, etc.

Ces segments sont en fait programmés comme des *line* de PostScript, munis d'une certaine épaisseur et avec des bouts arrondis, un point étant alors un segment de longueur nulle. Toute la force du *Delorme* est d'avoir été programmé de façon analytique, comme le « L » de la figure 4.2. Chaque segment est donc fait d'instructions

```
x  y  moveto
x' y' lineto
```

mais ces valeurs dépendent, comme le montre la figure 6.12, de valeurs d (distance du centre d'un trait au bord théorique du carré 5×5 du caractère) et 2e (épaisseur de ce trait), ces deux valeurs étant liées à la graisse ; de plus, pour permettre l'italicisation du caractère, les abscisses de certains points sont multipliées par la tangente de l'angle d'italique.

Nous allons reprendre les propriétés de ce caractère dans la section suivante.

6.3 Fontes contextuelles

Les paramètres passés à une fonte dynamiques peuvent évidemment être quelconques. Néanmoins, il nous semble que l'on peut classer ces passages de paramètres en fonction du contexte où se trouve le caractère concerné.

Il y a toutefois un problème fondamental de choix : faut-il faire ce genre de chose lors de l'impression ? Certaines peuvent être faites dès la composition puisqu'elles sont décidables, par exemple le choix de lettre (finale, initiale, etc.) en fonction de la voisine (c'est par exemple ainsi que TEX choisit les « ligatures »). Nous y reviendrons en 6.4.

Mais dans une vision un peu futuriste, par exemple hypermédia ou calligraphie animée, cette séparation composition/impression peut ne plus avoir de sens. Il est donc, de toute façon, intéressant de travailler dans cet esprit.

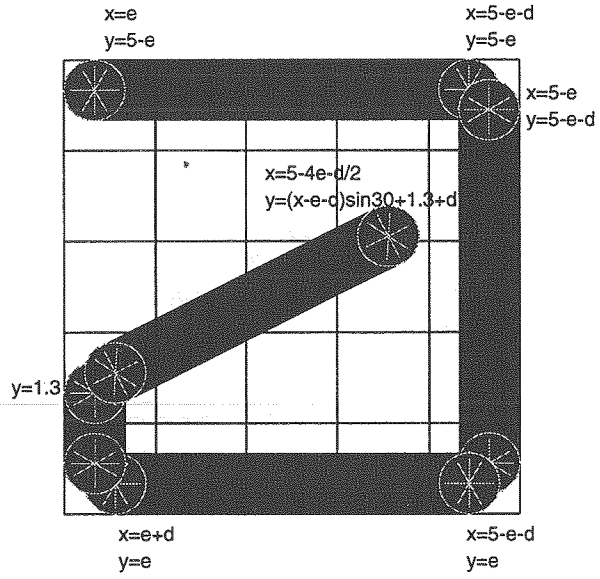


FIG. 6.12 - *Les coordonnées des extrémités des segments du* Delorme *sont analytiques. Ici, pour ce « a maigre », g = la « graisse », c'est-à-dire l'épaisseur des traits, d = g/3 et e = g/2.*

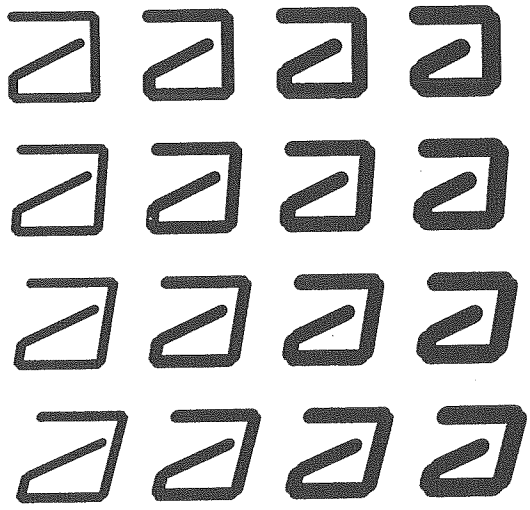


FIG. 6.13 - *Tous ces caractères en* Delorme *ont été imprimés par la même instruction* (a) show *mais après avoir passé en paramètre l'angle d'italique et l'épaisseur du trait*

6.3.1 Contexte typographique local

Les coordonnées analytiques des points des segments (dans le cas du *Delorme*) et de façon plus générale des courbes de Bézier peuvent être paramétrées. C'est ce que nous avons fait pour les formules mathématiques. Le *Delorme* se prête bien sûr particulièrement à cela (figures 6.12 et 6.13).

6.3.2 Contexte de voisinage

Lorsque l'on imprime un caractère, il suffit de quelque automate à mémoire pour pouvoir connaître le caractère que l'on vient de tracer, ou le suivant. Les applications dynamiques de ceci sont fondamentales car, en fait, pratiquement toute la typographie repose sur ces problèmes de voisinage (voir 3.7). Voici des exemples, tirés du *Delorme*. De telles opérations sont possibles ici car nous avons au départ la volonté de les faire (ce n'est pas le cas par exemple du *Times* qui a été dessiné il y a près d'un siècle - il faut alors découvrir ses propriétés topologiques comme le fait BETRISEY [125]) et d'autre part nous ne travaillons qu'avec des segments de droite. Mais ça a le gros avantage d'avoir été possible dès 1989.

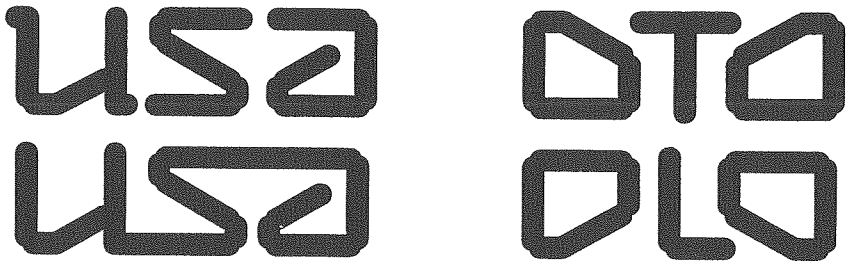


FIG. 6.14 - *Calcul automatique de ligatures et choix de caractères en* Delorme

a) choix de lettres et ligatures automatiques

Lorsque l'on imprime un caractère, on connaît ses voisins. Dans le *Delorme*, nous avons associé à chaque caractère une table topologique signalant la présence d'une barre verticale à gauche (par exemple pour un « p ») ou à droite (« q »), la présence d'un creux en bas à gauche (« v »), d'une pointe en haut à droite (« T »), etc. Un tableau de valeurs booléennes en petit nombre suffit alors à décrire chaque caractère et des opérations logiques permettent de faire des choix.

La figure 6.14 montre en haut à gauche trois lettres « usa » normales et en dessous les mêmes en mode dynamique : la ligature « us » a été calculée automatiquement par la présence d'un trait horizontal en bas à droite du « u » face à celui en bas à gauche du « s » ; de même entre « s » et « a ».

La figure 6.14 montre à droite comment le crénage entre « O » et « T » ou « L » est réglé par rotation du « O ».

On trouvera dans [162] d'autres exemples.

b) espacement entre lettres

Ces mêmes propriétés permettent de calculer automatiquement l'espacement entre caractères. Il n'y a donc plus de table de chasse ni de table de crénage, mais une fonction de calcul de l'espace entre les « œils » (la surface imprimée) de chaque caractère basée sur leur topologie. La figure 6.15 montre quelques unes de ces approches « à la volée ».

6.3.3 Environnement géométrique

Ces calculs d'approche pourraient être faits dès le formatage. Par contre, il est des cas où, comme pour les caractères mathématiques, il est difficile de prévoir une fonte avec toutes les possibilités de variations de forme

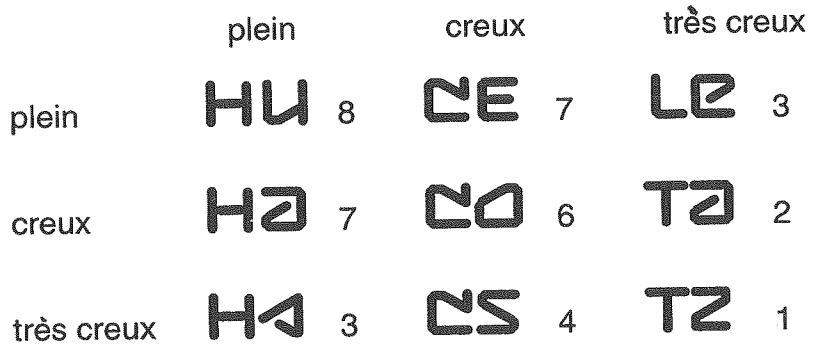


FIG. 6.15 - *Table de calcul d'espace entre caractères. Chaque case de la matrice donne, en fonction de la forme du côté droit du caractère de gauche (en ordonnée) et de celle de la forme du côté gauche du caractère de droite (en abscisse) un exemple (par exemple «Hu») et la valeur de l'espace à mettre entre ces deux signes (ici 8 points* Delorme*).*

géométrique. C'est le cas des caractères arabes ou hébreux que l'on allonge pour justifier un texte. Mais en général, ceci est fait par allongement d'une partie rectiligne du caractère. Nous avons montré [157] que l'on pouvait ainsi dessiner des caractères arabes dépendant de leur boîte exacte au moment de l'impression tout en gardant leur courbure. La figure 6.16 montre une lettre arabe réelle dessinée par Dan BERRY.

6.3.4 Autres contextes

Divers autres contextes peuvent être imaginés. Par exemple, Pierre DI SCIULO dessine des caractères en fonction de leur prononciation (voir figure 6.17).

Signalons aussi un essai amusant où l'auteur remplace dynamiquement chaque parenthèse d'une expression Lisp par une parenthèse marquée en fonction de son degré d'imbrication (il utilise donc une pile pour faire ceci) ; voir figure 6.18.

6.3.5 Calligraphie animée

Il y a quelques années, Roger LAUFER avait essayé de faire de la « calligraphie animée par ordinateur » [177] [178], mais ses recherches n'avaient pas vraiment abouti, logiciels et matériels ne s'y prêtant pas à l'époque.

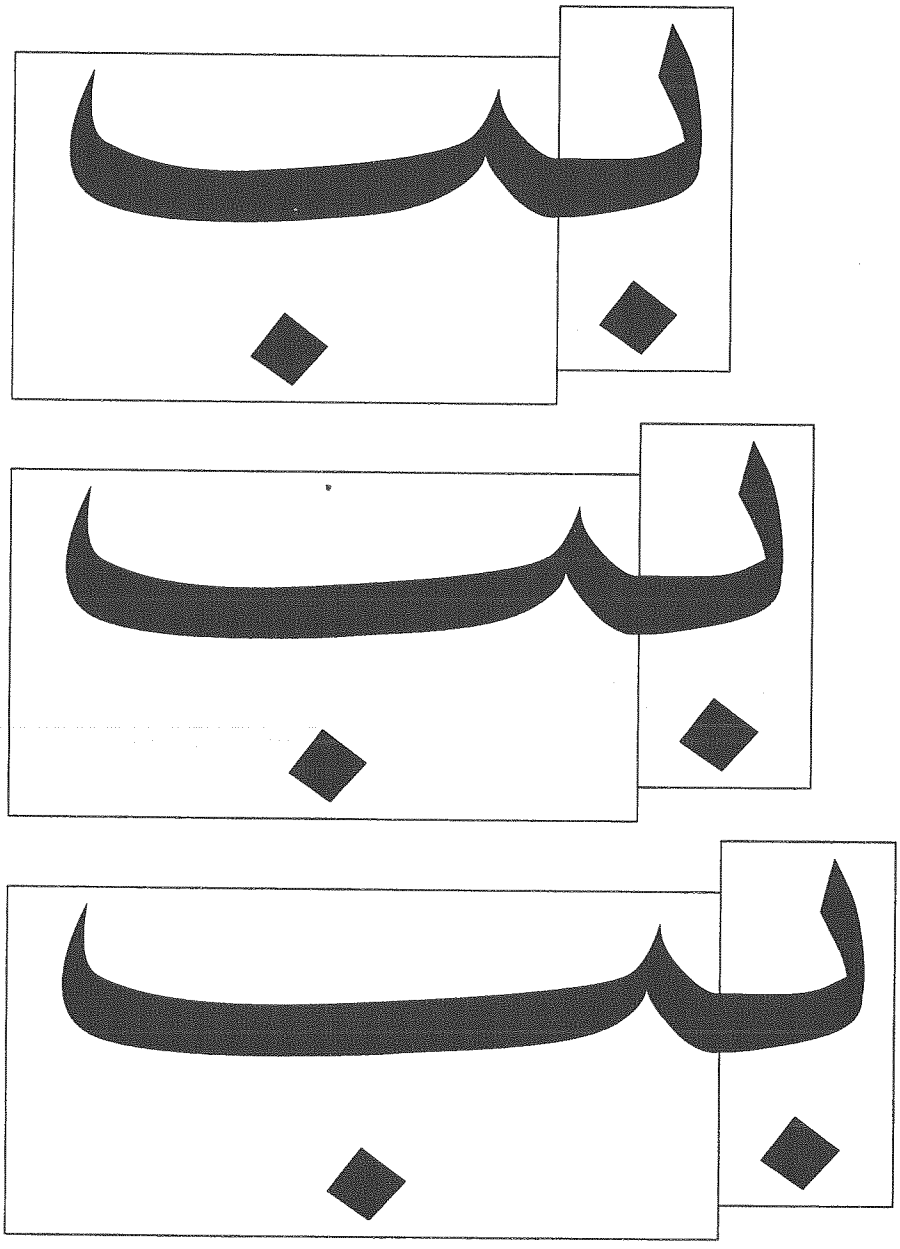


FIG. 6.16 - *Caractères arabes calculés au moment de l'impression de façon qu'ils entrent dans la boîte voulue tout en gardant leur courbure. Dessin de Dan* BERRY *[167]*

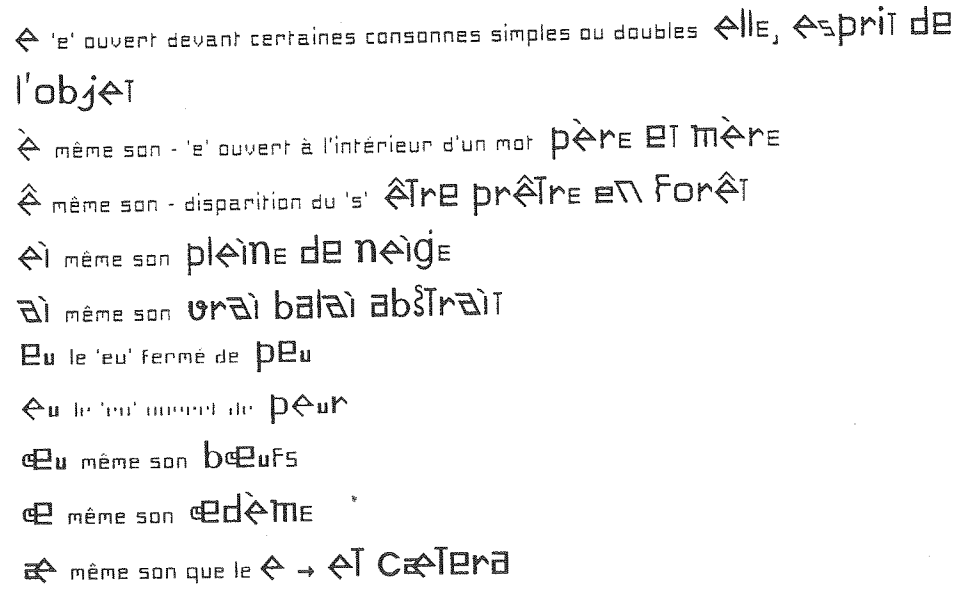


FIG. 6.17 - *Caractères dépendant de leur prononciation - par* DI SCIULO *[171]*

```
(DEFUN ANY (LST)
 (COND ((NULL LST) NIL)
       ((CAR LST) T)
       (T (ANY (CDR LST))) ) )
```

FIG. 6.18 - *Les parenthèses de ce programme sont marquées dynamiquement en fonction de leur niveau d'emboitage - par Michael* COHEN *[172]*

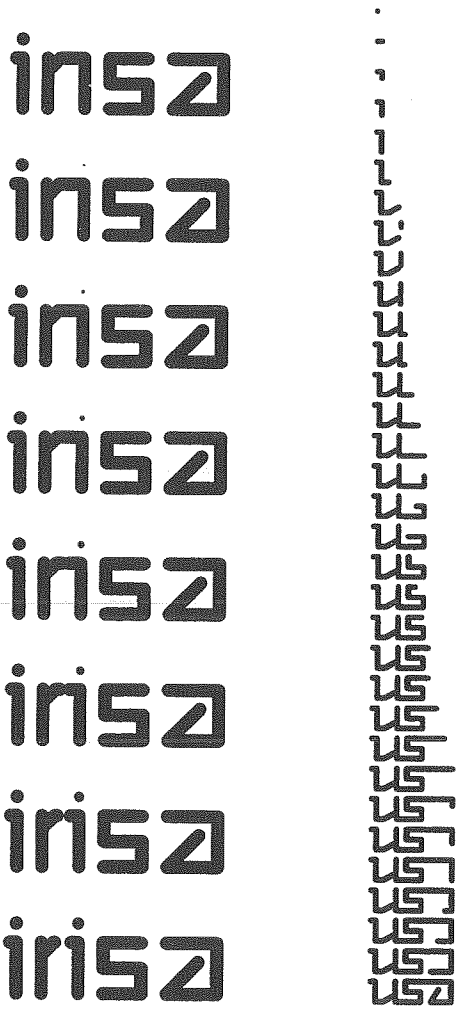


FIG. 6.19 - *Résultat d'appels à des temps successifs de* `(insa) show` *et de* `(usa) show`

Aujourd'hui, le traitement d'image dispose de moyens ultra-rapides de calculs et on devrait pouvoir y faire des calculs de fontes dynamiques où chaque caractère verrait ses coordonnées de points de contrôles calculés en fonction du temps. La figure 6.19-gauche montre une déformation d'un caractère avec $\delta x = t$ appliquée ici avec une boucle comme :

```
pour t = 0, 5, 30 faire
    début
      delta x := t
      (insa) show
    fin
```

tandis que la figure de droite simule le suivi du ductus d'un caractère. Cet exemple est fait avec un caractère à la *Delorme*. Faire de même pour des caractères calligraphiques réels nécessite d'une part de faire varier l'épaisseur des courbes (KNUTH utilisait dans sa première version de METAFONT cette notion de ductus ; certains chercheurs, par exemple KLASSEN [176] étudient la possibilité de définir des *splines* d'épaisseur variable) et d'autre part de suivre ce ductus dans le temps (donc de n'utiliser à un instant donné qu'une partie de *spline*). Il nous semble d'ailleurs que cette déformation du « n » de « insa » en « ri » relève du même esprit que le passage du « e » *Garamond* corps 8 au « e » corps 36 (figure 4.6), c'est-à-dire du dessin animé. On comparera ces figures à celles de SEDERBERG et GREENWOOD [188].

Il y a donc là toute une voie de recherches à faire. Les applications de telles fontes sont évidentes en télévision numérique et dans le domaine plus général des multimédia.

6.4 Comparaison des différents mécanismes

Comparons, pour terminer, divers mécanismes de génération de fontes, en l'occurrence PostScript (avec ou sans mécanisme de cache), METAFONT et les *Master fonts* d'Adobe [156] [179]. Il en existe d'autres, comme *MultiType* d'URW [141], mais ils ressemblent à ceux-ci.

(1) METAFONT travaille avant le moment de l'impression : il reçoit les descriptions des contours (en général sous forme analytique), il calcule les plans de bits qui sont alors chargés lors de l'impression. Les valeurs des variables sont donc évaluées une fois pour toutes, lors du calcul du plan de bits.

(2) PostScript, lorsque le mécanisme de cache est activé, travaille lors de l'impression. Il reçoit les descriptions des contours. Lors de la première instanciation d'un caractère, il calcule le plan de bits correspondant et le met en mémoire cache. Un nouvel appel de ce caractère revient à réutiliser ce plan de bit, sans aucune révaluation des variables éventuelles.

(3) *Multiple Masters* travaille en deux fois : d'abord un *set up* est traité, permettant à partir d'une forme analytique (par exemple la description du GARAMOND de la figure4.6), de créer toutes les fontes PostScript ordinaires prévues (donc les divers forces de corps dans diverses graisses). Notons au passage que ceci prend beaucoup de place en mémoire... Ensuite, chacune de ces fontes est utilisée comme une fonte ordinaire. Là encore, les variables analtyiques ne sont évaluées qu'une fois (lors du *set up*).

(4) PostScript, avec le mécanisme de cache désactivé, travaille au moment de l'impression. Mais chaque plan de bits est réévalué à chaque instanciation d'un caractère : les variables analytiques sont donc recalculées à chaque appel. PostScript offre donc ainsi les trois mécanismes de passage de paramètres des langages de programmation (par valeur, par référence et par nom).

Mais on ne peut pas dire qu'une méthode est supérieure à une autre : tout dépend de ce que l'on veut faire. La méthode de METAFONT est très bonne pour du texte qui ne subit pas de rotations et qui sort sur du matériel constant (il suffit alors de calculer une fois pour toutes les plans de bits). PostScript sans mécanisme de cache permet toutes les créations. Entre les deux, on a beaucoup de choix.

Conclusion

NOUS AVONS montré dans ce mémoire que la typographie numérique est un domaine complexe, utilisant des algorithmes compliqués et devant gérer ces grandes bases de données de procédures que sont les fontes. Nous avons aussi montré que la typographie était toujours à la recherche d'une meilleure qualité pour le confort du lecteur. Nous avons enfin montré que, au lieu de regarder le passé avec regret, il fallait au contraire que les typographes regardent l'avenir avec un grand espoir car ils disposent désormais de nouveaux concepts offrant des possibilités de création originale.

Nous voudrions, en guise de conclusion, montrer les points qui, à notre avis, devraient faire l'objet de recherches qui pourraient être proposées à des chercheurs dans les quelques années à venir.

Mathématiques et typographie

Les études menées depuis quelques années sur la représentation des caractères à l'aide de splines et le remplissage des plans de bits relèvent désormais du milieu industriel. Mais ça ne veut pas dire qu'il ne faille pas aller plus loin. En particulier, il y a encore beaucoup à faire pour l'étude des ligatures (au sens de liaisons) entre caractères, manuscrits notamment, pour lesquelles il semble que des quintites soient recommandables [185]. De même, la notion de ductus, initialement présente dans METAFONT, puis abandonnée, demande l'étude des variations d'épaisseur dans les traits [176]. Combiné avec des algorithmes de *shape blending* [188] et des études sur la vélocité, ceci devrait permettre l'étude du tracé de caractères dans le temps (comme nous l'avons ébauché dans notre figure 6.19), indispensable si l'on veut vraiment faire du multimédia ou de la calligraphie animée. Le passage du plomb aux fontes dynamiques est équivalent à celui du *volumen* à l'hypertexte.

Un autre aspect des mathématiques utilisables en typographie est la statistique. Nous manquons, comme nous l'avons dit (section 4.2.2), de données quantifiées (et non purement subjectives) sur les caractères. À part celles de Peter KAROW [113], on ne voit pas beaucoup de publications à citer ! Toutefois, nous avons commencé des analyses factorielles sur les données des AFM en collaboration avec Brigitte ESCOFIER à l'Irisa et qu'il faudrait compléter avec des études sur les caractères eux-mêmes. Une première finalité est de remplacer ces vieilles classifications typographiques qui n'ont

plus aucun sens aujourd'hui et permettre de remplacer une fonte par une autre *similar to.*

Informatique et typographie

Si la typographie numérique a longtemps été l'objet de travaux de spécialistes des images 2D, elle devient maintenant une affaire d'informaticiens (en incluant dans le mot « informatique » les sens « intelligence artificielle », « bases de connaissance », etc.).

- Bien qu'à la limite du sujet de notre étude, la création de caractères, nous avons fait allusion à la notion de serveur de fonte et d'AFM (section 3.4.2). De nouveaux schémas globaux d'accès aux fontes, comme *NSFF - New Font Selection Scheme* lié à TEX [68, pages 157-214], ont été proposés. Mais ils restent encore limités aux anciennes notions de fontes statiques. Il faut dire que la notion de fonte dynamique est indissociable de la notion de passage de paramètre à une procédure. Il faudrait que les langages de description de page, comme PostScript, aient ce mécanisme de façon plus implicite avec bien sûr les propriétés d'héritage et d'encapsulation des langages orientés objets [155].

- La notion de fonte paramétrée permet de résoudre les problèmes d'ajustement optique (voir section 4.2). Mais encore faut-il savoir comment faire varier ces paramètres. C'est ici où la notion de modèle de fonte prend une importance toute nouvelle et où les recherches sont les plus urgentes. Déjà, plusieurs modèles ont été proposés, touchant d'ailleurs des aspects divers. Pour les uns, comme celui de HERSCH-BETRISEY [137] [125], il s'agit de définir la topologie des caractères ; pour d'autres, HERSCH-HERTZ [98, pages 261-272] la finalité est de décrire le « cheminement » le long d'un caractère (ici en utilisant des techniques de *string matching*, montrant encore le rapport entre typographie et langages formels). Martin DÜRST, de son côté [98, pages 133-143] utilise Prolog pour la description abstraite des éléments de caractères Kanji.

  Il ne s'agit plus, à présent, de croire à la possibilité de définir des caractères comme étant composés d'éléments simplement juxtaposés : depuis la thèse de COUEIGNOUX en 1975 [105], diverses recherches ont été faites pour dessiner des caractères à partir de morceaux plus élémentaires, par exemple par Debra ADAMS au Parc [24] ou Marc NANARD à Montpellier [152] ; outre certaines difficultés techniques (jonction des éléments entre eux), il semble que ces méthodes ne puissent vraiment être utilisées pour créer de vrais caractères. Les modèles en cours, par contre, laissent un grand espoir.

Lisibilité

Une des applications des modèles en cours de définition est sûrement de proposer un cadre à ces études très nombreuses mais cloisonnées qui relèvent de domaines aussi variés que la reconnaissance des caractères, les études physiologiques sur la vision, les mathématiques de la vision, etc. (voir section 3.7.1). Il en est effet frappant de voir combien ces études ignorent souvent la typographie...

Création de caractère et activité humaine

La notion de caractère et par là celle de création de caractère commencent à intéresser les philosophes et non plus les seuls graphistes ou sémiologues. À un niveau plus macroscopique, nous voyons un peu la même chose en édition électronique : un document, est-ce l'image que l'on voit sur un écran ou sur le papier (et alors on peut utiliser des produits *Wysiwyg*), ou bien est-ce la façon d'obtenir cette image (et alors il faut utiliser un éditeur structuré, voire un hypertexte)? Cette image est-elle unique (notion de généricité)? ou finalement a-t-elle quelque importance (les hypertextes ont tendance à faire croire que non !)?

On retrouve donc un peu la même chose en typographie microscopique et des auteurs se posent des questions sur les relations entre la forme d'une lettre et cette lettre (notamment Richard SOUTHALL [57], NOODZIJ [46]), voire sur la notion même de lettre ou de glyphe (KNUTH [142], HOFSTADTER [207]), etc.

Il ne s'agit pas de tomber dans une techno-philosophie pseudo scientifique, mais, pour nous informaticiens, d'orienter les outils d'aide à la conception dans le bon sens (doit-on, par exemple, penser qu'il suffit de méthodes cognitivistes, comme le fait MACGRAW à Indiana [147], pour « créer » des caractères? ou, comme le laissent penser les outils du Macintosh tels que *FontoGrapher* ou *FontoSHop*, qu'une lettre se réduit à quelques courbes de Bézier?).

POUR CONCLURE, disons que nous avons été surpris en entrant dans ce domaine de la typographie numérique de trouver un monde très riche et aux aspects très variés puisque faisant aussi bien appel aux mathématiques qu'au sens esthétique et à l'informatique. Nous retrouvons donc, plusieurs centaines d'années plus tard, ce que disait FOURNIER et que nous avons cité dès notre page 2 : *Il n'y a que celui qui réunit la science de ces parties que l'on puisse appeler un typographe.* En même temps, ce qui est frappant c'est le côté très ouvert de ce domaine et il est finalement rassurant de voir que la recherche débouche sur de nouvelles recherches.

Page vide

Bibliographie

Les ouvrages et articles sont cités dans l'ordre suivant:

(1) Typographie - généralités, histoire

(2) Typographie - notes techniques, description de fontes

(3) Édition électronique, systèmes de manipulation de documents

(4) Langages de description de page, PostScript

(5) Généralités sur la typographie numérique
Introductions, synthèses, actes de colloques

(6) Notes techniques liées à la typographie numérique

(7) Fontes dynamiques

(8) Courbes et mathématiques

(9) Lisibilité

(10) Normes

(11) Divers

et, dans chaque classe, les références sont triées par ordre alphabétique du premier auteur puis par ordre chronologique.

Les publications citées dans cette bibliographie ne sont pas toutes référencées depuis le texte: celles qui ne le sont pas peuvent en effet être de quelque intérêt dans le domaine considéré.

A.1 Typographie : généralités, histoire

[1] Pierre ABRIOUX, *Histoire de l'alphabet*, chez l'auteur, Paris, 1975.

[2] Maurice AUDIN, *Histoire de l'imprimerie*, A. et J. Picard, Paris, 1972.

[3] Fernand BAUDIN, *La typographie au tableau noir*, éditions Retz, Paris, 1984.

[4] Guy BECHTEL, *Gutenberg et l'invention de l'imprimerie - une enquête*, Fayard, 1992.

[5] Gérard BLANCHARD, *Pour une sémiologie de la typographie*, thèse présentée à l'École pratique des hautes études en sciences sociales, Paris, 1980. {Les planches ont été publiées, sous le même titre, par les Rencontres de Lure et l'École des Beaux-Arts de Besançon chez Rémy Magermans, éditeur à Andenne, Belgique, 1979. La seule version disponible commercialement en est la traduction italienne : *L'eredita Gutenberg*, Gianfranco Altieri Editore, 1989.}

[6] Roger CHARTIER et Henri-Jean MARTIN (sous la direction de), *Histoire de l'édition française*, Fayard, 1990.

[7] John DREYFUS et François RICHAUDEAU (sous la direction de), *La chose imprimée*, Retz, 1977.

[8] Jean-Claude FAUDOUAS, *Dictionnaire des grands noms de la chose imprimée*, Retz, 1991.

[9] Lucien FEBVRE et Henri-Jean MARTIN, *L'apparition du livre*, coll. L'évolution de l'humanité, Albin Michel, 1971.

[10] FOURNIER le jeune (Pierre, dit) *Manuel typographique utile aux gens de letrres & à ceux qui exercent les différentes partie de l'Art de l'Imprimerie*, Paris MDCCLXIV, 2 tomes.

[11] A. FREY, *Nouveau manuel complet de typographie contenant les principes théoriques et pratiques de cet art*, Manuels Roret, Paris 1857. Édition fac-similée offset, chez Leonce Laget, Paris, 1979.

[12] *De plomb, d'encre et de lumière*, ouvrage collectif, Imprimerie nationale, Paris, 1982.

[13] *Les caractères de l'Imprimerie nationale*, Imprimerie nationale Éditions, Paris, 1990.

[14] Georges JEAN, *L'écriture, mémoire des hommes*, coll. Découvertes, Gallimard, 1987.

[15] Jacques JOURQUIN, *Gutenberg: de l'or au plomb*, Jacques Damase, Paris, 1988.

[16] Ruari MCLEAN, *The Thames and Hudson Manual of Typography*, Thames and Hudson, Londres, 1980.

[17] MAC LUHAN *La Galaxie Gutenberg*, Mame, 1967.

[18] Alan MARSHALL, *Ruptures et continuités dans un changement de système technique - le remplacement du plomb par la lumière dans la composition typographique*, thèse, Grenoble, 18 décembre 1991. Parue comme *Publication interne Irisa*, n° 638, mars 1992.

[19] Ph. MINARD, *Typographie des lumières*, Champ Vallon, 1989.

[20] Jérôme PEIGNOT, *De l'écriture à la typographie*, Gallimard, 1967.

[21] René PONOT, « De l'influence de la technique », [12].

[22] René PONOT, « Caractères d'imprimerie », art. de *Les sciences de l'écrit*, sous la direction de Robert Estival, éd. Retz, Paris, 1993, 103–106.

[23] Walter TRACY, *Letters of Credit - A view of type design*, Gordon Fraser, Londres, 1986.

A.2 Typographie traditionnelle et art : notes techniques, description de fontes, recherches

[24] D. ADAMS, « abcdefg, a better constraint driven environment for font generation », in [99, pages 54–70].

[25] Debra ADAMS and Richard SOUTHALL, « Problems of quality assessment », in [99, pages 213–222].

[26] Jacques ANDRÉ, « Métrique des fontes en typographie traditionnelle », *Cahiers GUTenberg*, n° 4, décembre 1989, 9–21.

[27] Fernand BAUDIN, « How to use ready-made alphabets », in [109, pages 165–196], 1993.

[28] Joseph BEAUNE et René PONOT, *Qui a ramassé la plume d'oie?*, Dessain et Dalra, Paris, 1979.

[29] Charles BIGELOW and Lynn RUGGLES (eds.), *The Computer and the Hand in Type Design*, numéro spécial de *Visible Language*, XIX, 1, hiver 1985.

[30] Christian DELORME, *Le logo*, Éditions d'organisation, Paris, 1992, seconde édition.

[31] Henk DROST, « Punch Cutting Demonstration », *in* [29, pages 99–105].

[32] Adrien FRUTIGER, *Type Signe Symbol*, ABC Zurich, 1981 (édition trilingue).

[33] Adrien FRUTIGER, « Typography with the IBM Selectric Composer », *Journal of Typographic Research*, vol. 1, n° 3, 285–292.

[34] Raymond GID, *Célébration de la lettre*, Morel, 1962 ; nouvelle édition : Fata Morgana, 1992.

[35] Antonio GUZMAN, avec la collaboration de J. ANDRÉ, *Ars+Machina I*, Catalogue de l'exposition, Maison de la culture de Rennes, 1980.

[36] *Les manuscrits des écrivains*, sous la direction de Louis HEY, éditions du CNRS et Hachette, 1993.

[37] HOTLHUSEN & POL, *Scangraphic Digital Type Collection*, Mannesmann Scangraphic, Wedel, 1990.

[38] Marcel JACNO, *Anatomie de la lettre*, éditions CFE, École Estienne, Paris, 1978.

[39] Mark JAMRA, « Some elements of proportion and optical image support in a typeface », in [109, pages 47–55], 1993.

[40] David KINDERSLEY, *Optical letter spacing for new printing systems*, The Wynyn de Worde Society, Lund Humphries Publishers Ltd, 1986.

[41] *Le texte en mouvement*, textes réunis par Roger LAUFER, Presses Universitaires de Vincennes, 1987.

[42] *Le texte et son inscription*, textes réunis par Roger LAUFER, Éditions du CNRS, Paris 1989.

[43] Ladislas MANDEL, « L'écriture typographique: vers une prise de conscience », *Communication et langages*, n° 77, 1988, 5–30. Voir aussi *EPODD*, vol. 6, n° 1, 1993.

[44] James MOSLEY, *Les Académiciens et la typographie moderne*, Musée de l'imprimerie et de la banque, à paraître, Lyon 1994.

[45] Stan NELSON, « Mould Making, Matrix Fitting, and Hand Casting », *in* [29, pages 106–121].

[46] Gerrit NOORDZIJ, « The shape of the stroke », in [114, pages 34–42], 1991 ; voir aussi *Communciation et langages*, n° 91, 1992.

[47] Michel OLYFF, École de la Cambre, Belgique.

[48] OULIPO, *Atlas de littérature potentielle*, Gallimard, folio/essais, n° 109, 1981.

[49] Tibor PAPP, « De la page mallarméenne à l'écran poétique », in [42, pages 193–206].

[50] José M. PARRAMÓN, *Comment dessiner lettres et logotypes*, Bordas, Paris, 1991.

[51] *Catalogue de la Fonderie Générale*, Musée de l'Imprimerie et de la Banque, Lyon, 1926.

[52] *Rapport, au ministre d'état de l'éducation nationale et de la culture, de la mission sur l'écriture, la calligraphie et la typographie* par Jérôme PEIGNOT et Anne-Marie CHRISTIN avec la collaboration de Jacques ANDRÉ et Béatrice FRAENKEL, janvier 1993.

[53] Jérome PEIGNOT, *Typoésie*, Imprimerie nationale Éditions, septembre 1993.

[54] Rencontres de Lure, *Réécriture de l'écriture*, 1982.

[55] Rencontres de Lure, *L'écriture télématique, année zéro*, Cahiers de Lure, 1985.

[56] « The Artist's Book: The Text and Its Rival », numéro spécial de la revue *Visible Language* préparé par Renée RIESE-HUBERT, vol. 25 n° 2/3, 1991.

[57] Richard SOUTHALL, « Character Description Techniques in Type Manufacture », in [114, pages 16–27], 1991.

[58] Isaac TAMARI, « Decipherability, Legibility and Readibility of Modern Hebrew Typefaces », in [99, pages 128–136].

[59] Walter TRACY, « The Point », *The Penrose Annual*, vol. 55, pages 63–70, 1961.

[60] Michael TWYMAN *et alii*, *Making type by hand*, film vidéo, Typographic & Graphic Communication Department, Université de Reading, Angleterre, 1993.

[61] Henri VEYRIER, *Jacno*, Musée de l'affiche et de la publicité, Paris, 1989.

[62] Dr WILD, « La typographie de GUTENBERG », *Cahiers GUTenberg*, n° 18, mai 1994, 5–21.

A.3 Édition électronique, systèmes de manipulation de documents, hypertextes

[63] Jacques ANDRÉ, « Langages de publication assistée par ordinateur », art. du *Traité d'informatique - Techniques de l'ingénieur*, H2440, 12 pages, 1989.

[64] J. ANDRÉ, R. FURUTA & V. QUINT, *Structured Documents*, Cambridge University Press, 1989.

[65] J. ANDRÉ, D. DECOUCHANT, H. RICHY et V. QUINT, « Vers un atelier éditorial pour les documents structurés », *Actes du congrès Afcet 1993*, vol. 4, 63–72. Aussi, *Publication interne Irisa* n° 715, mars 1993.

[66] R. FURUTA, V. QUINT, and J. ANDRÉ, « Interactively Editing Structured Documents », *EPODD, Electronic Publishing - Origination, Dissemination and Design*, vol. 1, n° 1, April 1988, 19–44.

[67] R. FURUTA (ed.), *EP90*, Cambridge University Press, 1990.

[68] Michel GOOSSENS, Frank MITTELBACH and Alexander SAMARIN, *The LaTeX Companion*, Addison-Wesley, Reading (USA), 1994.

[69] B.W. KERNIGHAN & L. L. CHERRY, « A system for Typesetting Mathematics », *Communications of the ACM*, 18, 151–157, 1975.

[70] D. KNUTH, « Tau Epsilon Chi, a system for technical text », *Stanford Computer Science report* number STAN-CS-78-675, september 1978. Now appears as [71].

[71] D. KNUTH, *The TeXbook*, Addison-Wesley: Reading, 1984.

[72] Donald E. KNUTH, « The new versions of TeX and METAFONT», *TUGboat*, vol. 10, n° 3, novembre 1989, 325–327. Traduit en français : « TeX 3.0 ou le TeX nouveau va arriver », *Cahiers GUTenberg*, n° 4, décembre 1989.

[73] Leslie LAMPORT, *LaTeX, A Document Preparation System, user's guide & reference manual*, Addison-Wesley, 1985.

[74] R. LAUFER et Domenico SCAVETTA, *Texte, hypertexte, hypermédia*, coll. Que sais-je?, n° 2629, PUF, 1992.

[75] G. LORETTE (ed.), *Proceedings of the 1st International Conference on Document Analysis and Recognition*, AFCET publ., Paris-Rennes, 1991.

[76] V. QUINT and I. VATTON, « Making structured documents active », *EPODD - Electronic Publishing-Origination, Dissemination and Design*, 1993, à paraître.

[77] V. QUINT, « Editing Mathematics on the Buroviseur », in N. NAFFAH (ed.), *Office Information Systems*, North-Holland pub., 1982, 149–159.

[78] V. QUINT & I. VATTON, « Grif: an Interactive System for Structured Document Manipulation », [82], 1986, 200–213.

[79] V. QUINT, I. VATTON, J. ANDRÉ & H. RICHY, « Grif et l'édition de documents structurés : nouveaux développements », *Cahiers GUTenberg*, 9, juillet 1991, 49–65.

[80] V. QUINT & C. HÜSER (eds.), *Proceedings of the EP'94 conference, EPODD*, vol. 6, n° 4, 1993, à paraître.

[81] A. RIZK, N. STREITZ & J. ANDRÉ (eds.), *Hypertext: concepts, systems and applications*, Cambridge University Press, 1990.

[82] H. VAN VLIET (ed.), *Text processing and document manipulation*, Cambridge University Press, 1986.

[83] H. VAN VLIET (ed.), *Document manipulation and Typography*, Cambridge University Press, 1988.

[84] C. VANOIRBEEK & G. CORAY (eds.), *EP92*, Cambridge University Press, 1992.

A.4 Langages de description de page, PostScript

[85] Adobe Systems Incorporated, *PostScript Language Reference Manual*, first edition, Addison-Wesley: Reading, 1985.

[86] Adobe Systems Incorporated, *PostScript Language Reference Manual*, second edition, Addison-Wesley: Reading, 1991. Paru en français : *Manuel de référence du langage PostScript*, traduction de Denys BONDEVILLE, Addison-Wesley France, Paris, 1992.

[87] Adobe Systems Incorporated, *Adobe Type 1 Font Format*, Addison-Wesley: Reading, 1990.

[88] Adobe Systems Incorporated, *PostScript Language Tutorial and Cookbook*, Addison-Wesley: Reading, 1985. Traduit en français : *PostScript par l'exemple* InterÉditions, 1987.

[89] Adobe Systems Incorporated, *PostScript Language Program Design*, Addison-Wesley Publishing Company, Reading, MA (USA), 1988.

[90] Adobe Systems Inc., *The Type 1 Format Specification*, Addison Wesley, 1990.

[91] Jacques ANDRÉ & Justin BUR, « Métrique des fontes PostScript », *Cahiers GUTenberg*, 8, Mars 1991, 29–50. Une version anglaise se trouve dans [109, pages 64–77].

[92] Apple Computer, *The True Type Font Format Specification*, july 1990.

[93] Bruno BORGHI, « Les concepts graphiques de PostScript», *Cahiers GUTenberg* numéro 1, avril 1989, 45–51.

[94] Bernard-Paul EMINET, *Le livre de PostScript*, éditions PSI, 1987.

[95] Stephen F. ROTH (ed.), *Real World PostScript*, Addison-Wesley Publishing Company, Reading (USA), 1988.

A.5 Typograhie numérique: introductions, synthèses, actes de colloques

[96] Jacques ANDRÉ, « Font design: a bibliography » in [101, pages 122–125], 1987.

[97] Jacques ANDRÉ, *Introduction à la typographie numérique*, à paraître dans *Terminologie des Industries et Arts Graphiques*, Tradoc ed., Lausanne 1994, 45 pages.

[98] Jacques ANDRÉ, Jakob GONCZAROWSKI and Richard SOUTHALL, *Proceedings of the 3rd Conference Raster Imaging and Digital Typography*, à paraître comme numéro spécial de la revue *Epodd*, Wiley ed., vol.6(3), 1993.

[99] Jacques ANDRÉ et Roger HERSCH (eds.), *Raster imaging and digital typography*, Cambridge University Press, 1989.

[100] Jacques ANDRÉ and Roger D. HERSCH, « Teaching Digital Typography », *EPODD – Electronic Pubishing, Origination, Dissemination and Design*, vol. 5, n° 2, June 1992, 79–90. Paru auparavant en français: « Enseigner la typographie numérique », *Bigre*, n° 79, mars 1992 ; et *Publication Interne Irisa*, n° 636, 1992.

[101] Jacques ANDRÉ et Moncef MLOUKA, *Workshop on Font Design Systems – WFSD*, May 1987, Sophia, Inria publ. Une partie de ces actes a été reprise dans [99].

[102] Chuck BIGELOW et Donald DAY, « La typographie numérique », *Pour la Science*, octobre 1983.

[103] Alison BLACK, *Typefaces for desktop publishing – a user guide*, Architecture Design and Technology Press, London, 1990.

[104] DIDOT PROJECT, « A curriculum on digital typography », *Didot internal report*, juillet 1993. Partiellement paru dans *Didot Bulletin*, 3, juillet 1993, 6–8.

[105] P. COUEIGNOUX, *Generation of roman printed fonts*, Ph.D. Thesis, Massachusets Institute of Technology, Cambridge, Mass. 1975.

[106] Piero DE MACCHI, *L'avventura Didot: caratteri da stampa e nuove technologie*, Demacchi Progetti Grafici, Turin, 1994 (à paraître). (En anglais et en italien).

[107] Jim FLOWERS, « Digital Type Manufacture: An Interactive Approach », *Computer magazine*, may 1984.

[108] R.D. HERSCH, « Introduction to Font Rasterization », in [99, pages 1–13], 1989.

[109] Roger HERSCH (ed.), *Visual and Technical Aspects of Types*, Cambridge University Press, 1993. {Actes de l'« École Didot » tenue à Lausanne en septembre 1991.}

[110] Françoise HOLZ-BONNEAU, *Lettre, image, ordinateur*, Hermès/INA, Paris, 1987.

[111] Peter KAROW, *Digital Formats for Typefaces*, URW Verlag, Hamburg (RFA), 1987.

[112] Peter KAROW, *Fonttechnology*, Springer-Verlag, Heidelberg, 1993. Paru en allemand sous le titre *Schrifftechnologie* chez le même éditeur en 1992.

[113] Peter KAROW, *Typefaces Statistics*, URW Verlag, Hamburg, 1993.

[114] Robert A. MORRIS & Jacques ANDRÉ (eds.), *Raster Imaging and Digital Typography II*, The Cambridge Series on Electronic Publishing, Cambridge University Press, 1991.

[115] Richard RUBINSTEIN, *Digital Typography, An Introduction to Type and Composition for Computer System Design*, Addison-Wesley, Reading (USA), 1988.

[116] Lynn RUGGLE, *Letterform design Systems*, Standford University Technical Report STAN-CS-83-971, 1983.

[117] J. SEYBOLD et F. DRESSLER, *La micro-édition selon Seybold*, Dunod, Paris, 1987.

A.6 Typographie numérique : notes techniques, fontes

[118] D. ADAMS & J. ANDRÉ, « New trends in digital typography », in [99, pages 14-21].

[119] Jacques ANDRÉ, « *Courier* ou le passage de la machine à écrire aux imprimantes à laser », note interne Opéra, 1992. À paraître dans les *Cahiers GUTenberg*, 1994.

[120] Jacques ANDRÉ, « PostScript calligrammes », *PostScript Language Journal*, vol. 3 n° 2, 1990, 9-13.

[121] Jacques ANDRÉ, « Ligatures et informatique », *Cahiers GUTenberg*, n° 18, 1994, à paraître.

[122] Jacques ANDRÉ, « New tools, new fonts », *Tradición e innovación tipogràfica* (Sanchez ed.), Bilbao, octobre 1993.

[123] Mark ARGETSINGER, « Adobe Garamond: A Review », *Printing History* 26/27, 1991-1992, p. 69-98.

[124] Claude BÉTRISEY & Catherine ANDRÉ, « An enhanced PostScript Previewer for experimenting and teaching new approaches in digital typograpy », Proceedings of TEP'92, Lausanne, 1992, *Bigre* 79, 83-95.

[125] Claude BETRISEY, *Génération automatique de contraintes pour caractères typographiques à l'aide d'un modèle topologique*, Thèse, EPFL, Lausanne, 28 juin 1993.

[126] C. BIGELOW & K. HOLMES, « The Design of Lucida: an Integrated Family of Types for Electronic Literacy », in [82, pages 3-17] 1986.

[127] Charles BIGELOW and Kris HOLMES, « The design of a Unicode font », à paraître dans [98].

[128] M. BLOCH, *Génération de tâches bicolores. Applications aux caractères d'imprimerie - Problèmes de nature ordinale*, thèse, Saint-Étienne, 1981.

[129] P. COUEIGNOUX and R. GUEDJ, « Computer generation of color planar patterns on TV-like rasters », *Proceedings of the IEEE*, vol. 68, n° 7, juillet 1980.

[130] P. COUEIGNOUX, « Character Generation by Computer », *Computer Graphics and Image Processing*, vol. 16, 1981, 240-269.

[131] Stephen DEACH, « Outline Fonts Hints and Rasterization: A Technology Primer », *The Seybold Report on Desktop Publishing*, vol. 6, n° 7, March 9, 1992, 21-32.

[132] R. DEBRY, A.W. GRIFFEE & J.P. HOFMEISTER, « Multiple font technologies in a distributed environment », in [99], 223–231.

[133] J. GONCZAROWSKI, « Curve Techniques for Auto-Tracing », in [109]

[134] J. GONCZAROWSKI, « Industry Standard Outline Font Formats », in [109].

[135] Yannis HARALAMBOUS, « The Holly Quoran », à paraître, *Cahiers GUTenberg*, n° 19, 1994.

[136] Yannis HARALAMBOUS, « Parametrization of PostScript Fonts through METAFONT », à paraître dans [98], 1994.

[137] R.D. HERSCH & C. BETRISEY, « Model-Based Matching and Hinting of Fonts », *Proceedings SIGGRAPH'91, ACM Computer Graphics*, vol. 24, 1991

[138] R.D. HERSCH, J. BUR, C. BETRISEY, A. GÜRTLER, « Perceptually-Tuned Generation og Grayscale Fonts », soumis à publication, 1994.

[139] Bridget Lynn JOHNSON, *A Model for Automatic Optical Scaling of Type Designs for Conventional and Digital Technology*, Master of Science, School Printing in the College of Graphic Arts of the Rochester Institute of Technology, May 1987.

[140] Vania JOLOBOFF and Daniel DARDAILLER, « FEDOR: a Font EDitOR », in [101, pages 112–121], 1987.

[141] Peter KAROW, « *hz*-program, Micro-typography for advanced typesetting », URW, Hamburg, 1993. Traduction française à paraître, *Cahiers GUTenberg*, n° 18, 1994. Voir aussi Hermann ZAPF in [98].

[142] D.E. KNUTH, « The concept of a meta-font », *Visible language*, XVI,1, 1982, 3–27. Paru en français : « Le concept de Metafonte », *Communication et langage*, n° 55.

[143] D.E. KNUTH, *Computer Modern Typefaces*, Addison-Wesley, Reading, MA, 1986.

[144] Donald KNUTH, *The* METAFONT*Book*, AddisonWesley: Reading, 1984.

[145] Donal KNUTH, « A punk meta-font », *TUGboat*, vol. 9, no. 2, August 1988, 152–168

[146] Philippe LOUARN, « *Lucida*, une fonte complète pour LaTeX et son installation », *Cahiers GUTenberg*, 9, juillet 1991, 32–40.

[147] Gary MCGRAW, « Letter Spirit: Recognition and Creation of Letterforms Based on Fluid Concepts », *CRCC Technical Report* n° 61, Indiana University, June 1992.

[148] Pierre MACKAY, « Looking at the Pixels. Quality Control for 300 dpi Laser Printer Fonts, especially METAFONTs », in [114], 205–215. Traduit en français dans le *Cahier GUTenberg* n° 12, décembre 1991, 21–37.

[149] Hand Ed. MEIER, « On the design of Barbedor and Syndor », in [109], 148–164.

[150] Avi C. NAIMAN, *High-Quality Texts for Raster Displays*, PhD Dissertation, University of Toronto, janvier 1985. Aussi : « Technical Report » (CSRI-253) Computer Systems Research Institute at the University of Toronto.

[151] Avi NAIMAN, *The use of grayscale for improved character presentation*, PhD Thesis, Université de Toronto, Canada, 1991.

[152] M. NANARD, J. NANARD, M. GAMDARA & N. PORTE, « A declaratrive approach for font design by incremental learning », in [99], 1989, 71–82.

[153] R. SOUTHALL & J. ANDRÉ, « Experiments in teaching METAFONT », in *TEX for Scientific Documentation* (D. Lucarella ed.), Addison-Wesley, 1985, 141–153.

[154] Richard SOUTHALL « Designing a new typeface with METAFONT », *TEX for scientific documentation* (J. DÉSARMÉNIEN ed.), Springer-Verlag, Berlin, 1986, 161–179.

[155] Beat STAMM, « Object-orientation and extensibility in a font-scaler », à paraître dans [98].

A.7 Fontes dynamiques

[156] Adobe, *MultiMasters specifications*, 1992.

[157] Jacques ANDRÉ and Bruno BORGHI, « Dynamic fonts », in [99], 198–203.

[158] Jacques ANDRÉ and Bruno BORGHI, « Dynamic fonts », *PostScript Language Journal - International Edition*, vol.2, no.3, 1989, 6–8.

[159] Jacques ANDRÉ, « The Scrabble font », *The PostScript Journal*, vol. 3, num. 1, 1990, 53–55.

[160] Jacques ANDRÉ, « Nouveaux caractères pour une nouvelle typographie », *Texte et ordinateur - les mutations du lire-écrire* (J. Anis et J.L. Lebrave eds.), volume hors série de la revue *Lynx*, Centre de recherches linguistiques, Paris X Nanterre, 1990, 43–52.

[161] Jacques ANDRÉ and Corinna KINCHIN, « Adaptating contour character shape to point size », *PostScript Review*, May 1991, 31–36.

[162] Jacques ANDRÉ et Christian DELORME, « Le *Delorme*: un caractère modulaire et dépendant du contexte », *Communication et langage*, 86, 1990, 65-76.

[163] Jacques ANDRÉ et Victor OSTROMOUKHOV, « *Punk*: de METAFONT à PostScript », *Cahiers GUTenberg*, 4, 1989, 23-28.

[164] Jacques ANDRÉ, « *JAVAL* une fonte pour la recherche de palindromes typographiques », Note interne Opéra, Irisa, Rennes, avril 1993 ; à paraître dans *Communication et langages*, n° 99, 1994.

[165] Jacques ANDRÉ, « Random fonts and inkspreading simulation » , note interne, projet Opéra, Rennes, novembre 1992.

[166] Jacques ANDRÉ et Irène VATTON, « Contextual Typesetting of Mathematical Symbols Taking Care of Optical Scaling », *Rapport de recherche Inria*, n° 1972, juin 1993 ; à paraître dans *EP-ODD - Electronic Publication, Origination, Dissemination and Design*, numéro spécial « Computer processing of type » (R. Hersch ed.), 1994.

[167] Daniel BERRY and J. SROUJI, « Arabic Formatting with ditroff/ffortid », *EPODD - Electronic Pubishing, Origination, Dissemination and Design*, vol. 5, n° 4, december 1992, 163-208.

[168] Erik VAN BLOKLAND and Just VAN ROSSUM, « Different Approaches to Lively Outlines », *Raster Imaging and Digital Typogaphy II* (R. Morris and J. André eds.), Cambridge University Press, 1991, 28-33.

[169] Erik VAN BLOKLAND & Just VAN ROSSUM, « Random code - the Beowolf Random Font », *PostScipt Journal*, 3,1, 8-11.

[170] « Erik van Blokland & Just van Rossum », *Emigre*, n° 18, 1991, p. 23-25.

[171] « Pierre di Sciullo », *Emigre*, n° 18, 1991,6-22.

[172] Michael COHEN, « Blush and Zebrackets: Two Schemes for Typographical Representation of Nested Associativity », *Proceedings IEEE Workshop on Visual Languages*, October 1992, Seattle, Washington, 264-266.

[173] Michael COHEN, « Zebrackets: A Pseudo-dynamic Contextually Adaptive Font », *TugBoat*, 1993.

[174] R.K. JOSHI, « Computerised Latin and Hindi Scripts: Back to the hand », paper presented at the non-Roman type meeting, *Type 90*, Oxford, August 1990.

[175] Robin KINROSS, « The digital wave », *Eye*, Vol.2, n° 7, 1992, 27-39.

[176] R. Victor KLASSEN, « Variables width splines: a possible font representation? », à paraître dans [98], 1994.

[177] Roger LAUFER, « La calligraphie animée », in [42, pages 223-233], 1989.

[178] Roger LAUFER, « Calligraphie synthétique animée », *Culture technique*, n° 17, mars 1987, 273-275.

[179] Jonathan SEYBOLD, « Adobe's 'MultiMasters' Technology: Breakthrough in Type Aesthetics », *The Seybold Report on Desktop Publishing*, vol. 5, n° 7, march 4, 1991, 3-7.

[180] John SHERMAN, « Image Paragraphs », Notre Dame University, USA, 1993.

A.8 Courbes et mathématiques

[181] R.H. BARTELS, J.C. BEATTY & B.A. BARSKY, *An introduction to Splines for Use in Computer Graphics and Geometric Modeling*, Morgan Kaufmann, Los Alos, CA, 1987.

[182] Carl DE BOOR, *A Practical Guide to Splines*, Springer Verlag, New York, 1978.

[183] Gerald FARIN, *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, San Diego, CA, 1982.

[184] R. HERSCH, « Efficient rendering of outline characters », *Proceedings SID*, Las Vegas, may 1990, vol. XXI, 392-394.

[185] Michael KOKULA, « On the fly generation of ligatures in scripts fonts », paper submitted to *EPODD-Electronic Publishing*, special issue on fonts (R. Hersch ed.), à paraître, 1994.

[186] W.M.NEWMAN, *Principles of Interactive Graphics*, McGraw-Hill, 1979.

[187] M. PLASS, M. STONE, « Curve fitting with piecewise parametric cubics », *ACM Computer Graphics*, Vol 17, n° 3, 229-239, July 1983

[188] Thomas W. SEDERBERG and Eugene GREENWOOD, « A Physically Based Approach to 2-D Shape Blending », *Computer Graphics*, 26(2) (Siggraph'92), July 1992, 25-34.

A.9 Lisibilité

[189] R. DUAN and R. MORRIS, « The importance of phase in the spectra of digital type », *Electronic Publishing - EPODD*, vol. 2, n° 1, 47–59, 1989.

[190] Émile JAVAL, *Physiologie de la lecture et de l'écriture*, Paris, Félix Arcan, 1905. Republié par Retz, Paris, 1978.

[191] E. KOWLER (ed.), *Eye movements and their role in visual and cognitive processes*, Elsevier Science publ. BV, 1990.

[192] G. LEGGE *et alii*, « Psychophysics of reading », *Vision research*, vol. 25, 2, 239–252, 1985.

[193] R.A. MORRIS, « Image processing aspects of type », in [83, pages 140–155], 1988.

[194] R.A. MORRIS, K. BERRY, K.A. HARGREAVES, « Towards quantification of the effects of typographic variation on readability », *Proceedings of the SID conference*, Seattle, may 1993. Voir aussi *Technical Report* n° 6, UMB, Boston, 1993.

[195] J.K. O'REGAN, « Eye movements in reading » in *Eye movements and their role in visual and cognitive processes.*, Elsevier Science pub. 1990.

[196] *Recherches actuelles sur la lisibilité*, sous la direction de François RICHAUDEAU, Retz, Paris, 1984.

[197] Miles A. THINKER, *Bases for Effective Reading*, University of Minnesota Press, Minneapolis ; *Legibility of print*, Iowa States University Press, 1963.

A.10 Normes

[198] Jacques ANDRÉ, « Une casse de 38 000 signes », *Caractères*, n° 373, février 1994, 32–36.

[199] Daniel DARDAILLER, « Normes et fontes », *Cahiers GUTenberg*, numéro 4, janvier 1990, 2–8.

[200] M. GOOSSENS & E. VAN HERWIJNEN, « Introduction à SGML, DSSSL et SPDL », *Cahiers GUTenberg*, n° 12, déc. 1991, 57–70.

[201] ISO/IEC DIS 9541, *Information processing - font and characterinformation interchange*, ISO, 1991.

[202] B. MARTI et co-auteurs, *Télématique - techniques, normes, services*, Dunod, 1990.

[203] THE UNICODE CONSORTIUM, *The Unicode Standard, Worldwide Character Encoding*, 2 volumes, Addison-Wesley, Reading (USA), 1991 et 1993.

A.11 Divers

[204] P. BACCHUS, J. ANDRÉ & C. PAIR (eds.), *Manuel du langage algorithmique Algol 68*, Hermann ed., 1975.

[205] André BERTRAND, « La typographie et la loi », *Cahiers GUTenberg*, 1991, n° 8, 10–20.

[206] Ch. BIGELOW, « Notes on typeface protection », *Tugboat*, vol. 7(3), 1986, 146–151 ; aussi : « Du piratage des fontes », *TSI - Technique et Science Informatique*, vol.6, n° 3, 1987, 255–259.

[207] Douglas HOFSTADTER, *Ma Thémagie*, InterEditions, 1988.

[208] S. NORA & A. MINC, *Rapport au président de la République : l'informatisation de la société*, Documentation française, Paris, 1978.

Index alphabétique