RÉFLEXIONS TYPOGRAPHIQUES

D'UN GROUPE
D'UTILISATEURS FRANCOPHONES
sur
« Le logiciel
de mise en pages idéal »

typographie@irisa.fr

La liste de diffusion « Typographie » (*http://www.cru.fr/listes/typographie@irisa.fr/*), créée en février 1997 par des usagers francophones, s'est saisie au début de l'année 1998 d'une idée née d'une discussion informelle entre un de ses intervenants (Olivier Randier) et un *Type Director* d'Adobe (David Lemon) à propos de son projet K2 : intervenir *en amont* de la conception du logiciel, afin d'y faire prévaloir nos préoccupations, notre point de vue de typographes.

Comme il arrive souvent dans ce type de discussions, nous avons discuté quelques semaines, puis le débat s'est enlisé. En dépit de sa richesse, du nombre d'idées qui ont été émises, aucune synthèse n'a pu être dégagée en son temps. Et ce n'est que dix mois plus tard, au hasard de rencontres individuelles, que nous nous parvenons à remettre ce travail à ses destinataires « naturels » : Adobe, bien entendu, mais aussi les autres principaux éditeurs de logiciels de mise en pages (Quark, Calamus) et aux groupes TeX.

Le document qui suit est le reflet de nos préoccupations spécifiques. Nous n'avons pas voulu intervenir, ou très peu, sur des sujets certainement très importants, comme le fonctionnement général d'un logiciel « idéal » de PAO, la gestion des couleurs, l'intégration des images, les calques, les feuilles de style, ou telle ou telle autre fonction qui nous manquerait ou ne serait pas suffisamment confortable sur notre logiciel habituel. Bref, nous n'avons pas voulu créer de « cahier des charges » complet.

Typographes, frustrés par le manque de contrôle typographique (et plus spécifiquement micro-typographique) et la difficulté de mise en œuvre d'une mise en pages élégante avec les différents logiciels disponibles, nous nous sommes bornés au seul terrain, déjà vaste, du texte pur. C'est ainsi qu'ont été abordés des sujets aussi importants que la gestion des espaces, le gris typographique, les ligatures contextuelles, la maîtrise sur l'agencement d'un flot de texte, les césures, etc. À chaque fois, nous n'avons pas seulement dressé un catalogue des outils qui nous manquent dans notre travail quotidien, mais tenté de trouver une solution pratique au problème posé.

On pourra s'étonner de la forme que prend le document qui résulte de ces discussions. Il est vrai que c'est plus un *rapport* qu'une *synthèse* : nous n'avons pas voulu gommer l'aspect foisonnant, parfois contradictoire et toujours passionné, des différentes interventions. Les abonnés de la liste viennent de mondes professionnels divers, leurs pratiques de la mise en pages sont hétérogènes, les logiciels employés sont nombreux : résumer les différentes interventions, en faire un tout homogène, lisse, c'eût été stériliser bon nombre de points abordés et bon nombre d'opinions parfois divergentes. Certes, le document qui suit peut paraître touffu et difficile à suivre ; cela nous a paru préférable à la production de propositions fermées, closes, définitives.

De la même façon, nous nous sommes refusés à rééquilibrer la longueur des différents chapitres de notre document, ou à « creuser » plus encore certaines pistes qui ne sont qu'ébauchées : nos « Réflexions » sont le reflet exact des débats, parfois désordonnés, tels qu'ils ont eu lieu. Ainsi, la partie traitant du difficile problème de la typographie des mathématiques est-elle sans doute d'une longueur disproportionnée par rapport à son enjeu réel. Mais, tout simplement, ce sujet ardu a entraîné de longues et passionnées discussions, peut-être au détriment d'autres sujets non moins importants. Nous engageons cependant le lecteur à lire attentivement cette partie de notre texte : il verra que, par-delà les données techniques du traitement du texte scientifique, c'est toute la problématique de la gestion du texte courant et de la maîtrise de ce texte par le typographe qui y est (parfois souterrainement) abordée. Les questions, et bien souvent les solutions, qui y sont dégagées valent en réalité pour l'ensemble du projet.

Ainsi et à l'inverse, la partie « Traitement du texte » est certainement trop courte, et n'a pas fait l'objet de débats suffisamment approfondis. Mais ne peut-on déduire ce qui s'y serait trouvé du reste des contributions ?

On pourra trouver d'autres exemples… (Certains points n'ont même pas été examinés ! Ils sont cependant dans notre sommaire.) C'est que ces « Réflexions » ne sont finalement que des préliminaires, l'image à un instant donné d'un chantier toujours ouvert, de questionnements toujours renouvelés, d'un souci de perfection qui (par définition) ne saurait trouver de contours exactement définissables.

*
* *

Les auteurs (« l'auteur collectif ») souhaitent avoir apporté un point de vue nouveau à la production des logiciels de mise en pages. Ils espèrent avoir attiré l'attention sur des questions trop souvent négligées.

On ne peut que souhaiter que les éditeurs et les auteurs de logiciels se saisissent à leur tour de nos réflexions, de nos demandes et de nos propositions, et qu'ils nous offrent enfin des programmes aptes à traiter correctement ce qu'ils prétendent traiter : le texte, la page.

Pour notre part, nous restons ouverts et disponibles à tout débat, à toute expertise et à toute collaboration, au bénéfice de la typographie.

Alain Hurtig

DISTRIBUTION

JACQUES ANDRÉ : directeur de recherche – Irisa/Inria-Rennes, France (*http://www.irisa.fr/*), éditeur des conférences RIDT (Raster Imaging and Digital Typography), éditeur en chef des *Cahiers GUTenberg*, administrateur de la liste *Typographie*.

THIERRY BOUCHE : enseignant-chercheur en mathématiques pures, typographe amateur, logiciel utilisé : LaTeX. *http://technopole.le-village.com/Experluette/outil.html*

MICHEL BOVANI : professeur de mathématiques (lycée Descartes à Tours et I.U.F.M. de l'académie d'Orléans-Tours). Logiciel utilisé : TeX.

MICHEL BUJARDET : 47 ans, journaliste scientifique depuis 1978, ayant collaboré à un grand nombres de revues informatiques françaises. Créateur de polices de caractères, il anime sa propre fonderie aux États-Unis, nommée Match Software. Site français : *http://www.matchfonts.com/fr*

EMMANUEL CURIS : étudiant en chimie, amateur de typo. Venu à la mise en page sur Atari avec Timeworks Publisher. Utilisation principale : rapports de stage, mémoires, affiches pour des conférences… bref le classique des écrits scientifiques. Logiciel utilisé : Calamus SL version GEM/TOS.

ALAIN HURTIG : maquettiste-typographe professionnel depuis plus de dix ans. Logiciels couramment ou plus occasionnellement utilisés : XPress, Photoshop, FreeHand, Fontographer et FontStudio. Se passionne pour le « gris typographique », le travail de labeur, l'édition de textes scientifiques (sciences humaines et sciences « exactes »), la micro-typographie. A travaillé en typographie des mathématiques et des langues anciennes (non-latines). On peut voir certains de ses travaux personnels sur *l'Outil* : *http://technopole.le-village.com/Experluette/outil.html*

PHILIPPE JALLON : 35 ans, journaliste. Directeur de la publication et rédacteur en chef du mensuel *Médias interAfrique*. Ce mensuel ne paraît pas tous les mois ; en Afrique, on appelle ça un « irrégulomadaire ». A travaillé pour les journaux les plus nuls de toute la presse panafricaine. Est diplômé des plus mauvaises universités. S'intéresse à la typographie, seulement les jours d'insomnie (et la nuit, il dort aussi).

ALAIN LABONTÉ : informaticien de réputation internationale spécialisé dans le domaine du soutien des langues nationales dans les technologies de l'information depuis plus de 12 ans (ancien spécialiste des systèmes d'exploitation, plus de 27 ans d'expérience en informatique), rédacteur de normes canadiennes et internationales, impliqué dans le développement d'Unicode et de la norme ISO correspondante. Œuvre maîtresse : le projet de norme internationale de classement, sur le modèle d'une invention de mon cru (devenue norme nationale du Canada) *http://www.tresor.gouv.qc.ca/doc/classm.htm ; http://www.tresor.gouv.qc.ca/doc/techtri.htm*

JEAN-PIERRE LACROUX : éditeur, correcteur, orthotypographe. Bibliographie orthotypographique : *http://users.skynet.be/sky37816/Lx.html*

GILLES PÉRES : 25 ans, typographe (débutant : c'est ma cinquième année dans la profession) et « pseudo-informaticien » (je suis responsable de deux dizaines d'ordinateur là où je travaille en tant que technicien informatique). Je partage mon temps entre ces deux occupations. Logiciels utilisés : — XPress : livres de critiques, fascicules, publicités, affiches, catalogues, logos. — TeX et LaTeX : livres de critiques littéraires, fascicules concernant la typo, affiche.

PAUL PICHAUREAU : cet intervenant a disparu au moment de la finition de ce document. Était scientifique du contingent au moment des faits. Préparerait une thèse de physique à Strasbourg.

OLIVIER RANDIER : 34 ans, exécutant-maquettiste en publicité depuis dix ans par nécessité (XPress, Illustrator), et typographe par goût, passionné de dessin de lettre (FontStudio†) et de technologie typographique (particulièrement par le support multilingue *http://technopole.le-village.com/Experluette/ index.html*). Actuellement animateur de la liste *Typographie*.

HUGHES RICHARD : utilisateur principalement de Pagemaker depuis 1991 (conception, réalisation et flashage). Actuellement spécialiste assurance qualité chez OneVision (logiciels prépresses d'édition et de normalisation de fichiers PS/EPS/PDF *http://www.one-vision.com*). Directeur de la publication de *CV Magazine* en parallèle.

JEAN-DENIS RONDINET : correcteur, puis typographe et linotypiste de labeur pendant vingt ans ; professeur de micro-édition pendant dix ans ; maintenant correcteur au *Figaro* à Paris.

FRANÇOIS HENRI VILLEBROD : né à Rouen en 1947, autodidacte, apprend la typographie en Grèce à la fin des années 80. Créateur de polices de caractères, spécialisé dans l'alphabet grec contemporain. Se consacre au dévelopement de familles typographiques internationales. Logiciels utilisés : XPress, Illustrator, FontStudio, Fontographer, FontLab, VisualTrueType et programmes maison. *http://www.typiko.com*

SOMMAIRE

I – TYPOGRAPHIE

I.1 – Gestion typographique de base

a) Attributs de style

O. RANDIER — La logique des attributs de style, qui trouve ses origines dans la ToolBox d'Apple, il me semble, a atteint sa limite. S'agissant de traitement typographique pointu, il faut s'en affranchir pour progresser. Ses défauts principaux : son caractère binaire (oui/non) et sa limitation.

Graisse

L'attribut « Bold » n'est pas suffisant pour gérer des polices aux graisses multiples, comme beaucoup le sont aujourd'hui. Certes, il faut conserver des couples gras/non gras logiques, mais il serait souhaitable de pouvoir naviguer dans les graisses par incréments, via un raccourci clavier et un curseur sur une échelle graduée et de pouvoir modifier ponctuellement la graisse appelée par l'attribut « Bold ».

Exemple des attributs :

<gras>
<graisse = 4/5>
<gras, graisse = +3>

Voir aussi le paragraphe Gestion avancée des Multiple Masters de ce même chapitre.

Italique/oblique

L'attribut italique est source de nombreuses erreurs, car il permet souvent d'afficher l'italisation d'une police qui ne possède pas de version italique, ce qui est une entorse au Wysiwyg. Il serait souhaitable que ne soit possible que l'affichage de polices existantes (c'est valable aussi pour le gras). Il faudrait distinguer la fonction standard d'italisation de celle d'« obliquisation » qui permettrait de pencher une police ne comportant pas d'italique.

Ce qui donnerait deux attributs :

<italique>
<oblique, angle = x°>

Relief (Outline)

Bien que cet attribut soit fort peu typographique, il est devenu, hélas, incontournable. Puisqu'il serait difficile aujourd'hui d'en faire admettre la disparition, autant qu'il soit géré convenablement. Rappelons que le filetage doit être extérieur à la lettre, et non à cheval sur son contour. Il serait souhaitable que l'épaisseur de ce fileté soit paramétrable et que sa couleur puisse être définie de manière distincte du fond de la lettre, qui peut éventuellement être transparent.

<outline, épaisseur de filet = a, couleur de filet = b, couleur de fond = c>

À noter que la traduction française usuelle de ce terme (relief) est absurde.

Ombré (Shadow)

Attribut aussi douteux typographiquement que le précédent. Inutilisable en l'état actuel, car aucunement paramétrable. Les graphistes ont recours pour faire des ombrés complexes à des utilitaires de tierce partie (ex. : ShadowCaster), qui génèrent des images bitmap. Il serait

souhaitable, plutôt, de pouvoir paramétrer directement angle et distance de décalage de l'ombre, ainsi que sa couleur et sa valeur de flou éventuel. Le tout étant enregistrable comme paramètre typographique local (feuille de style) et générant simplement le code postscript adéquat à l'impression, sans oublier la transparence.

<ombré, angle = a°, distance = b, rayon de flou = c, couleur de la lettre = d, couleurs de l'ombre = e [t], f [t]>

Souligné/barré

L'attribut de soulignement fait intervenir une fonction postscript simpliste, intégré au format de fonte, qui consiste simplement en une épaisseur relative au corps et une hauteur relative à la ligne de base. Le résultat est généralement laid. Il serait préférable (certaines XTensions le font plus ou moins bien) de pouvoir spécifier soi-même épaisseur, distance de la ligne de base, et, éventuellement couleur. Dans ce dernier cas, il faut pouvoir préciser si le filet est devant ou derrière la lettre ! Une option importante serait de pouvoir interrompre le filet chaque fois que celui-ci rencontre une partie de lettre (descendante, en général), en spécifiant la distance par rapport à celle-ci.

<souligné, distance = a, épaisseur = b, couleur = c, plan = devant/derrière>

Barré et mot souligné ne sont qu'anecdotiques, et peuvent être considérés comme des variantes du soulignement. À noter que Souligné ne devrait pas être un attribut de la lettre, mais de la ligne, parce que, si on souligne un texte comportant des indices ou des exposants, on a des surprises désagréables…

Tout capitales

Rien de particulier à dire sur cet attribut, excepté qu'il doit impérativement respecter l'accentuation des bas-de-casse (c'est généralement le cas, heureusement), sauf si c'est spécifié par l'utilisateur. Dans le cadre du support multilingue, il doit gérer correctement les translittérations (par ex. : ß = SS, en allemand). Il faut aussi gérer de façon pointue la distinction entre l'attribut de style Tout caps et le changement de casse brut.

Petites caps

L'horreur absolue ! Cet attribut, qui ne provient pas de la ToolBox, est un bricolage immonde, qui procède par réduction (paramétrable) des capitales, d'où incohérence des graisses relatives. Deux solutions :

— pour les fontes possédant une police Expert ou SC, un lien comme celui qui existe entre romain et italique d'une même fonte permettrait de résoudre le problème (modification des spécifications de format du Type 1 ?). Resterait toutefois le problème de certains signes de ponctuations (apostrophe, point d'interrogation, d'exclamation, etc., qui ne sont pas nécessairement dans la casse approprié dans ce cas).

Une translittération par Unicode (voir chapitre I.3), du même type que celle qui intervient (en 8 bits) avec l'attribut Tout caps permettrait de résoudre le problème de façon élégante, en conservant aussi la distinction utile entre attribut ponctuel et changement de casse.

— pour les fontes Multiple Masters (voir la section consacrée à celles-ci), en l'absence de vraies petites caps, celles-ci pourraient (?) être générées par calcul d'une instance sur les axes corps, chasse et graisse. La hauteur des petites caps devrait alors être automatiquement calculée par rapport à la hauteur d'œil des bas-de-casse, sauf spécification contraire de l'utilisateur.

Exposant/indice/supérieur

Mêmes problèmes que ceux posés par les petites caps. Mêmes solutions*, donc. De plus, pour les deux premiers, il faut noter que leur paramétrage (réduction + décalage vertical) laisse la place à un arbitraire douteux. Pour l'exposant, XPress a introduit une amélioration avec l'attribut Supérieur, qui n'a qu'un paramètre de réduction de taille, le décalage vertical étant calculé de manière à aligner le supérieur sur la hauteur d'œil des capitales, ce qui est typographiquement correct. Ceci pourrait être étendu aux indices (alignement sur les descendantes). D'autre part, dans le cadre de la composition scientifique, il faudrait que ces attributs puissent être cumulatifs (exposants et indices à multiples niveaux), avec une limite de corps (absolue). De plus, il faudrait pouvoir spécifier les paramètres localement (feuilles de style) et non au niveau de tout le document (comme c'est le cas dans XPress). Bien entendu, ce paramétrage doit pouvoir se faire en relatif (%, par défaut), comme en absolu (corps).

* Toutefois, la solution Multiple Masters serait préférable, car elle éviterait de surcharger Unicode avec d'innombrables variations de casse.

Ceci clôt la liste des attributs courants de style de la ToolBox, mais pas celle des attributs qui seraient souhaitables pour gérer les problèmes typographiques plus complexes, notamment la composition scientifique. Nous n'allons pas énumérer ici tous les attributs envisageables (ce serait d'ailleurs sans doute impossible). Nous y reviendrons dans la section I.2.a.

b) Architecture d'attributs de style

Notons quand même que les attributs devraient reposer sur une architecture ouverte, avec une interface de « programmation », pour résoudre des problèmes spécifiques. Celle-ci doit permettre d'accéder à des paramètres plus variés, par exemple :

— extension d'un caractère sur plusieurs lignes (accolades, intégrales, etc.),
— Position relative à un caractère, un mot ou une expression sélectionnée (exposant et indice simultanés, vecteurs, position de la barre de fraction par rapport au signe égal, etc.),
— Décalage horizontal,
— Rotation,
— Symétrie,
— Inclinaison,
— …

Ceci implique de pouvoir s'affranchir ponctuellement de la logique linéaire de composition (alignement de *bounding boxes* les unes derrière les autres).

> P. PICHAUREAU — Quand j'en parle, j'ai l'habitude de distinguer attribut typographique et attribut graphique. La distinction est fondée sur la tradition.
>
> Les attributs typographiques sont les enrichissements typographiques les plus anciennement employés : gras, italique, couleur, capitales et petites capitales. Les autres sont plus là parce qu'on sait faire, plutôt que parce qu'on en a besoin : barré, souligné, relief, déformations, etc.
>
> Techniquement, les premiers nécessitent souvent une création spécifique de la part de l'auteur de la fonte, les seconds sont le plus souvent obtenus par un trai-

tement donné d'une fonte existante. Cette distinction permet de plus de différencier ce qui est du domaine typographique de ce qui est du domaine plutôt graphique (traitements utilisés non pas dans le texte, mais dans les titres, les affiches, les accroches, etc.).

O. RANDIER — C'est une bonne idée. Je pense qu'on pourrait, d'une part, les séparer, comme tu le suggères, dans le document et, d'autre part, proposer qu'une telle distinction soit apparente dans l'interface, dans un but éducatif.

J.-P. LACROUX — Je crois que cette répartition des « enrichissements » en deux catégories est trop « typo… graphique » et qu'elle masque l'essentiel : la langue écrite.

• Les capitales ne peuvent être considérées comme de simples attributs ou enrichissements typographiques pour la bonne raison que leur emploi le plus fréquent est d'ordre syntaxique et orthographique (majuscules). La preuve : cet attribut est en principe géré par les bons correcteurs orthographiques et grammaticaux.

• L'emploi de l'italique, des petites capitales, des lettres ou des chiffres supérieurs est en partie codifié et il ressortit à l'orthotypographie. On peut envisager que certaines occurrences soient prises en compte par les correcteurs.

• Le gras et la couleur sont purement typographiques (ce qui n'est pas le cas du « barré »), au même titre que le relief (!), l'ombré (!) ou les diverses déformations. Le gras, que bien souvent l'on considère comme un frère de l'italique, est à mon sens beaucoup plus proche du changement de corps, voire de police. La preuve : il y a plusieurs gras, plus ou moins gras, alors que l'italique n'est pas (encore…) plus ou moins italique, les capitales plus ou moins capitales.

Bref, un texte « ordinaire* » ne peut vivre sans capitales, sauf à être orthographiquement fautif. Il peut vivre sans italique et sans petites capitales, sans lettres ou chiffres supérieurs, mais il est possible qu'il soit alors orthotypographiquement défectueux. En revanche, il peut toujours vivre sans gras, sans souligné, sans relief, sans couleur, etc. : aucun de ces « manques » ne le rendra « fautif »…

* Cela ne concerne donc ni les faiseurs de vers ou de formules…

Je crois qu'il serait pédagogique que les logiciels cessent d'entretenir la confusion et qu'ils distinguent nettement (menus,

fenêtres de paramétrage, etc.) ces trois plans (orthographique, orthotypographique, typographique). Jusqu'alors, la confusion était comique. Avec les nouvelles techniques (polices MM), elle risque de devenir désastreuse. D'où l'intérêt de bien isoler ce qui n'est pas uniquement graphique…

O. RANDIER — Un point que j'ai oublié : il faut distinguer le comportement de l'italique ou du gras *d'emphase* de celui utilisé pour les variables, par exemple. Dans un cas, il s'inverse par rapport au style du texte courant (il devient romain si le texte est en italique), dans l'autre cas, il ne doit surtout pas bouger. Dans la façon de gérer ça, je verrai bien un sous-attribut « verrouillé » et un « inversible » applicable à n'importe quel attribut de style. En conservant la possibilité de l'« écraser » localement, en cas de besoin.

I - TYPOGRAPHIE

I.1 - Gestion typographique de base

d) Gestion du crénage

O. RANDIER — Actuellement la gestion du crénage fait appel à un système extrêmement fruste, les approches de paires. Celles-ci font partie du format de fonte, et doivent être programmées par le créateur de la fonte, travail délicat et fastidieux, de ce fait souvent réduit au strict nécessaire. Si les paires nécessaires sont absentes, l'opérateur peut les modifier, mais ce travail est rarement fait. L'inconvénient majeur de ce système est qu'il perd toute pertinence en présence de paires de caractères de fontes différentes, ou parfois simplement de corps différents (exposants, indices).

Calamus propose un système différent. Les fontes qu'il utilise présentent, en plus de la traditionnelle *bouding box*, une description sommaire de la forme de chaque signe, par juxtaposition de petits rectangles. Dès lors, c'est le programme qui se charge du crénage, en ajustant l'espace entre les signes de façon à imbriquer le mieux possible ces rectangles.

De même, le logiciel de création de fonte FontStudio, pour calculer les tables d'approches, offrait une fonction analogue, en proposant à l'utilisateur de créer, dans une sorte de couche alpha, une description vectorielle sommaire de la partie pertinente pour le crénage de la forme de la lettre. On pouvait ensuite, à partir d'un certain nombre de paramètres, lancer un calcul automatique des paires d'approche ou des talus gauches et droits.

Bien que le marché actuel des fontes ne propose pas ce genre de fonctionnalités, nous pensons qu'un tel système pourrait résoudre des problèmes qui vont devenir de plus en plus cruciaux avec l'accroissement de la composition multilingue. Il serait donc souhaitable qu'à côté de la méthode de crénage actuelle, on puisse avoir recours à un crénage calculé par le programme, soit par la méthode Calamus, soit par la méthode FontStudio (préférable, je pense), soit par d'autres méthodes (calcul de poids d'encre ?), peut-être susceptibles de ne pas nécessiter de modifications des formats de fontes. La puissance de calcul des machines permet d'envisager aujourd'hui ces solutions plus gourmandes.

> T. BOUCHE — Ce qui ressort de la discussion récente, c'est que le schéma hérité du plomb (approches moyennes + corrections éventuelles) est lui-même gênant dès qu'on sort du « mot moyen » (le genre de mots qu'écrivent les Français moyens ?). La question pourrait être : pourquoi ne pas ignorer les approches et placer intelligemment les caractères en fonction de leur place dans un mot, sur une ligne…
>
> > O. RANDIER — Dans le cadre d'un programme qui sait ce qu'il fait, oui. Mais il ne faut pas oublier que les fontes ne sont pas faites que pour être utilisées dans des logiciels de P.A.O. Les talus d'approche sont généralement réglés pour donner un résultat correct, même dans un logiciel qui ne tient pas compte des approches de paires.
> >
> > Dans mon idée, il y aurait trois niveaux de gestion des fontes :

— Chasses simplement juxtaposées, pour le petit logiciel à 10 F (calepin, SimpleText),
— Chasses + corrections d'approche pour le traitement de texte plus élaboré,
— Enveloppes + crénage logiciel pour le logiciel de mise en pages.

F.H. VILLEBROD — Juste remarque, et c'est correct pour les deux premiers niveaux, ce qui est d'ailleurs le propre de la situation actuelle. Mais pour ce qui est du crénage logiciel il faudra attendre une génération ultérieure d'intelligence artificielle, lorsque — peut-être malheureusement — on n'aura plus besoin de l'œil du typographe pour équilibrer les approches !

[…]

La gestion du crénage par enveloppe demeure toujours limitée dans ses possibilités. L'enveloppe doit de toute façon être créée par le dessinateur (avec FontStudio par ex.), ce qui représente pas mal de travail à tâtonner sans grande assurance de satisfaction dans tous les cas de combinaison de paires.

L'œil — mais surtout le cerveau qui travaille en arrière — opère une série de calculations complexes sans que le dessinateur lui-même ne s'en rende compte. La difficulté est facilement constatée lorsque, après obtention manuelle d'un équilibre plaisant des distances, aucune règle géométrique simple (de maintien d'une distance d'enveloppe par ex.) ne semble pouvoir s'appliquer pour expliquer le résultat.

Une manipulation comparative des valeurs de blanc et de noir a lieu à l'interface des deux caractères, mais elle ne tient pas compte d'une seule série de valeurs selon un axe particulier (le contour de l'enveloppe), mais plutôt d'une multitude hiérarchisée de séries de valeurs sur différents axes et à différents points du voisinage de contact des deux caractères. Des facteurs subjectifs et esthétiques entrent aussi en jeu. Essayez donc de normaliser cette procédure complexe et de la passer sur silicone !

Pour améliorer la situation du crénage dans le logiciel de mise en pages de haut de gamme, restons donc pragmatiques. Je pense qu'il faudrait plutôt demander une meilleure procédure de gestion des paires, y compris des paires avec espace (je me suis plaint récemment auprès de Quark à ce sujet), et même des triplets et quadruplets*, ainsi que des artifices de simplifications tels le regroupement des lettres aux contours latéraux identiques en classes de crénage*. (* particularités d'ailleurs prévues par le format de fontes OpenType).

O. RANDIER — Pour les classes de crénage, à noter que FontStudio peut s'en servir pour générer les paires d'approches depuis la dernière version, qui date de… 1990.

A. HURTIG — […] le crénage sur trois lettres, qui renvoie aux problèmes de crénage avec les espaces soulevé par T. Bouche (me semble-t-il). Dans une pub d'une très honorable société (en l'occurence mon FAI), je lis : « MAIS IL Y A DES LIMITES » (tout en cap, dans un espèce de caractère bâton du genre Frutiger, vous voyez ce que je veux dire — il est probable qu'avec un caractère à empattements, le problème serait moins visible).

L'espace entre « IL Y » et « Y A » est trop grand, optiquement il est presque le double de celui entre « MAIS IL » ou « A DES ». Il ne me semble pas certain qu'il faille créner automatiquement entre « {sp}Y », « Y{sp} », « {sp}A », etc. (l'approche « {sp}A » semble *presque* bonne) ; ça dépend d'ailleurs sans doute des polices. Ce n'est pas le doublet qui en cause, mais le triplet. On devrait pouvoir gérer automatiquement ça…

F.H. VILLEBROD — Un auxiliaire d'édition de paires doit pouvoir utiliser le texte de mise en pages lui-même (zoomable de préférence à 1 000 % plutôt qu'à 400 %) pour emmagasiner les corrections d'approche faites par l'utilisateur en plein texte, au gré de son travail. Les nouveaux tableaux de crénage obtenus doivent non seulement voyager avec le document, mais aussi être récupérables sur d'autres documents et pouvoir être exportés sous format AFM en vue de récupération sur un éditeur de fontes. Cela revient à gérer le crénage tout comme on gère les exceptions orthographiques ou de césure.

On devrait même pouvoir enregistrer des tableaux de crénages différents selon le corps utilisé, puisque le crénage du texte et celui des titrages sont différents.

Le fin du fin serait évidemment de pouvoir jouer sur les talus d'approche afin de modifier aussi le placement des glyphes dans leur *bounding box*. Faudra-t-il alors aussi un dictionnaire des talus d'approche associé à celui des crénages ? Ça se complique, mais pourquoi pas ?

I – TYPOGRAPHIE

I.1 – Gestion typographique de base

e) Gestion des C & J

A. HURTIG — Il ne saurait être question de traiter ici de l'entièreté du problème des césures, lequel fait l'objet de thèses savantes et de travaux érudits. Il ne saurait non plus être question de s'aventurer dans le domaine des césures *hors français* : pourtant, il y aurait sûrement à dire sur le césurage dans d'autres langues que la nôtre…

Il s'agit simplement de marquer une insatisfaction : le césurage n'est pas bon, et ceci dans aucun des logiciels présents sur le marché. Au point, que pour XPress, des plug-in spécifiques ont été développés pour césurer correctement dans différentes langues…

> Est-il admissible de devoir payer un surcoût pour obtenir un logiciel de composition qui césure correctement en français ?
>
> Est-il admissible de devoir payer encore plus pour pouvoir césurer dans différentes langues ?
>
> Est-il admissible de devoir passer des heures à corriger manuellement des césures incorrectes ?

Pourtant, il semble que les règles communes du césurage français soient connues des auteurs de logiciels (la plupart des interdits de césure semblent correctement gérés, par exemple). Alors, qu'est-ce qui se passe ?

En première instance, on ne peut que constater que la plupart des logiciels se limitent à trois paramétrages de césure :

1. Longueur minimum des mots à césurer, nombre minimum de lettres avant et après la césure (ceci influant sur le gris typographique).
2. Césure autorisée ou interdite pour les mots commençant par une capitale.
3. Nombre de césures consécutives.

En ce qui concerne les seules césures françaises, et compte non tenu de leur interaction avec les algorithmes de justification (qui feront l'objet d'une autre contribution), le paramétrage devrait porter sur les points suivants :

1. Longueur minimum du mot, nombre minimum de lettres avant et après.

1bis. nombre de césures consécutives autorisées.

2. Césure sur mots tout en capitales — incluant évidemment les capitales « en dur » comme les capitales comme attribut de style. (Oui-Non).
3. Césure sur mots *commençant* par une capitale (Oui-Non).

3bis. *Sauf* s'ils commencent une phrase (donc précédés par un point ou une fin de paragraphe) (Oui-Non).

4. Dans les mots composés (césure obligatoirement sur le trait d'union, ou obligatoirement hors du trait d'union) (Oui-Non).

5. Césure avant une syllabe finale muette (Oui-Non). [en ce qui me concerne, c'est non !]
6. Césure sur une avant-dernière ligne de paragraphe (Oui-Non).
6 bis et ter. Sur une dernière ligne de page, de page de gauche, de page de droite (Oui-Non).

A. LA BONTÉ — Ça fait des années que je me plains de cette erreur de conception des logiciels américains quand ils traitent le français. Il faut absolument recourir à un dictionnaire en anglais, alors qu'en français, le tout peut se traiter algorithmiquement. Tout petit écolier francophone sait où couper une syllabe… ce n'est pas le cas pour les anglophones, les règles sont beaucoup plus nombreuses, tellement qu'il est absolument nécessaire au commun des mortels d'avoir recours à un dictionnaire (tous les dictionnaires anglais qui se respectent indiquent où couper les syllabes dans chaque mot). C'est ce que les logiciels de conception saxophonique font… mais lorsque le mot n'est pas au dictionnaire, ils font des approximations, la plupart du temps malheureuses en français […].

Pour une fois que le français est plus simple à traiter, on n'a pas réduit la complexité des logiciels d'origine (mauvaise adaptation et mauvaise conception dans un but d'internationalisation).

Il vaudrait la peine de documenter les règles de manière exhaustive. Grevisse le fait mais il ne liste pas les exceptions (dont plusieurs me semblent d'ailleurs optionnelles si l'on interprète ouvertement ce que *Le bon usage* en dit), notamment le principal, obligatoire, soit la liste des digrammes insécables.

G. PERES — Dans le *Webster*, dictionnaire d'anglais, on utilise deux signes pour signaler la division en fin de ligne : le trait d'union pour les coupures de mots qui ne contiennent pas de trait d'union à l'endroit coupé, et une sorte de « = montant » pour indiquer que le mot porte un trait d'union et que la césure est à l'endroit du trait d'union .

J.-P. LACROUX — Oui, mais je ferai au *Webster* le même reproche qu'au *Lexique du français pratique*… Si le fait d'employer deux signes est une excellente idée, il est dommage de ne pas laisser le « vrai » trait d'union aux mots qui en possèdent un ! Je préfère la méthode inverse (celle de Girault-Duvivier) :

sous-
marin
anti= (ou ¬, ou, mieux, car moins lourd, trait d'union incliné)
brouillard

P. JALLON — D'aucuns ont suggéré la possibilité d'utiliser le signe égale (=) ou un trait d'union incliné. Dans le premier cas, le résultat ne caresse pas l'œil dans le sens du sourcil. Dans le second cas, il s'avère que certaines polices utilisent déjà un trait d'union incliné qui, par conséquent, ne devient pas discriminant.

Je pense toutefois qu'il y aurait peut-être une solution ; reste à savoir si elle peut techniquement être mise en œuvre. Quand on parle de traits d'union inclinés, il s'agit *toujours* d'une inclinaison du bas vers le haut (en partant du principe que les langues romanes se lisent de gauche à droite). Eh bien, pourquoi ne pas générer un trait d'union spécial, incliné… du haut vers le bas ?

Bien sûr, on rétorquera qu'un tel signe n'existe pas et que même les fondeurs les plus tordus n'ont pas songé à l'inclure dans leur production. Il suffit que le logiciel réussisse à produire des césures (quand elles doivent se substituer à des traits d'union) inclinées du haut vers le bas, et le tour est joué. […]

> O. RANDIER — C'est marrant, j'ai pensé exactement la même chose que toi. Un trait d'union incliné vers le haut et un tiret de césure incliné vers le bas serait peut-être une bonne solution. Sur le plan mise en œuvre, ça ne devrait poser aucune (?) difficulté à un logiciel capable de faire un *vertical flip* d'un signe. Par contre, je me demande quand même si ce serait suffisamment discriminant.
>
> > T. BOUCHE — C'est marrant, elle me plait assez peu. Le trait d'union est incliné dans certaines polices par cohérence avec son axe (imaginer une écriture à la plume par un droitier : un trait conçu comme horizontal sera souvent légèrement incliné vers le haut ; c'est ce modèle qu'ont suivi les premiers humanistes). Rien n'interdit par suite
> >
> > — d'imaginer des caractères avec un axe « contre-oblique » donc tiret incliné vers le bas ;
> >
> > — de penser que l'introduction d'un signe déterminé *a priori*, en contradiction avec la cohérence de la fonte, n'est pas très typo (cf. discussions sur le symbole de l'euro…).
>
> O. RANDIER — Une autre solution pourrait nous être suggérée par les manuscripts médiévaux, où l'on remarque que la césure, notamment en gothique, est souvent exprimée par un double trait oblique. On revient à l'idée du signe =, mais ce ne serait alors pas un signe égal, effectivement gênant, mais un double trait d'union. Là encore, ça ne devrait pas poser de problème à un logiciel capable de superposer deux signes et de jouer sur leur position par rapport à la ligne de base. L'avantage de cette dernière solution, c'est qu'elle est compatible avec les deux types de traits d'union. Un trait d'union droit redoublé donnera un double tiret très court (distinct du signe égal), un trait d'union incliné donnera un double trait incliné (évidemment beaucoup plus élégant). Si en plus ça peut être justifié historiquement…
>
> > T. BOUCHE — Je crois que là encore, la bonne idée à retenir est qu'il est parfois avisé d'utiliser comme « div » un signe différent du trait d'union. C'est possible en TeX… J'ai déjà utilisé cette fonctionnalité pour :
> >
> > — supprimer purement et simplement la division dans telle compo « en colonne » ;

— utiliser une div à chasse déclarée nulle, donc pendante (on pourrait aussi imaginer des astuces de crénage spécifique avec ce caractère, par exemple crénage <fin de ligne>, même dans le cas où le glyphe imprimé est le trait d'union).

De là à admettre qu'un programme soit capable de créer tout seul un double trait de césure à la façon des moines copistes, typographiquement correct (épaisseur du trait et du blanc circonscrit), ce serait penser qu'il peut aussi bien déduire la place idoine des accents, ou composer nos guillemets à partir des *guilsinglleft*…

O. RANDIER — Pourquoi pas ? ;) Non, ce que je voyais plutôt, c'est un petit éditeur de « metaglyphe », qui permet de composer son signe à l'écran en en mixant plusieurs, lequel est ensuite enregistré comme une sorte de macro et peut être rappelé d'un bloc par l'utilisateur (ou par le programme dans le cas qui nous préoccuppe). Une sorte de version *lite* de l'éditeur de maths, en quelque sorte (ou une extension de l'usage d'icelui). Il me semble qu'il y a (avait ?) une Xtension qui utilisait ce genre de système pour pallier l'absence des diacrités « estrangers » dans le codage courant.

Et ça devient nettement plus concevable dans le cadre d'un logiciel qui crène lui-même ou qui sait gérer les triplets ou quadruplets.

E. CURIS — Je travaillais comme ça avec Calamus et les fontes sans majuscule accentuée, avant de recevoir un éditeur de fonte pour Calamus. Ça dépanne une fois de temps en temps, mais je trouve que ça n'est pas franchement pratique dès que le caractère trafiqué revient souvent. Même en associant, par exemple, control-M/a au bloc « A<correction d'approche>accent seul<correction d'approche> », on en a vite marre. À mon avis, un chercher/remplacer est plus pratique d'emploi.

Il est vrai aussi qu'il n'y avait pas moyen de l'inclure de façon plus automatique dans le programme que de l'associer à une combinaison de touches particulière, donc... Et qu'il fallait une combinaison par caractère trafiqué *et* style de ce caractère, ce qui peut vite devenir lourd.

A. HURTIG — C'est de plus très dangereux… pour la typographie ! Il y a un petit plug-in freeware pour XPress qui fait la même chose (c'est à ça qu'Olivier faisait allusion, je suppose), et de quoi s'aperçoit-on ?

D'abord que le réglage de la position des accents est presque impossible à l'écran. Ça n'a rien de difficile à faire, mais c'est minutieux et 72 dpi sont insuffisants : il faut imprimer la lettre, dans différents corps, sur papier… Et puis ne pas oublier de générer la correction d'approche dans toutes les déclinaisons de la police (ital., gras, etc.) Et tout ça pour voir ses

efforts ruinés dès que le logiciel écarte un peu les lettres, ou au contraire les rapproche, pour justifier une ligne !

Cela dit, dans la perspective d'un système qui utiliserait des *metafontes* (j'espère que le document est suffisement explicite sur ce point : quelle que soit la méthode employée, il *faut* avoir quelque chose dans ce genre), je me demande comment on peut se passer d'un éditeur de « glyphes » (je continue à ne pas aimer ce mot !) pour :

— récupérer des caractères autrement inaccessibles (les caractères islandais, le signe multiplié, etc., qui restent cachés dans le codage Mac des polices Adobe…) ;

— Construire des lettres diacritées « en dur virtuel » (sans correction d'approche, mais en simulant l'assemblage de plusieurs signes codés comme s'il s'agissait d'un seul caractère) lorsque celles-ci ne sont pas disponibles en standard ;

— Générer des fontes virtuelles hétérogènes, rassemblant des lettres venues de différentes polices. Par exemple, pour ne pas changer de police quand on fait des maths, ou pour faciliter la gestion des ligatures ornementales.

On n'a pas tout ça, alors qu'en aval, PostScript sait très bien le gérer, et qu'en amont, TeX s'en occupe aussi, preuve que c'est possible à faire, mais sans éditeur digne de ce nom, si j'ai bien compris.

Actuellement, la seule solution est de *fabriquer* une nouvelle police, en lisant les polices existantes à l'aide d'un éditeur de fontes, puis de générer le code d'une « nouvelle » police. C'est quand même un peu lourd…

e) Gestion des C & J — espaces spéciales

A. HURTIG — […] Il existe des caractères spéciaux dont les typos ont besoin.

Dans l'immédiat, je pense aux espaces non justifiantes, généralement absentes des logiciels (XPress en connaît une, de la valeur du demi-cadratin je crois).

Réclamons donc trois espaces non-justifiantes, une de la valeur de l'espace mot, une autre d'un demi-cadratin, une autre de la valeur d'une fine.

> T. BOUCHE — Tu les vois insécables ou pas ? (ou est-ce un attribut supplémentaire ?)

A. HURTIG — Y a-t-il d'autres caractères microtypographiques dont nous avons besoin ?

> T. BOUCHE — début et fin de mot ou ligne ? En TeX il y a un *Compound word mark*, qu'on insère entre deux mots composés en allemand, ce qui permet de les couper proprement (sans tiret de césure ?)
>
> il y a aussi les histoires de ligatures plus ou moins intelligentes, ce qui dépend beaucoup de problèmes de codage ou de la capacité du programme à discuter avec ses fontes (capacité à créer à la volée des ligatures entre tel et tel caractère substitué par tel autre, si l'AFM contient une instruction L par exemple), donc de s'adapter tout seul à des fontes spéciales qui ont plein de ligatures par exemple.
>
> > J.-D. RONDINET — Avec :
> > — une fine quart de cadratin insécable,
> > — une espace justifiante insécable,
> > — une espace justifiante sécable,
> > je pourrai produire du texte dans tous les domaines (généralistes, presse, labeur, édition) que je connais.
> >
> > La « fine quart de point » (ou une quelconque « micro-fine ») serait un plus, car elle permettrait de régler systématiquement par cherche-remplace certains problèmes que nous résolvons actuellement manuellement par une modification d'approche.

T. BOUCHE — si c'est un vrai problème (du genre crénage <f><è> absent ou <T><ÿ> erroné), je pense qu'il vaut mieux pouvoir faire la modif' depuis le programme, et sauver directement un nouvel AFM ? Sinon c'est vrai que j'ai eu occasionnellement recours à ce genre de chose.

> J.-D. RONDINET — Non, je pensais à des problèmes comme :
>
> … comme on l'a vu dans mon rapport[1].
>
> La fine 1/4 cad. est un peu trop large pour séparer le 1 supérieur. Idem si une supérieure ou une parenthèse fermante en romain suit un mot en ital', il y a quelquefois « choc »… Des problèmes non systéma-

tiques et dus à une fonte, donc, mais d'« œil » et au cas par cas, qu'on peut résoudre par « cherche-remplace ».

T. BOUCHE — Dans le système idéal dont nous parlons, la connaissance de l'enveloppe des caractères *devrait* permettre de placer correctement les exposants et de gérer comme il faut la correction italique… Mais ne tombons pas dans le travers qui consiste à renvoyer la balle au génie informatique ou logiciel (tout virtuel…) en nous en lavant les mains… Oui je crois qu'une ultra-fine comme ça (disons 1/2-point ou 1/20 cadratin devrait faire l'affaire) a son utilité, mais alors j'en réclame *aussi* une négative !

E. CURIS — Quelle différence avec l'insertion d'un code d'approche manuelle ?

J.-D. RONDINET — Le « cherche-remplace » ! Très important !

E. CURIS — Argument rejeté : on peut faire du « chercher/remplacer » sur et avec des codes d'approche manuelle, si le programme est bien conçu. Pas encore trop subtil (c'est-à-dire, pas de « remplacer toutes approches inférieures à 1 p par des approches de 1 p » — mais c'est pareil avec les espaces, alors), mais c'est déjà un bon début : rien n'empêche de faire « chercher « fè » et remplacer par « f[approche corrigée]è », tout autant que « chercher fè et remplacer par f[espace de x % cadratins]è ».

Donc ma question demeure, si ce n'est qu'avec une approche, on peut aussi modifier la position verticale et pas avec une espace.

J.-D. RONDINET — Si cette commande est dispo' dans K2, j'ai rien à dire, O.K…

Mais qui dit qu'elle y sera, ou qu'on le demandera ? Cette notion n'existe dans aucun TDT ni PAO… sauf le tien. Ce ne sera peut-être pas un des chevaux de bataille d'Adobe !

Alors… j'aime mieux tenir que courir !

T. BOUCHE — la différence, c'est que ça n'est pas parce qu'un chiffre supérieur est mal placé que ses approches sont incorrectes, c'est plutôt que la notion d'approches est inadaptée à cette situation (refrain habituel…). Ma philosophie dans toute cette discussion est de dire qu'il faut demander le maximum d'automatismes, car c'est plus cool pour nous ;-) mais aussi de *toujours* pouvoir corriger à la main les erreurs commises par le programme.

J. ANDRÉ — C'est aussi la mienne et je pense que là se trouve la grande différence entre les tenants du tout wysiwyg et ceux plus habitués à des produits qu'on disait « batch » autrefois.

A. HURTIG — Je ne vois pas où est le problème. On peut parfaitement avoir un programme tout Wysiwyg *et* pouvoir accéder aux entrailles du document, sous forme de métalangage par exemple.

XPress le fait déjà un peu, avec une exportation de fichier qui tague toutes indications typo du flot de texte, et qu'on peut aller corriger à la main (ça ne concerne que le texte et éventuellement les images ancrées, et plein de choses ne sont pas prises en compte, notamment la position des pavés de texte dans la page, ni leur taille, ce genre de trucs — cela dit, ça dépanne!).

> J. ANDRÉ — C'est le contraire que je veux dire : si on a un outil automatique (qui ne peut pas toujours être parfait), il faut aussi lui ajouter des possibilités Wysiwyg pour faire des mises au point locales.

A. HURTIG — Comprend pas… « Mise au point locales »? Mais je veux visualiser ma page, moi, toute entière, pas localement. Et je veux même visualiser toutes mes pages, et les modifier à l'écran en temps réel.

> J. ANDRÉ — Mais je pense quand même qu'il vaut mieux avoir d'abord un outil automatique qui fait le principal du travail, sinon Ducros il se décarcasse pour rien!

A. HURTIG — Comprend encore moins… Il y a *forcément* un « outil automatique qui fait le travail ».

Le Wysiwyg est une surcouche, à la fois de visualisation (percevoir ce que l'on fait, sans passer par un long travail d'abstraction mentale) et de production (déplacer un bloc à la souris, modifier un vecteur, dessiner, enfin tout ça…)

> T. BOUCHE — Oui, enfin le Wysiwyg dont on parle là en ce moment, parce que bien souvent, on ne peut faire *que* ce que l'on voit, avec du Wysiwyg pas évolué. C'est pour ça qu'on parle d'automatismes débrayables, et de contrôles plus fins que ce qui est possible à 72 dpi avec une souris.
>
> L'idéal pour moi, ce serait de laisser la micro-typo à un programme comme TeX ou hz ou nts ou… et l'interface Wysiwyg pour l'aspect table de montage virtuelle. Par exemple, j'essaierai de répondre sur les maths plus tard, mais en maths l'espacement de la lettre *f* n'est pas le même si cette lettre est une fonction, une relation ; une virgule peut représenter une ponctuation (de nature textuelle) une ponctuation mathématique (intervalle [a, b]) ou le séparateur décimal (1,2). Comment faire, en Wysiwyg, pour utiliser le même glyphe espacé différemment en fonction de sa nature ?
>
> Donc, comme déjà dit : interface Wysiwyg, avec langage à balises sous-jacent, qu'il est possible d'éditer directement, ou simplement accès par (petite) boite de dialogue pour modifier tel paramètre à tel endroit. Une fois qu'on a cette architecture, je pense effectivement que le développement de plug-in doit être facilité, donc qu'il n'est pas nécessaire de tout avoir tout de suite.

E. CURIS — Et puis, si on va dans ce sens-là, une seule espace dont on peut demander la largeur, c'est plus puissant et aussi pratique, non ? Et pour les programmeurs, ça doit être plus simple (pensons à eux : -). Sous réserve qu'il existe des raccourcis pour avoir cette espace avec déjà telle ou telle valeur, pour que l'utilisateur soit content aussi.

T. BOUCHE — Oui, ça, c'est un détail d'implémentation. Chez moi, rajouter ce genre d'espace, ça veut dire taper \kern-.1ex, alors un peu de convivialité dans tout ça…

J.-D. RONDINET — Tant que j'y pense : attention que les codes d'appel de ces espaces puissent être insérés au clavier dans un dialogue de « cherche-remplace » (pas de pomme-espace ou de Ctrl-quelque chose !), au pire sous la forme d'une balise absconse (\f) ou d'un code ($111)...

P.-S. : — Quelle est la valeur (relative à quoi ?) de l'espace-mot dans XPress et autres ? et celle de l'espace ponctuation ? Les manuels ne sont pas diserts sur le sujet…

T. BOUCHE — elle dépend de la police. Dans les corps usuels, elle est plus ou moins identique à la chasse d'un « i ». Elle devrait diminuer dans les grands corps mais augmenter dans les petits. En général, son élasticité est gérée par le programme, du genre valeur donnée dans l'AFM ± x%. C'est souvent de l'ordre du quart de cadratin.

J.-D. RONDINET — Question annexe, mais concernant aussi les espaces : dans toutes les fontes de labeur actuelles, les chiffres ont-ils toujours la chasse d'un 1/2 cadratin ? C'est important si on exige de K2 un tableautage digne de ce nom […].

T. BOUCHE — On l'a déjà dit, je suppose que ça atterrira un jour dans la FAQ, il y a plusieurs sortes de chiffres en circulation de nos jours : capitales — habituellement à chasse fixe, mais pas toujours, Adobe fournit parfois un *onefitted* dont la chasse réduite évite le massacre du gris dans un titre — qui vivent plus ou moins dans la *bounding box* d'un N ; bas de casses ou elzéviriens — habituellement à chasse fixe *aussi*, contrairement aux idées reçues à ce sujet — qui sont souvent fournis avec des crénages permettant de les combiner harmonieusement dans du texte (mais là c'est un exemple archétypal du problème des approches en bout de « mot ») ; on trouve encore, mais plus rarement (par exemple en Bell) des chiffres « petites capitales » (i.e. alignés sur ces dernières) qui correspondent bien à cette (fausse bonne ?) idée anglo-saxonne que les petites caps sont un moyen de céder à l'amour immodéré de la capitalisation sans que ça ne se voie trop…

O. RANDIER — Les chiffres « petites capitales » sont pourtant bien utiles pour créer des fractions : si tu as une fonte com-

portant des chiffres exposants et des chiffres « petites caps », il ne te manque que la barre de fraction (et non le slash, comme je te le faisais remarquer en un autre lieu) pour composer toutes les fractions possibles. Avec une bonne gestion du crénage, ça marche tout seul et ça évite le recours à des non-sens style fontes de fractions.

T. BOUCHE — je ne suis pas certain que nous parlions de la même chose. Il y a dans le complément expert des chiffres que j'ai oublié dans ma liste parce qu'ils ne relevaient pas vraiment du problème posé : les inférieurs et les supérieurs, les premiers moins hauts que l'œil, les seconds plus ou moins alignés sur les caps. Ils sont effectivement idoines pour s'assembler en fractions ou pour être utilisés en exposants/indices, mais un peu petits pour servir dans du texte ou des tables. Ce dont je parlais, sous le nom de chiffres petites caps, c'est ceux qu'on trouve dans MT Bell Expert (à la place des elzéviriens) qui sont vraiment aux chiffres capitales ce que les petites caps sont aux caps (géo)métriquement. Les Américains (enfin certains, dont Hudson) appellent ça *short digits*.

T. BOUCHE — Le système à chasse fixe + crénage est le plus simple pour avoir quelque chose de cohérent dans le texte, et des alignements corrects dans les tables : il suffit d'ignorer le crénage dans les tables.

O. RANDIER — Sauf que dans le système que nous envisageons, ce serait plutôt le contraire : on crène en ignorant les chasses partout, sauf dans les tableaux. Ce qui rend le *onefitted* inutile.

T. BOUCHE — Pour le texte, l'idéal est évidemment les chiffres bas de casse à chasses proportionnelles (+ crénage dans certains cas comme <7><4>, et crénage avec l'espace, le point <7><.>, …).

Pour les titres en capitales, l'idéal est des chiffres capitales à chasses proportionnelles.

Pour les tableaux, en général des chiffres elzéviriens à chasse fixe sans crénage.

Dans d'autres situations comme les appels de notes, les exposants mathématiques, les chiffres bas de casse sont souvent bizarres, mais les autres mal foutus et pas très lisibles et pour tout dire incongrus.

J'en profite pour revenir sur un de mes dadas : Unicode prévoit une distinction entre capitales et bas de casse (enfin, entre capitales et petites… lettres), des tas

de formes d'astérisques, mais pas des chiffres capitales et bas de casse ! C'est très stupide, car ça rend très difficile la cohabitation des deux dans un même document, ce qui est pourtant une nécessité si on a un titre en capitales, ou si le programme de mise en page reprend des bouts de texte pour les réemployer dans d'autres contextes (qui peuvent être tous caps, on peut penser à des titres de section, aux fragments en petites capitales après une lettrine, …)

O. RANDIER — En fait, c'est beaucoup plus tordu que ça. Dans Unicode, il y a des chiffres capitales, des chiffres exposants, des chiffres « petites caps », bref, tout, sauf des chiffres bas-de-casse (à moins que, dans leur esprit, les chiffres « petites caps » soient des bas-de-casse…) ! Ce qui nous ramène au débat à la c… sur la distinction caractère/glyphe dans Unicode. Parce que, s'il y a des chiffres petites caps, on ne voit pas trop pourquoi il n'y aurait pas aussi des lettres. Et, après tout, tous ces chiffres ne sont que des glyphes différents d'un même caractère.

T. BOUCHE — À moins que la solution souhaitable ne repose sur une profusion de fontes utilisées en fonction du contexte ?

T. BOUCHE — Je signale aux intervenants sur ce sujet la publication du standard MathML (voir *http://www.w3.org/TR/PR-math/toc.html*). […]

Sur les espaces, voici ce qui est prévu (on voit qu'Unicode en a prévu quelques-unes, avec des noms très glyphiques, à mon goût, les « entités MathML » ayant des noms beaucoup plus raisonnables) :

Entity name	Unicode	Description
	X09	tabulator stop
X10	force a line break
&IndentingNewLine;	—	force a line break and indent appropriately on next line
	—	never break line here
&GoodBreak;	—	if a linebreak is needed, here is a good spot
&BadBreak;	—	if a linebreak is needed, try to avoid breaking here
&Space;	0020	one em of space in the current font
	00A0	space that is not a legal breakpoint
	200B	space of no width at all
	2009	space of width 1/18 em
	2006	space of width 3/18 em
	2005	space of width 4/18 em
	2004	space of width 5/18 em

Entity name	Unicode	Description
	—	space of width –1/18 em
	—	space of width –3/18 em
	—	space of width –4/18 em
	—	space of width –5/18 em
⁣	—	used as a separator, e.g., in indices (Section 3.2.4)
⁢	—	marks multiplication when it is understood without a mark (Section 3.2.4)
⁡	—	character showing function application in presentation tagging (Section 3.2.4)

(— = pas dans Unicode)

e) Gestion des C & J — route naturelle du texte

A. HURTIG — Un effet de mise en pages qui était courant dans la presse (et parfois dans le livre) consiste à mettre un texte sur deux colonnes et les intertitres sur une seule, cette colonne étant de la largeur des deux colonnes de texte.

C'est pas clair ? Alors voilà :

TITRE-TITRE
XIXIXI XIXIXI
XIXIXI XIXIXI
XIXIXI XIXIXI

Ça existe toujours, me direz-vous. Non, pas tellement. On trouve toujours des inters, ou des citations en gros caractères placées en fausse justification et enjambant plusieurs colonnes, mais le flot de texte ne suit presque plus jamais sa route naturelle, qui serait de rester bien gentiment bloqué sous son intertitre :

TITRE-TITRE
XIXIXI XIXIXI
XIXIXI XIXIXI
XIXIXI XIXIXI

TITRE-TITRE
IOIOII IOIOII
IOIOII IOIOII
IOIOII IOIOII
IOIOII IOIOII

On voit plutôt des horreurs du genre :

TITRE-TITRE
XIXIXI XIXIXI
XIXIXI IOIOII
XIXIXI IOIOII

TITRE-TITRE
XIXIXI IOIOII
XIXIXI IOIOII
IOIOII IOIOII
IOIOII IOIOII

Et ceci pour une raison simple : la « route naturelle de texte » est horriblement difficile à mettre en place.

> En tout cas dans XPress, où le nombre de colonnes est un attribut du bloc qui *contient* le texte, et non pas un attribut du texte lui-même, c'est une vraie galère de faire ça. Évidemment : la moindre correction, raccourcissant ou allongeant la partie « XIXIXI », fait chasser la parte « IOIOII » et tout est à recomposer !
>
> La solution serait alors que le colonnage et la justification soit un attribut du texte (dans la feuille de style par exemple) et non pas du « cadre » virtuel qui le

contient. Le cadre détermine la justification (horizontale et verticale), mais c'est le style du texte qui dit au logiciel comment il se dispose…

L'intérêt de cette méthode serait qu'elle permettrait de mieux gérer tout autre élément extérieur au flot de texte principal, comme des imports d'images, des tableaux, et j'en passe.

D'ailleurs, ça existe peut-être déjà (j'avais vu un vieux logiciel de compo qui marchait comme ça, mais je ne me souviens plus de son nom), sur Calamus par exemple, va-t-en savoir :-)))

e) Gestion des C & J — débordement de texte

J.-D. RONDINET — […] l'extermination de la notion même de « petit carré avec une croix » pour les textes trop longs, au profit d'une indication (beaucoup) moins discrète.

f) Habillage

J.-D. RONDINET — Il faudrait pouvoir appliquer une C&J à une zone de texte, indépendamment du format de paragraphe, pour gérer finement les césures lors d'un habillage.

I – TYPOGRAPHIE

I.1 – Gestion typographique de base

g) Gestion avancée des Multiple Masters

o. randier — La plupart des typographes sont d'accord pour estimer que Multiple Masters a été une évolution majeure des formats de fontes, notamment parce qu'elles permettent de retrouver les subtilités des corrections optiques en fonction du corps, impossibles jusqu'alors en P.A.O. Malheureusement, ces fonctionnalités ne sont pas exploitées pour le moment, et la mise en œuvre des Multiple Masters est telle qu'elle dissuade généralement les utilisateurs de s'en servir. Un logiciel moderne de mise en pages ne peut plus les ignorer et doit leur assurer un support total, qui implique une gestion interactive et non une démarche *a priori* de l'utilisateur (je crée mes instances et je compose ensuite). En présence d'un texte utilisant une police Multiple Masters, le logiciel*, devrait, pour toute commande compatible, être capable de créer l'instance appropriée « au vol », de la modifier, et l'enregistrer comme commande de création d'instance dans le document, celle-ci étant interprétée par l'environnement d'affichage ou d'impression.

* Il n'est pas nécessaire que ce soit le logiciel qui prenne en charge cette gestion. En fait, il serait même préférable que ce soit le rastériseur (ATM ou Display PostScript pour l'affichage, PostScript pour l'impression) qui s'en charge, ce qui permettrait d'utiliser ces fonctions dans d'autres contextes.

Par exemple, la réduction ou l'augmentation de corps provoquera la création ou la modification de la ou des instance(s) utilisée(s) pour la composition du texte sélectionnée. De même, pour une commande de modification de chasse. Et, pour la graisse, en plus de l'échelle de graisse évoquée dans la section I.1.a, on disposera de la possibilité de créer des graisses intermédiaires, par saisie d'un pourcentage, par exemple.

Il s'agit là du minimum vital, les fonctionnalités standards de Multiple Masters.

Examinons maintenant la gestion avancée.

En effet, on peut avancer que Multiple Masters pourrait être la solution pour pallier l'absence de polices Expert et le refus d'Unicode d'intégrer des variantes glyphiques.

Petites capitales/exposants/indices

Il doit être possible de générer automatiquement des petites caps typographiquement correctes par calcul d'une instance basée sur la hauteur d'œil des bas-de-casse, le corps optique et la graisse de celles-ci. De même pour les exposants et les indices.

Composition scientifique

La composition scientifique fait appel à des notations qui pourraient être gérées efficacement par la création d'instances appropriées. Citons, en vrac, les accolades, crochets, parenthèses de matrices, racines carrées, qui s'étendent sur un nombre variable de lignes, mais doivent conserver la même graisse absolue, les notations d'angles ou de vecteurs, qui comportent des signes « étirés » au-dessus ou au-dessous d'une expression, etc.

T. BOUCHE — il faut pouvoir débrayer les automatismes dont tu parles. Par exemple, je peux avoir envie de faire un effet spécial avec le dessin prévu pour 6 points de Minion dans un titre. Cela dit, ça peut se résoudre juste avec le nom. Dans le système actuel, quand tu utilises MinionMM en 6 points, c'est l'« instance par défaut » qui est descendue en 6 points. Il me semble évident que ce devrait être la version 6 pt qui devrait être utilisée. Toujours dans le système actuel, tu peux utiliser MinionMM_345wd_555wt_6op comme n'importe quelle SM, donc dans le corps de ton choix pourvu que tu aies créé l'instance dans ATM *a priori.* Il me semble que rien n'empêche les deux systèmes de cohabiter.

> O. RANDIER — Tout à fait d'accord. L'idée est que, par défaut, un enrichissement provoque l'application de l'instance appropriée, sauf si c'est spécifié par l'utilisateur. Mais il me semble important que ce soit la création d'instance qui prime, sinon à quoi ça sert que Ducros y… ? J'avais d'ailleurs l'intention de faire un petit quelque chose sur tout un tas d'automatismes débrayables, qui pourraient être très éducatifs, comme de limiter l'étroitisation monstrueuse (par exemple, si un novice étroitise de plus 10 %, un signal bloque la fonction, puis, s'il insiste, une alerte demande confirmation).

h) Formatage des « paragraphes »

O. RANDIER — D'une façon générale, on regrette de ne pouvoir placer quelque chose d'autre qu'un filet tout autour d'un paragraphe ou de lignes. Notamment, la gestion des guillemets de second rang selon les règles typographiques françaises est toujours impossible :

Par exemple : Bla bla bla bla bla bla bla bla bla bla bla mot pour bla : « Bla bla bla bla bla bla bla bla bla et il a dit : « Mon père disait que bla bla bla bla bla bla bla bla bla bla bla « bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla « bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla « bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla « bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla », ce qui prouve que son papa blaguait bla bla bla bla bla bla bla bla bla. » Bla bla bla.

Dans le même esprit, il faudrait pouvoir sortir de la justif' pour placer des éléments quelconques (puces sortantes, blocs, filets, etc.) liés au texte. Ceci étant un attribut de la ligne plutôt que du paragraphe.

> A. HURTIG — Le problème, c'est simplement de pouvoir mettre automatiquement un ou plusieurs signes, à chaque fois identiques, dans la police, le corps, avec les attributs que l'on veut, au début de chaque ligne. Le guillemetage continu ne vient ici qu'à titre d'exemple, à mon avis.
>
> NB : il ne s'agit pas, dans mon esprit, de gestion de lettrine. Mais pour les lettrines on a aussi besoin de pouvoir déterminer la police et les attributs (il faut que ce soit automatique, pas à la main !), et on a aussi besoin de dire au logiciel : voilà, au début de chaque paragraphe, tu me mets telle lettre de telle police, systématiquement, sans que j'ai besoin de te le rappeler (typiquement : une puce).

I – TYPOGRAPHIE

I.2 – Typographie complexe

a) Langages scientifiques

P. PICHAUREAU — D'expérience, je pense que c'est une erreur de mêler l'aspect texte scientifique à ce projet. Et croyez-moi, ça me coûte de dire ça.

> E. CURIS — D'expérience, je ne suis pas d'accord. Il n'y a pas de raison que l'on s'en prive : je vois deux aspects dans les textes scientifiques.
>
> 1. Tout ce qui est formules, croquis, cartes, … À part les formules de math et de physique, un import depuis un logiciel extérieur spécialisé me semble ce qu'il y a de plus simple à faire : il faut en fait réclamer une bonne gestion des imports extérieurs et la possibilité d'en faire ce que l'on veut : corrections *in situ* (très pratique quand on s'est aperçu que l'on a oublié une double liaison : -) pour les petites choses, insertion dans le texte comme on veut [image fixe ou se déplaçant avec], possibilité de faire des renvois vers ça, …
>
> > T. BOUCHE — La « solution » que tu évoques, qui est équivalente au schéma de travail qu'avait décrit Olivier ici même, revient à demander à TeX de composer des EPS qu'on inclurait ? Que se passe-t-il si on change de fonte, de corps ou de maquette ? Non, il faut que ce soit beaucoup plus intégré que ça !
>
> E. CURIS — L'idéal c'est aussi d'avoir quelques outils pour faire sur place les trucs simples (pas besoin de lancer un éditeur de formules lourd pour un bête benzène non substitué…) permettant de se faire des bibliothèques.
>
> 2. Tout ce qui est gestion des index, notes, tables des matières, renvois et autres. Là, à mon avis, c'est utile pas seulement pour les sciences [j'entends par science aussi bien sciences « dures » que « littéraires »] et je ne vois aucune raison de ne pas l'inclure : c'est quand même des outils de base et qui, au moins dans une version de base, ne me semblent pas si difficiles à inclure si le cœur du programme est bien fait.

P. PICHAUREAU — Mais il faut voir que la compo de texte scientifique est quelque chose de très particulier, lourd à mettre en œuvre, difficile à gérer par la suite. C'est un peu comme si on demandait de gérer la composition des partitions musicales.

> E. CURIS — Ben oui, pourquoi pas : -)
>
> Sérieusement, la compo. scientifique est quand même relativement textuelle : elle s'insère donc beaucoup plus dans le cadre d'un programme de mise en page « classique ».
>
> Même si elle peut faire partie de modules spécialisés, optionnels.

P. PICHAUREAU — Il se trouve qu'en science une bonne solution existe déjà,

E. CURIS — Oui, mais qu'est-ce qu'elle est lourde ! :-) Et en plus elle est fortement adaptée aux maths et à la physique (théorique), mais nettement moins convaincante pour la chimie et la bio à mon goût (avis très personnel, inutile de lancer une guerre : -).

T. BOUCHE — […] À supposer que ta « bonne solution » soit (la)tex, j'abonde encore moins. D'un point de vue strictement typographique, TeX, en tant que moteur de composition, est ce qui se fait de plus sophistiqué pour les maths. Point. Les sources sont lisibles par tous : à eux de s'en inspirer pour faire mieux (et Dieu sait si on peut faire mieux, en termes de souplesse d'utilisation, de normalisation et de paramétrabilité débrayable). On peut donc imaginer qu'il existe des bibliothèques modulaires apportant certaines fonctionnalités supplémentaires (on peut penser à : automatisation du recours aux MM, fonctions mathématiques, tables, index… ne pratiquant pas les plug-in je ne sais pas s'ils sont totalement intégrés comme ce que j'imagine ici).

En tant que programme de mise en page, TeX est une abomination : il faut être ingénieur troisième dan pour définir une maquette, fêlé du ciboulot pour installer une nouvelle fonte, sous perfusion internautique pour débuter seul. À part le monsieur de Quixote Digital Typo, je ne connais pas d'authentique typo qui maîtrise TeX. Donc je mets immédiatement dans le cahier des charges : être capable de faire des formules aussi bien que tex, et de pouvoir les placer, etc. avec la même interface graphique.

P. PICHAUREAU — Évidemment, l'idée de module à la Xpress est bonne, même si il y a des risques de dérive commerciale.

N'empêche que l'aspect scientifique ne me semble pas fondamental : en clair, un module Science oui, bien sûr, mais, dans le noyau, un strict minimum.

T. BOUCHE — je ne crois pas que ça tienne debout. Comment mettre en place un « module Science » sans avoir dans le noyau la possibilité de placer les caractères un peu n'importe où verticalement, de composer des blocs à partir de sous-blocs non juxtaposés horizontalement, d'avoir des notions de positions relatives séparément pour les indices, les exposants (penser aux intégrales) ? Et une fois tout ça dans le noyau, pourquoi ne pas s'en servir ?

P.-S. : une idée qui n'était pas encore apparue : possibilité de non-Wysiwyg pour paramétrer finement dans des conditions critiques (notion de fenêtres *pop-up* acceptant un langage de commande). Ça pourrait s'étendre à rentrer des coordonnées pour des alignements parfaits, chan-

ger des attributs ou des paramètres sur un objet, etc. comme on le voit déjà dans des programmes de dessin. Qu'en dîtes-vous ?

E. CURIS — Que ça me semble faire partie des fonctionnalités de base. Donc effectivement, à ajouter immédiatement sur la liste.

À quel point puissant (à part « le plus possible » :-) :

— position d'un cadre par ses coordonnées absolues

— positionnement relatif de cadres (du genre « créer un cadre qui soit la copie de celui actif, mais décalé de 2 cm, élargi de 3 et plus haut de 2 »)

— positionnement relatif ET conservé en mémoire → qui modifie l'un si on modifie l'autre si on prend l'exemple d'un cadre (bloc selon la terminologie Xpress ?)

Mais je ne vois pas trop en quoi c'est « non-Wysiwyg » : ai-je bien compris l'idée ?

T. BOUCHE — Non Wysiwyg au sens non-étymologique que ce terme a pris aujourd'hui : un truc intuitif où « je n'obtiens que ce que je peux produire à l'écran à l'aide d'une souris et de raccourcis clavier ». Les programmes de dessin vectoriel (en tout cas ceux de création de fontes) permettent de placer les points de contrôle à la souris, mais aussi par exemple de préciser un angle de façon exacte en ouvrant une fenêtre dans laquelle on rentre un nombre.

T. BOUCHE — mon idée initiale, qui était de pouvoir ouvrir une fenêtre non Wysiwyg dans laquelle je tripatouille le ventre de la bête, conçue et énoncée pour le problème de formules mathématiques, « pourrait s'étendre à… » pas mal d'autres situations, comme alignements ou positionnements exacts, copie etc.

E. CURIS — Et puis le « ça pourrait s'étendre à… » me gêne aussi : pour moi c'est la base, à partir de laquelle on peut faire de plus en plus puissant, c'est pour ça que je m'interroge sur ma compréhension… (Enfin, en fait si, je devine un peu pour le texte : ce serait par exemple une fenêtre où le texte apparaît brut, indiquant tous les codes de contrôle permettant de modifier l'affichage (changements de style, espaces tordues, approches modifiées,...) et autorisant le paramétrage fin de tous ces codes. C'est ça ?)

T. BOUCHE — Oui, ou une boîte de dialogue donnant accès aux paramètres que la feuille de style détermine pour un mot donné, et qu'on veut changer pour ce mot.

A. HURTIG — On peut imaginer encore plus rusé (avec une bonne intégration, ça ne ferait pas trop usine à gaz).

Mon idée est qu'un éditeur de maths doit générer un EPS (en fichier séparé ou intégré dans le corps du document, je m'en

fiche), cet EPS étant inséré dans le flot du texte. Xpress comme PageMaker savent faire ça : un « bloc image » est inséré dans le texte, et le logiciel considère ce bloc comme un « caractère ».

T. BOUCHE — Mon souci, face à ça, c'est de ne pas introduire un bastion rigide « image », il faut que la typo reste cohérente, qu'il y ait une sorte d'héritage du corps par défaut du texte sur les paramètres en math, par exemple, il faut aussi qu'une formule tapée au kilomètre dans le texte, ne soit pas rendue de façon incohérente avec sa version isolée en formule centrée. J'aimerais aussi qu'une fois la formule produite, je puisse éditer des détails directement dedans sans appeler un module. Imaginons que j'ai écrit une formule du genre tg(3x+2) que je voudrais changer en tan(3x+2) : j'aimerais pouvoir le faire directement. Bon, ça, c'est un détail, ça dépend de la difficulté de la mise en œuvre, mais le fait que la typo des maths vive au même rythme que celle du texte est en revanche crucial.

O. RANDIER — Je suis parfaitement d'accord avec Thierry : la démarche par génération d'images importées est potentiellement source de problème : comment gérer le trapping dans les EPS ? Si on change la fonte utilisée, va-t-il falloir refaire toutes les images ? Comment va se faire le placement, par exemple, de la barre de fraction d'une formule importée, par rapport au deux-points du texte courant ?…

A. HURTIG — Il y a maldonne ! Je ne dis *pas* qu'il faut importer de l'EPS, je dis que je m'en fiche. Le logiciel se débrouille pour construire ses formules, les retrouver, en modifier les caractéristiques si on change un paramètre.

Qu'il les stocke dans le document lui-même ou qu'il pointe sur un fichier externe n'est pas de mon ressort (même si je préfère la première solution, plus simple à gérer pour l'utilisateur).

O. RANDIER — Dans ma brochure Texas, il y avait 400 imports ; avec les formules, j'en aurais eu combien ? 2 000 ?

Après ma petite expérience avec MathX, je suis convaincu que le texte doit rester du texte, fut-il trituré par postscript. Ca me semble possible, surtout si le logiciel est capable de gérer des créations d'instances MM ou des commandes de déformations complexes de caractères comme celles cités par Jacques dans sa thèse. Je pense qu'on peut demander ça aux créateurs de Postscript.

A. HURTIG — En EPS, on n'a pas du texte ? Je ne comprends pas… En tout cas, *mon* expérience avec XMaths (un plug-in de XPress) qui se contente d'utiliser du texte (en étroitisant, parangonnant, bref en faisant tout un bordel à partir des fonctions standard du logiciel « hôte ») est que ça marche mal, que c'est lourd à gérer, que les résultats ne sont satisfaisants qu'à

force d'un énorme travail, et que la correction d'une formule est un calvaire.

Mais bon, cela dit, l'essentiel n'est pas là, pas pour moi en tout cas. L'essentiel est que nous disposions d'un *add-on* évolué, qui marche bien et vite, et qui soit correct typographiquement.

A. HURTIG — Un double-clic sur le bloc en question suffirait à ouvrir l'éditeur mathématique et scientifique.

Celui-ci comprendrait :

1. Une interface graphique pour monter les formules, à l'exemple de l'excellent MathType (qui existe sur Mac et PC, et est surtout connu comme complément de Word), ou du moins excellent FrameMaker (un logiciel Adobe, soit dit en passant).

L'écriture d'une formule est absolument magique : on tape simplement des commandes au clavier (ou on les appelle dans les menus), et la formule se monte et se démonte en temps réel, se construit ou se corrige à la volée.

2. Un paramétrage typographique fin, avec feuilles de styles (afin de pouvoir conserver ses paramètres et les réappliquer ou les modifier d'une formule à l'autre, d'un document à l'autre). Faut-il ici entrer dans les détails de ce paramétrage ? Thierry en a donné la plupart des éléments, je crois, mais on peut faire une recension.

T. BOUCHE — Ce qui serait important, là, ce serait de recenser les attributs et fonctions typographiques *supplémentaires* par rapport à ce qu'a tout programme de typo. Si on n'en dispose pas dès le départ, la suite ne sert à rien. Une grande erreur (typique) de la plupart des programmes qui composent les maths comme du texte, c'est qu'ils gèrent très mal les indices et les exposants : dans une intégrale, l'« indice » doit être beaucoup plus à gauche que l'« exposant », mais c'est pareil pour une lettre italique, il y a aussi le problème de placer (verticalement et horizontalement) en exposant une formule déjà complexe, de prévoir un espacement différent pour une même lettre selon ce qu'elle représente.

O. RANDIER — C'est exactement ce que je voulais dire quand j'avais commencé à aborder l'idée d'architecture ouverte de styles. Mais, contrairement à ce que vous pensez, je crois que le travail de recensement n'est pas si important que ça, si on admet le fait que ces attributs peuvent être cumulatifs. Le problème du placement des indices et des exposants relève plutôt de la gestion du crénage (via le crénage par enveloppe).

A. HURTIG — 3. Une interface texte (macro-commandes, tags, appelons ça comme on voudra) qui permette d'effectuer des modifications fines si nécessaire. Est-ce à cela que vous pensez ?

T. BOUCHE — Moi oui. Pour moi 1. et 3. sont deux facettes pour entrer l'information à mettre en forme, qui peuvent être indépendantes et auto-suffisante. Par expérience, je sais que c'est plus facile pour un mathématicien de taper \int_a^b que d'aller dans un éditeur d'équation, reconnaître la forme graphique de la formule, cliquer 20 000 fois, etc. Je sais aussi qu'ils sont à peu près seuls dans ce cas. Ce dont on parle permettrait de faire cohabiter sans heurts les deux approches. Les feuilles de style qui interviennent à un autre niveau, sur les paramètres de formatage, relèveraient d'un maquettiste.

O. RANDIER — Je vais essayer d'orienter le débat, puisque c'est moi qui suis censé le gérer. Je pense qu'il faut distinguer plusieurs niveaux du problème.

1. La gestion avancée du texte, qui comprend la possibilité de faire subir à des signes des modifications un peu plus complexes que ce qu'on fait d'habitude (c'est-à-dire des applications linéaires, si j'ai bien compris un des articles que j'ai composé dans ma brochure Texas).

Cela implique sans doute l'introduction de certaines routines postscript, soit pour effectuer les calculs nécessaires, soit pour introduire des formats de fontes susceptibles de les accepter (Multiple Masters, Math-Fly). De ces routines, on compose des « styles » applicables par une simple commande (par exemple, le style « en vecteur » applique les déformations nécessaires pour composer un signe — par exemple une flèche — au-dessus et tout le long d'une expression).

H. RICHARD — Attention, c'est bien de penser à PostScript mais il ne faut pas oublier qu'avant de cracher du code postscript, les applications vont devoir afficher une *preview* à l'écran utilisant le moteur graphique de MacOS ou de Windows, moteurs qui n'ont pas les mêmes capacités que PostScript… Moteur qui utilisent par ailleurs des polices TrueType, c'est-à-dire non gérée par ATM.

PageMaker a dans sa palette texte un style contour, mais uniquement sur Mac. Pourquoi ce style est-il manquant sous Windows ? Visiblement, la facilité de fabriquer une police contour en PostScript ne leur a pas été d'un grand secours…

O. RANDIER — On peut déjà rapidement recenser les « styles » de base indispensables et suffisants dans la plupart des cas ; je suis convaincu qu'il n'y en a pas tant que ça. À noter que certains de ses styles seraient déjà utiles et nécessaires pour la composition courante, je pense, par exemple, aux accolades ou crochets extensibles sur plusieurs lignes.

Cette architecture doit être ouverte, et ses spécifications facilement accessibles, dans l'optique des plug-in Illustrator, par exemple, pour pouvoir ultérieurement introduire facilement des routines ou des « styles » qui auraient été négligés.

2. L'éditeur de formules, qui, comme l'ont suggéré Thierry et Alain, pourrait avoir deux facettes :

— La gestion typographique, qui n'est selon moi qu'une extension des concepts qui existent déjà (feuilles de styles spécifiques, crénage élaboré).

— La saisie de formules selon l'habitude des scientifiques. Là, selon moi, c'est essentiellement une question d'interface. Et il n'est pas absolument nécessaire qu'Adobe en soit le maître d'œuvre, ce peut très bien être un plug-in de tierce partie, l'essentiel étant que les routines qui l'autorisent soient présentes. Là, on peut tout imaginer, du simple éditeur en mode texte, jusqu'à la passerelle vers un logiciel de maths. Même si Adobe décide d'intégrer un super éditeur de formules, il sera de peu d'usage au typographe qui travaille à Marie-Claire ;-)

O. RANDIER — Bon, je vais commencer sur cette histoire de recensement des « attributs de style » supplémentaires, nécessaires pour les maths. Je précise que je ne connais rien aux maths, je parle juste de ce que j'ai eu l'occasion de voir, notamment dans la thèse de Jacques.

Je distingue trois parties :

Les attributs simples

Certains existent déjà, je les cite pour mémoire, et pour préciser quelques détails.

• Gras

Pas utilisé en math, je crois ?

> M. BOVANI — Ben si, hélas… […] Par ailleurs certains éditeurs (Masson) font du gras (en maths) un usage assez systématique.
>
> E. CURIS — Il y a (eu ?) une mode de noter les grandeurs vectorielles en gras, plutôt qu'en utilisant les classiques flèches…
>
> Sinon, le gras intervient en chimie pour numéroter les molécules, mais je ne pense pas que ce soit suffisamment spécifique pour nécessiter quelque chose de plus que le gras normal.

• Italique

Variables. Ne doit pas être affecté par une modification générale du texte.

• Exposant/indice simple

Ceux que l'on connaît, voir les remarques déjà faites concernant la section I.1.a

Les attributs secondaires

Les attributs suivants ne réclament pas d'opérations de déformation complexes, mais établissent une relation entre une sélection « flèche » et une sélection « cible ».

• Exposant/indice relatif

Affecte la sélection comme exposant de la cible, même s'il y a d'autres caractères entre les deux. Permet d'affecter simultanément un exposant et un indice à une expression, comme, par exemple, les poids et numéro atomiques d'un élément, ou les indice et exposant d'une intégrale. Peut donc se faire à gauche et à droite.

• Adscrit/souscrit

Place la sélection sous/sur la cible (transformée, dérivée, exposant et indice de sommes ou de produits).
• Haut/bas de fraction
Intercale un filet entre la sélection et la cible, et la souscrit/adscrit à celle-ci.
• Haut/bas registre simple
Déforme la sélection au-dessus/en dessous de la cible (segment, angle, barre de racine… pas d'idée d'exemple pour en dessous).

E. CURIS — Segment = mesure algébrique je pense :-)

O. RANDIER — J' sais pas, c'est quand on met une barre au-dessus de quelque chose. Dans le code de l'I.N., il donne comme exemple :

$\overline{AB}$

E. CURIS — C'est donc bien la mesure algébrique (le segment, c'est [AB]) Logiquement, ça peut être fait par du soulignement bien réglé (dans l'hypothèse où les fonctions nécessaires sont implémentées). Sauf si on veut des mesures algébriques soulignées, mais là bof :-)

Pour la barre de racine, je me demande s'il vaut mieux faire comme ça ou avoir un signe complet qui se déforme (même si, en interne, il gère ça comme ça ou pas), mais alors pas du tout aussi simplement ⟶ plutôt dans le registre complexe ?

O. RANDIER — À mon avis, il vaut mieux s'en tenir au filet placé au-dessus du contenu de la racine. C'est quand même plus simple que de demander des contorsions complexes à un signe unique (déformation non linéaire verticale d'une partie du signe par rapport au registre droit + déformation linéaire horizontale d'une autre partie du signe par rapport au registre inférieur). Ceci dit, ce n'est qu'une variante de ce qui ce ferait pour une somme ou un produit, donc ce doit être faisable (et plus satisfaisant intellectuellement). Mais de deux solutions donnant le même résultat, il vaut mieux choisir la plus simple.

E. CURIS — La raison pour laquelle je demandais ça, c'est que séparer le radical en deux morceaux (radical avant + filet) risque de compliquer le positionnement vertical du filet pour qu'il colle avec le radical, quand par exemple le filet doit remonter à cause d'exposants. Il me semblait justement plus simple d'avoir un seul truc qui s'adapte comme il faut.

Ceci dit, au fond, c'est aux programmeurs après de décider comment l'implémenter, du moment que l'utilisateur peut contrôler.

• Registre* gauche/droit simple
Déforme la sélection à gauche/à droite de la cible (matrice carrée, valeur absolue, norme).

E. CURIS — J'ai un peu de mal à saisir la différence que vous faites entre cet effet et celui des valeurs absolues et normes (ce qui fait que je laisse mon commentaire, même s'il est caduc…)

O. RANDIER — Ben justement, ça s'applique à ce que tu demandais concernant les matrices : La distinction correspond à des signes qui ne peuvent être facilement déformés de façon linéaire pour obtenir le résultat escompté, au contraire des barres de valeur absolue, normes, matrices carrées. Une barre étirée verticalement reste une barre, une accolade déformée verticalement ou agrandie d'autant est une abomination typographique. Pour obtenir un résultat correct, il faut avoir recours à des méthodes de calculs plus complexes (interpolations style Multiple Masters). Pour plus de précisions, cf. la thèse de Jacques. En fait, il n'y a pas lieu de distinguer les deux attributs au niveau de l'interface, je n'ai précisé la chose que pour spécifier la façon dont le programme devrait gérer cela.

E. CURIS — Pour les matrices, il faudrait quelque chose qui dise que ça s'applique sur plusieurs lignes, comme les accolades, les parenthèses (cas des matrices justement) ou les barres verticales (déterminant). Pour la chimie, ce serait bien d'avoir aussi des crochets qui s'adaptent à la hauteur de la ligne, pour faire des trucs du genre :

$$\left| \textit{Complexe} \right|^{2+}$$

où complexe est la formule d'un truc quelconque (donc *a priori* une image importée et insérée dans le texte, parce que sinon je ne vois pas trop comment faire sans chambouler toute l'architecture usuelle d'un programme de PAO)

O. RANDIER — Mais, justement, le sujet est bien de chambouler toute l'architecture ;-) T'avais pas compris qu'il s'agissait d'un jeu de massacre ?

Attributs complexes
Nécessitent des déformations complexes de la sélection en fonction de la cible.

• Haut/bas registre complexe
Déforme la sélection non linéairement** au-dessus/en dessous de la cible (arc, vecteur… pas d'idée d'exemple pour en dessous, non plus).

M. BOVANI — On utilise assez souvent des accolades horizontales « extensibles » en dessous d'une expression (par exemple pour donner un

nom à cette expression, ou dans un but explicatif) : certains n'aiment pas cela, et d'autres en usent et en abusent…

E. CURIS — En chimie, on utilise (utilisait faute de facilité à le faire :-) la notation

Pour indiquer qu'un composé précipite lors d'une réaction (et la même chose mais au-dessus pour un composé qui se dégage sous forme gazeuse)

• Registre gauche/droit complexe
Déforme la sélection non linéairement à gauche/à droite de la cible (racine, intégrale, accolade, parenthèse, crochet, matrice).
• Selon registres
Déforme la sélection non linéairement par rapport à ses adscrit/souscrit (somme, produit).

* Registre n'est peut-être pas le terme le plus approprié, désolé.

** Les déformations non linéaires peuvent se faire selon deux principes : interpolation (générations d'instances à la MM [accolade, parenthèse…]), translation le long de segments horizontaux ou verticaux (MathFly [vecteur, produit…]).

Voilà, ce n'est sans doute pas exhaustif, à vous de compléter, mais franchement, c'est tout ce que j'ai trouvé en feuilletant la partie maths du Code de l'I.N. et la thèse de Jacques. En admettant que ces attributs sont cumulatifs, il me semble que cela devrait permettre de traiter la majeure partie des problèmes de la composition des maths (il faudra sans doute revenir en deuxième semaine pour la chimie).

E. CURIS — Pas forcément, sauf si vous tenez à pouvoir dessiner des formules développées directement avec les contraintes que ça implique. Mais là, j'ai peur que ça demande de repenser complètement la gestion du texte, parce que l'aspect « ligne » n'a plus beaucoup de sens… Je ne suis pas sûr que le jeu en vaille la chandelle. Pour le reste, à part la liste des symboles étranges et les exposants/indices avant ou sur plusieurs niveaux, je ne vois rien de particulier.

Sinon, une question en passant : quel statut demander pour l'alphabet grec dans les maths et la physique ? Ce qui me semble le plus logique, c'est d'avoir une fonte mathématique sur deux octets qui contienne cet alphabet en plus des signes spéciaux [enfin, une fonte = un type de fonte], donc un fonctionnement tout à fait distinct de celui de l'alphabet grec en tant qu'alphabet de langage — bien que cela paraisse redondant dans le cadre d'un programme Unicode dont les fontes de texte contiendraient aussi l'alphabet grec.

Si on va par là, on va me dire qu'il faudrait faire la même chose pour les lettres de l'alphabet latin qui servent de noms de variable.

Je répondrai donc : pourquoi pas, ca résoudrait élégamment le problème de toujours devoir passer en italique (et conserver le style et tout et tout) et éviterait peut-être, en cas de fonte « française », de voir des capitales en italique (TeX par défaut :-()

Et, tant que l'on y est, y inclure les notations usuelles des constantes fondamentales :-)

Qu'en dites-vous ? Est-ce trop utopique ?

M. BOVANI — Je pense que ton point de vue est résolument descriptif : c'est sans doute le mieux pour un cahier des charges, mais peut-être qu'une petite composante structurelle permettrait de jouer plus finement (la remarque de Thierry concernant la différence d'espacement qui peut intervenir pour un même signe suivant sa fonction dans la formule illustre finalement assez bien ce que je veux dire par là).

M. BOVANI — Il me semble qu'il n'a pas été question des problèmes de ligne de base :

$$x = \frac{\frac{a}{b}}{c+1} \quad \text{ou } x = \frac{a}{\frac{b}{c+1}} \quad \rightarrow \text{ligne de base}$$

Naturellement tout éditeur de formule permet de régler la plupart des problèmes de ce type de façon implicite, mais il y a des cas où…

$$x = \left| \frac{\frac{a}{b}}{c+1} \right| \quad \text{ou } x = \left| \frac{\frac{a}{b}}{c+1} \right|$$

Autrement dit, les délimiteurs (parenthèses, accolades, barres de valeur absolue…) doivent-ils être centrés sur la ligne de base ? L'expérience prouve que c'est parfois oui parfois non et que cela ne dépend pas seulement de la nature du délimiteur. Il faudrait donc prévoir une fonction du genre « ajuster la hauteur ». (L'expérience prouve aussi que si une telle fonction n'est pas prévue, c'est la croix et la bannière pour obtenir ce que l'on veut.)

A. HURTIG — Ça me semble un peu court, mais là je n'ai pas le temps d'aller regarder dans mes docs.

Je me propose au demeurant de faire le travail à l'envers, en fonction des paramétrages…

> … J'ai commencé, en allant relire le manuel de MathType (au fait, il ne génère pas que de l'EPS, mais aussi du code TeX, ce truc-là). Et je me suis arrêté, écrasé sous l'immensité de la tache et surtout convaincu de l'écrasante inutilité de ce boulot.
>
> Après tout, ce n'est pas à nous de définir comment un module de maths doit être paramétré, ni quelles doivent être ses fonctions. Il nous suffit d'expliquer qu'on veut qu'il soit paramétrable dans tous les sens, que sa gestion typo doit être nickel, et grosso modo comment on veut qu'il marche. Pour le reste, que le chef de projet se débrouille…

T. BOUCHE — Je signale aux intervenants sur ce sujet la publication du standard MathML. Ils ne retiennent que 3 types d'objets : identifieurs, nombres et opérateurs [typiquement, les identifieurs (?) sont en italique et munis d'approches corrigées, les nombres en romain, les opérateurs entourés d'espaces] les délimiteurs sont gérés de façon automatique.

Un minimum, pour K2, serait d'importer et d'exporter du MathML, de tenir compte de [et de permettre de contrôler] tous les paramètres déjà prévus, tout en offrant une interface conviviale et une correction automatique d'erreur dans les démonstrations. S'ils savent faire ça, je pense qu'on pourra leur faire confiance pour le reste…

Sur les problèmes de codage, il y a trois initiatives en cours, Unicode qui contient déjà beaucoup de symboles mathématiques, « Unicode glyph registry » qui devrait contenir tous ceux constatés dans des documents publiés (y compris quand c'est seulement les approches qui changent : deux signes identiques munis d'approches différentes sont des glyphes distincts $\longrightarrow$ corriger mes précédentes remarques à ce sujet).

MathML devrait s'appuyer là-dessus, mais va probablement devoir introduire de nouvelles « entités ».

La « communauté » TeX travaille à de nouveaux codages sur 8 bits (dans la fonte principale, on a un alphabet latin italique dont les glyphes sont corrigés pour un usage mathématique, et deux alphabets grecs [droit et italique]). Pour une fois, ça ne se fait pas dans une tour d'ivoire : un autre projet mené par les principaux éditeurs scientifiques devrait converger si bien que ces codages devraient aboutir à un standard de fait, donc tentant pour les fondeurs. Par exemple, ces codages contiennent un e, un i, un d pour (respectivement) noter la base de l'exponentielle, la racine carrée de moins un, et les éléments différentiels. Si j'ai bien compris, les Américains les utilisent italiques, les Français utilisent les deux premiers droits (constantes!) tandis que les Allemands ou les physiciens les utilisent tous droits

Une autre info, c'est que des gens qui travaillent à un successeur de TeX se disent prêts à lui donner une architecture de type bibliothèque avec fonctions de typo mathématiques appelables par un tiers (la façon selon laquelle hz était sup-

posé fonctionner). Une bibliothèque de fonctions de typo plus paramétrable que TeX, et *gratuite*! ça pourrait motiver les bâtisseurs de logiciels du côté de la plus-value pas cher payée?

Je crois comme Alain qu'il n'est pas nécessaire d'inventer des codages, de mettre à plat tout ce que doit faire un logiciel de compo scientifique. C'est le boulot des implémenteurs!

A. HURTIG — C'est exactement ça, et je crois que ça implique qu'on explique le *type* de paramètres que nous souhaitons, et non pas qu'on les liste (cette liste étant probablement non-finie, on risquerait en plus d'y mettre un bout de temps).

A. HURTIG — Si la typo est uniquement conçue pour le texte, on sera condamné à un fonctionnement « image » genre EPS importé. Si la typo permet de placer n'importe quoi n'importe où (essentiellement, comme dans la recension d'Olivier, on doit pouvoir affubler un bloc (qui sera en général un signe, mais peut aussi bien être une formule entre parenthèses) de modificateurs placés au nord, nord est, est, sud est, sud, sud ouest, ouest, nord ouest, ces modificateurs pouvant eux-mêmes être des objets complexes [affublés eux-mêmes de …]). C'est le premier point important. Le second, c'est la capacité à enjamber des lignes ou des colonnes. Le troisième, c'est la possibilité de *bien* placer les caractères (du genre paires d'approches corrigées différentes en exposant, en indice, etc.) et bien les *espacer*. Le quatrième, c'est que les polices doivent rester cohérentes avec celles du texte, mais les attributs comme gras ou italiques doivent rester fixes.

O. RANDIER — J'en rajoute une couche : le module scientifique doit pouvoir agir *a posteriori*, c'est-à-dire remettre en forme selon les paramètres choisis une formule composée vite fait « à la main » ou en détecter une lors de l'import (y compris selon la notation scientifique conventionnelle évoquée par Thierry) et la traiter correctement.

I – TYPOGRAPHIE

I.2 – Typographie complexe

b) Substitution automatique

A. HURTIG — Dans une composition soignée, le typographe peut souhaiter substituer certains caractères à ceux de la police courante.

Il s'agit en particulier des ligatures ornementales (*ffi*, *ffl*, *ſt*, *ct*, etc.), des « vraies » petites capitales, des « vrais » indices et exposants, des chiffres elzéviriens. Mais la liste ne saurait être limitative.

En raison de la limite actuelle des polices à 256 caractères*, cette substitution se fait actuellement « à la main », par « chercher-remplacer » par exemple, ou modification des attributs typographiques. Ce travail fait perdre un temps inutile au compositeur, est source d'erreurs éventuelles, et fait obstacle à une reprise ultérieure du texte (dans une autre composition par exemple), puisque certaines ligatures sont codées d'une façon qui peut sembler absurde (positions occupées par les capitales dans les polices non-expert).

Un logiciel de composition professionnel aura donc la faculté de substituer certains caractères à d'autres, de façon automatique et transparente pour l'opérateur. Cette substitution doit cependant être paramétrable sur tous les points, et débrayable sur tout ou partie d'un document. On peut la diviser en deux groupes :

1. Les modifications stylistiques.

Il s'agit ici de substituer une police de caractère par une autre (petites capitales, indices et exposants, chiffres…). Le paramétrage autorisera de désigner les caractères ou groupes de caractères devant être substitués, en fonction tant de leur codage ASCII que de leurs éventuels attributs, et la police dans lesquels ils doivent être substitués (avec indication éventuelle du code ASCII correspondant). Il ne s'appliquera éventuellement pas dans certaines parties du document, en fonction de désignations précises : tableau, appel de note (ceci pour les chiffres), à la demande de l'opérateur. Outre les cas d'exclusions qui peuvent être prévues dans la boîte de dialogue[1] du paramétrage, on prévoira un attribut spécial et invisible : « Ne pas substituer », que l'opérateur pourra appliquer à un caractère en cours de composition.

2. Les substitutions contextuelles.

> O. RANDIER — Attention, le terme « contextuelle », s'agissant de ligatures, désigne des assemblages de glyphes qui varient en fonction du contexte, et s'applique essentiellement à des langues comme l'arabe ou certaines langues indiennes dans lesquelles la forme d'une lettre va être différente selon sa place dans le mot. Les ligatures latines ne sont contextuelles que par rapport à la césure. C'est un élément stylistique, non une nécessité de la langue. Il convient donc de distinguer trois types de ligatures : — les lettres ligaturées (œ, æ etc.), — les ligatures stylistiques ou techniques (*ct* est une ligature stylistique, *fi* est une ligature technique), — les ligatures contextuelles (arabe, etc.). Ceci pour clarifier un point qui semble constamment poser problème, y compris dans les débats sur les normes.

A. HURTIG — Il s'agit évidemment des ligatures ornementales (auxquelles on peut ajouter l'éventuelle substitution du « s » en « f » non barré). Elles ne peuvent être que contextuelles, en raison des césures : on ligaturera par exemple {ffi}, que le logiciel aura éventuellement besoin de diviser en [f-{fi}].

Un paramétrage permettra de pointer sur la ou les polices à utiliser pour les substitutions. La liste des groupes de caractères à substituer sera ouverte, afin de permettre aux créateurs de caractères de dessiner des lettres ligaturées non standardisées.

> O. RANDIER — Globalement d'accord avec ce que tu dis. À noter que l'interfaçage avec Unicode permettrait de simplifier grandement tout ça. Dans mon idée, il faudrait pouvoir faire pour tout un tas de signes ce qu'XPress fait très bien pour les deux malheureuses ligatures (fi et fl) présentes dans le codage Mac standard (sauf qu'on aimerait pouvoir le faire localement, et pas pour tout le doc). Dans Unicode, il y a tout ce qu'il faut pour cela (indication de la décomposition des signes, de leur translittération).
>
> C'est pour cela que, lors d'un autre débat, je militais pour l'inclusion des petites caps dans Unicode, parce qu'à mon sens, la translittération est le moyen le plus simple d'obtenir ce genre de résultat. J'y reviendrais dans le chapitre sur la gestion multilingue, que j'envisage saignant.
>
> Et ce qu'Unicode ne gère pas, WorldScript peut le faire.

A. HURTIG — * Cette barrière semble devoir demeurer pendant quelque temps, GX (sur Macintosh) ayant rencontré un insuccès commercial remarquable, et Unicode (qui n'intègre d'ailleurs pas les petites capitales ni certaines ligatures ornementales) ayant du mal à s'imposer.

> F.H. VILLEBROD — Alain Hurtig vient juste d'aborder avec grande clarté le sujet de la substitution automatique — mais aussi manuelle. Cette nouvelle fonctionnalité qui nous arrivera avec les fontes OpenType* de qualité va satisfaire bon nombres des demandes avancées par l'enquête K2, en particulier pour ce qui est des « bizarreries » et même des affaires d'espace et de justification.
>
> Il fallait donc en parler, d'autant plus que cette enquête bénéficiera aux logiciels de la prochaine génération, lorsque OpenType, justement, commencera à battre son plein et que les logiciels d'édition devront en tirer profit de la manière la plus efficace.

A. HURTIG — Ouh là ! Je suggère que nous n'entrions pas dans cette problématique, afin de ne pas lier notre réflexion et nos propositions à un futur (et par voie de conséquence hypothétique) format de polices. Open Type ou pas, voilà ce dont nous avons besoin ! De substitutions, et de toutes sortes d'autres choses.

Et puis, personne ne sait encore comment OpenType, si OpenType il y a, sera géré par les OS de nos bécanes, ni comment on va se débrouiller avec les polices que nous avons déjà et qui sont au format PostScript classique.

T. BOUCHE — Mais c'est tout de même mieux que ces choses se trouvent dans les polices. Si le programme est prêt à réagir à ce qu'il trouve dans la police, on peut tout imaginer. À l'heure actuelle, les programmes me semblent dépendre énormément de standards de codage (c'est un problème inhérent au Wysiwyg, je suppose) : si Courier n'a pas l'absurde ligature <fi> au bon endroit, un changement de fonte (ou une erreur postscript ;-) va rendre illisible le chef-d'œuvre… Ceci parce que le programme va demander un code et pas un glyphe, …

Je tente d'être plus clair : j'utilise pour ma part constamment les fontes virtuelles de TeX, que je code conformément à ISO-latin pour les lettres utiles en français, et je mets les ligatures inhabituelles à des endroits inhabituels. Le format des fontes de TeX m'autorise à mettre *dans* la fonte ce qu'il faut pour que TeX, cherchant à imprimer la séquence <c><t>, apprenne que la fonte dispose à tel endroit non normalisé d'un glyphe <ct>, et le substitue. Si je change de fonte, que ladite ligature n'existe pas ⟶ pas de substitution. Ça me semble un fonctionnement insurpassable car *seules les fontes savent ce qu'elles contiennent* ou alors on aboutit à l'absurdité de dessiner un Helvetica Unicode avec la ligature ct et un s-long…

H. RICHARD — On peut très bien savoir avec un fichier AFM, quels caractères sont contenus dans une fonte ainsi que quelles ligatures sont proposées… C'est l'applicatif qui ne fait pas son boulot. C'est tout.

T. BOUCHE — J'ai envie de dire « les fontes sont les objets de la typographie » : c'est ce qu'on ne modifie pas, nous tentons seulement de placer ceci ici ou là. C'est donc aux fontes de nous dire ce dont elles disposent. En cela le format Opentype me semble prometteur. Maintenant, il est évident qu'on apprécierait de pouvoir jouer ce genre de jeu sans avoir à racheter toute sa bibliothèque de fontes type 1…

Pour ce qui est des ligatures, le programme le plus sophistiqué est certainement TeX, ça serait peut-être utile que des gens qui ne pratiquent pas cet outil aillent voir dans *http://www.univ-rennes1.fr/pub/GUTenberg/publicationsPS/22-yannis.ps.gz* la description de ce qui existe (il y a *plusieurs* modes de substitution) et nous disent ce qu'ils pensent être utile là-dedans.

Conclusion ?

D'accord sur tout ce qui concerne les fonctionnalités souhaitées. Mais ce serait plus souple si c'était dans le format de fontes.

A. HURTIG — Ah ! mais ça paraît possible ça, sans s'embarquer dans des spéculations sur OpenType, Rhapsody et autres. Et le système de fontes virtuelles me semble correspondre à ce que nous voulons, puisqu'il n'interagit pas avec l'OS (sous réserve d'une interface commode, of course — s'il faut commencer à écrire des modules et à bidouiller, c'est même pas la peine : pensons à ceux [la plupart d'entre nous] qui sont à la production et qui n'ont pas le temps de jouer à ce genre de choses…)

H. RICHARD — Effectuer ou non la ligature au dernier moment, c'est-à-dire dans l'imprimante est dangereux. En effet une ligature peut foutre en l'air une justification (qui est généralement fixée dans le code postscript et non par le code postscript).

Il ne faut pas confondre TeX et PostScript. TeX est pour moi un applicatif dynamique. Mais il ne faut pas demander à PostScript d'avoir toutes les fonctionnalités typographiques de celui-ci. C'est au niveau applicatif de faire un effort. Une fois de plus, c'est Quark et Adobe qui sont responsables, pas l'imprimante.

T. BOUCHE — [les fontes virtuelles de TeX ont aussi une qualité énorme : la virtualité. En d'autres termes, si la bonne solution (informatique) pour choisir un glyphe contextuel dans un monde désincarné à la Unicode, est de sélectionner une fonte approprié en fonction du contexte (fonte toute caps, fonte à chiffres elzéviriens, fonte caps & petites caps, etc.) il suffit de charger une fois pour toute le réservoir de glyphes dans le postscript et d'appeler ces glyphes de façons variées (idem pour les approches variables en fonction du contexte, comme les italiques en mathématiques). La méthode TeX consiste à arnaquer le programme en lui faisant croire qu'il y a plein de fontes différentes, avec des métriques différentes, qui ne font qu'une au moment où on génère le postscript. Ça doit être plus ardu en Wysiwyg parce qu'il faut tout de même que le programme affiche quelque chose… mais je suis certain qu'un certain décollement entre « fonte logique » et « fonte réelle » permettant des recodages, des composites… peut être très utile.]

H. RICHARD — PostScript gère très bien cela avec les fontes CID qui permettent de fabriquer des fontes « recomposées » en piquant des glyphes à d'autres polices, par ci, par là.

F.H. VILLEBROD — « A. HURTIG — Un logiciel de composition professionnel aura donc la faculté de substituer certains caractères à d'autres, de façon automatique et transparente pour l'opérateur. Cette substitution doit cependant être paramétrable sur tous les points, et débrayable sur tout ou partie d'un document. »

Il s'agit de la substitution des glyphes et non pas du caractère — comme il est dit ci-dessus par habitude de langage. Le caractère de codage est préservé et le texte reste intact lors de la substitution.

Celle-ci doit effectivement rester entièrement paramétrable par l'utilisateur. Alain fait bien de critiquer plus loin les lentes et encombrantes boîtes de dialogue. Que pensez-vous d'un petit menu déroulant qui s'ouvre sous le curseur ou sous la sélection lorsque l'on presse une combinaison de touches (Macintosh) ou le bouton-souris de droite sur Windows ?

Ajoutons que le paramétrage ou le choix des glyphes de substitution doit aussi être accessible au sein des procédures de recherche et de remplacement, avec la conséquence logique que les textes des champs de saisie doivent être rendus dans les fontes qui leur sont attribuées et non pas dans la fonte-système.

Voici quelques exemples de substitutions qui pourraient être fort utiles. S'il y a d'autres substitutions qui vous intéressent, c'est l'occasion d'en faire la liste. Information précieuse en vue de l'amélioration des fontes de demain !

Quelques substitutions proposées en sus des ligatures bien connues :

— chiffres elzéviriens,
— autres styles de chiffres (trois-quart-hauts, romains, etc.),
— chiffres appropriés pour indices et exposants (par exemple la commande *subscript* va chercher le substitut au lieu de réduire le chiffre standard)
— chiffre zéro avec barre ou détrompeur (pour les codes alphanumériques où l'ambiguïté existe par exemple)
— symboles & (et) alternatifs,
— symboles @ (arrobes) alternatifs,
— choix d'espaces de différentes chasses,
— lettres alternatives, en particulier pour les débuts et fins de mots (le remplacement peut être automatique, mais toujours par décision de l'opérateur). Très utile pour une famille comme Avant Garde qui possède un grand choix d'alternatives esthétiques.

* La substitution est déjà possible avec le standard GX sur Macintosh, mais celui-ci n'a jamais pu prendre son envol, par manque de support de la part des applications, tant de mise en page que de création de fontes. Espérons qu'OpenType

fera mieux ; il a au moins Adobe et Microsoft pour moteurs principaux et je pense qu'Adobe ne va pas tarder à mettre sur le marché ses premières fontes OpenType.

A. HURTIG — J'adopte tout ça ! On doit pouvoir substituer n'importe quoi par n'importe quoi, en tout état de cause, de façon transparente et automatique.

Ça signifie, surtout pour le dernier exemple, que le contexte doit pouvoir être décrit par l'utilisateur (au moins par l'indication de ce qu'il y a avant et après la ou les lettres qu'il faut remplacer), et ceci en sus du problème de césure que je mentionnais.

I – TYPOGRAPHIE

I.1.3 Support multilingue

a) Unicode

O.RANDIER — On a déjà évoqué à plusieurs reprises Unicode et les simplifications qu'il pourrait apporter à certains aspects de notre travail (translittérations, substitutions de glyphes, etc.). C'est dans le cadre du support multilingue que ses avantages seront le plus évident. Pour ma part, ma position est claire : même si Unicode est encore loin d'être parfait (air connu), je n'échangerais pas un baril d'Unicode contre deux barils d'ISO-Latin, fût-il neuf (9).

Notre futur logiciel de mise en pages idéal devra fonctionner en Unicode natif, point barre. Quitte à prévoir des « appauvrissements » à l'export et des conversions à l'import. Il faut faire tomber le bastion 8 bits (et je ne parle même pas du 7 bits).

Ceci étant posé, Unicode ne résout pas tout. Unicode est une norme qui ne régit que les caractères (on s'en est assez plaint). Il reste à définir l'interfaçage avec, d'une part les codes saisis au clavier et, d'autre part, les divers systèmes de glyphes (c'est-à-dire les fontes). Certes, cela n'est pas réellement du ressort de l'éditeur de logiciel, mais il est difficile d'ignorer complètement le problème, et comme en plus on a chez nous quelqu'un qui s'occupe d'un gestionnaire de fontes (ATM), autant en profiter. On ne peut évidemment qu'effleurer le problème, mais je crois qu'il y a quelques petites choses qui méritent d'être dites. Concernant l'interfaçage Unicode/clavier, WorldScript me semble un bon modèle de fonctionnement, comme je l'évoque dans la partie b. Toutefois, il date d'avant Unicode, donc du 8 bits. Dans le cadre d'Unicode, la sélection d'une langue devrait pointer non sur une fonte correspondante, mais vers le sous-ensemble approprié d'Unicode. Ce qui nous amène au gros morceau, l'interfaçage caractère/glyphe. Il me semble, si j'ai bien tout compris, qu'on est train d'essayer, *via* les *codepages*, de morceler Unicode en fragments 8 bits, pour y faire coller les fontes actuelles. C'est débile. Certes, il y a bien des sous-ensembles logiques dans Unicode, mais, sauf coup de chance, très peu d'entre eux rentreront dans 8 bits. Celui qui nous concerne le plus, le latin (ou roman) comprend au moins 800 signes (probablement beaucoup plus). La démarche devrait être plutôt de rassembler des fontes pour correspondre à ces sous-ensembles. Ce rôle pourrait, me semble-t-il, être dévolu au gestionnaire de fontes. On pourrait ainsi rassembler des fontes au standard Mac-latin, Adobe expert, Mac-CE, Adobe OSF, etc., pour constituer une *métafonte* qui contiendrait enfin à peu près ce dont on a besoin. Objet qui serait manipulé d'un seul tenant dans le logiciel de mise en pages. À nous alors les substitutions automatiques (Unicode sait composer/décomposer un digramme ou un trigramme). Pour combler les manques d'Unicode (au hasard les petites caps), on pourrait aussi affecter une série de glyphes à un caractère Unicode, afin de gérer les glyphes alternatifs (fonte alternate). Avec une convention bien gérée, ça devrait permettre de sélectionner un texte, de choisir « petites caps » afin qu'il aille chercher le signe affecté à la position *x* du caractère Unicode, alors que pour avoir les *alternate*, on irait chercher la position *y* (s'il n'y a rien, on laisse le signe standard). Idem pour les différentes casses de chiffres…

Combien de glyphes maxi pour un même caractère ? Ben, je verrais bien 256, pour pouvoir utiliser des fontes genre fontes d'esperluettes (ou d'Euros) en restant Unicodement correct… Accessoirement, si un système de ce genre se mettait en place comme standard, ça pourrait inciter les concepteurs d'Unicode à libérer un peu de place en virant les 12 astérisques, les 25 versions de cases à cocher ou de flèches qui n'ont rien à y faire… Inversement, un même glyphe doit pouvoir être affecté à plusieurs caractères, ça éviterait du travail inutile aux concepteurs de fontes, merci pour eux. Bien entendu, une partie du travail devrait pouvoir être fait automatiquement, au moins pour les codages standards, mais on doit pouvoir bidouiller ces propres tables de correspondances (comme pour les exemples cités précédemment).

Ceci, combiné à la possibilité offerte par OpenType de conjuguer des glyphes pour en créer de nouveaux, devrait permettre de résoudre pas mal de nos problèmes actuels (fontes CE inexistantes, etc.).

P.-S. : Et ce serait une bonne idée d'embaucher Günther Blashek, ce bienfaiteur de l'humanité, pour qu'il nous concocte le PopChar Unicode/WorldScript du prochain millénaire…

I – TYPOGRAPHIE

I.1.3 Support multilingue

b) WorldScript et équivalents

o.randier — Jusqu'ici, le support multilingue des logiciels de P.A.O. tient de la pure escroquerie. En effet, depuis MacOS 7.1 (ça fait quand même un moment), les ressources et les API permettant la composition multilingue sont intégrées au système, et donc disponibles pour les développeurs. Or Adobe et Quark ont refusé de les intégrer à leurs produits pour vendre leurs propres solutions à un prix prohibitif. Résultat, WorldScript, une des technologies les plus prometteuses d'Apple, périclite, alors que rien n'existe pour le moment pour le remplacer efficacement. Quand on compare à ce qui s'est passé dans d'autres domaines, avec QuickTime ou QuickDraw 3D, qui sont devenus des standards multiplateformes grâce au soutien des éditeurs (et pour le plus grand bénéfice des utilisateurs), il y a de quoi se bouffer les ongles de rage.

Je peux admettre qu'on fasse payer un petit sus (petit, j'ai dit, pas 2 000 balles par langue !), sous formes de modules optionnels de langues, pour le développement des vérificateurs orthographiques, des algorithmes et dictionnaires de césure spécifiques à chaque langue, mais il est, par contre, inacceptable de ne pouvoir disposer, dans des logiciels de ce prix, de fonctionnalités existants dans… SimpleText ! De même, je peux admettre des XTensions spéciales pour gérer les langues idéographiques sur 2 octets (CJK), mais pas de devoir désactiver mon système d'écriture cyrillique pour pouvoir composer du Russe sur XPress (non Passport), à l'aide de ressources maison...

Donc, on doit pouvoir, au minimum, utiliser les ressources du système pour composer au moins dans toutes les langues latines, grecques et cyrilliques (tout ce qui se compose bêtement de gauche à droite). Pour les systèmes d'écriture un peu plus complexes (ligatures contextuelles de l'arabe et des langues indiennes, langues idéographiques...), on peut admettre que certains problèmes de développement amènent un léger surcoût (encore que). Des modules additionnels optionnels pourront apporter dictionnaires et algorithmes évolués nécessaires dans un cadre de production, mais, en aucun cas, le logiciel *lite* ne devrait interdire de recourir aux ressources du système pour composer trois lignes de japonais.

Donc :

— Support intégral de WorldScript (ou équivalent),
— Au minimum, dictionnaires et algorithmes et tout ce qui va bien pour toutes les langues latines, le grec et le cyrillique,
— Éventuellement, modules additionnels optionnels (qui peuvent être de tierce partie) pour les langues complexes.

Bon, je ne suis pas un fanatique de WorldScript, je sais qu'il existe un système équivalent sous Fenêtres™ (mais nettement moins bien, à ce que j'en sais) et qu'il est prévu quelque chose dans OpenType (Uniscribe), je serais d'ailleurs très intéressé à échanger des informations sur ces systèmes, si certains en ont l'expérience ici.

M. Bujardet — Windows 95/98 a en effet un système équivalent à WorldScript qui fonctionne plutôt bien. Encore qu'il ne soit pas possible, comme sur Mac, de créer des claviers. Probablement parce que Windows supporte plus que les polices 8 bits habituelles.

Le support multilanguage comporte un nombre impressionnant de langues avec les systèmes d'écritures romain, cyrillique et grec. Avec les claviers idoines. Les polices natives du système sont multilingues, Unicode 16 bits (pratiquement, avec quelques limitations). Lorsqu'elles sont utilisées par les applications, elles présentent les scripts appropriés : Ouest, Baltique, Est-européen, Cyrillique, Grec en 8 bits, avec support des imprimantes et de la plupart des applications anciennes. On peut aussi utiliser les polices aux standards code-page habituels, CP-1250, 1251, etc. qui sont alors reconnues par le système de la même manière. Et qui fonctionnent avec toutes les applications, y compris les plus anciennes. C'est là qu'est le problème : des applications plus modernes fonctionneraient probablement parfaitement avec des polices Unicode, mais les systèmes doivent rester compatibles avec les applications existantes.

Tout cela est fort bien fait, et j'avoue que vu du côté du développeur de polices de caractères, ça semble fonctionner impeccablement. Les claviers sont très intelligemment intégrés aux applications. Par exemple, dans un même document où l'on aurait entré du français, du russe et du grec dans différents paragraphes, le clavier correspondant est automatiquement activé lorsqu'on parcourt le texte avec le curseur.

J'imagine qu'Apple devrait proposer quelque chose du même genre dans le cadre de son support Unicode. Cela étant, cela risque de changer quelque peu la donne du côté du support des claviers. Si un système de polices Unicode est utilisé, avec des scripts qui présentent aux applications et aux imprimantes des pages restant compatibles avec les encodages 8 bits actuels, le clavier devra devenir plus intelligent. Par exemple, le clavier russe devra comporter l'information correspondant au script nécessaire (Cyrillique), et chaque touche devra générer non pas simplement un code ASCII, mais une requête pour un glyphe Unicode, que le système ira chercher selon le cas dans la page Unicode appropriée, ou dans le jeu de caractères en encodage Apple 8 bits actuel. Du moins, si Apple envisage de conserver le support des polices WorldScript actuelles, ce qui semble vraisemblable.

D'une manière générale, ce qu'ont fait les gens de Windows semble plus ou moins inévitable, si Apple veut supporter des polices 16 bits sans pour autant jeter aux orties tout ce qui a été fait en polices 8 bits. Et cela comprend pas mal d'applications.

Il est intéressant en tout cas de voir que tes préoccupations recoupent exactement ce que semblent avoir rencontré les gens de Windows. Unicode seul ne résout pas tout, loin s'en faut. Les claviers non plus. Et le système d'écriture doit de plus en plus incorporer des règles d'intelligence…

o.randier — L'important, dans tout ça, c'est qu'on ait une interface correcte entre Unicode et le clavier. De ce côté-là, WorldScript me semble insurpassable. Ce que je ne veux pas, en tout cas, c'est devoir saisir des codes abscons à la Windows, genre alt-197 ou pire #A6B7 (les codes hexa d'Unicode, même s'il pourrait s'avérer utile d'y recourir occasionnellement), pour obtenir un bête alpha. Je ne veux pas non plus qu'on me balance tout ce qui sort de l'ordinaire dans une table des symboles, parce qu'un tableau de 65 000 signes, ça ne rentre pas dans mon 17″ ! Ce que je veux, c'est pouvoir continuer à faire ce que je fais à l'heure actuelle, c'est-à-dire sélectionner la langue dans un menu, ce qui remappe mon clavier, change la fonte de façon appropriée, modifie les critères de tri, les séparateurs décimaux et de milliers, etc. Et les ressources *ad hoc* doivent rester facilement éditables (elles pourraient même l'être plus), pour que je puisse bidouiller mon clavier transliterré si les ressources adaptées à mon Azerty français ne sont pas fournies. Bref, WorldScript marche, et bien, je ne vois pas de raison d'en changer. Si en plus ça reste compatible avec PopChar, c'est top !

Par contre, une chose qui serait importante dans ce contexte, ce serait de pouvoir baliser les langues dans le texte, afin de pouvoir gérer les appels aux dictionnaires, les C&J, etc., en fonction d'ycelles. Et ces balises pourraient être affectées de feuilles de styles (par exemple pour automatiser la mise en italique des mots étrangers dans un texte).

II – TRAITEMENT DE TEXTES

II.1 – Fonctions de base

b) Indexation/table des matières

E. CURIS — Avant d'avoir tout oublié des discussions entre Alain Hurtig et moi, voici ce que nous avions conclu à propos de l'indexation.

- On doit pouvoir indexer n'importe quoi (texte, y compris ses enrichissements ; images ; …)

> J. ANDRÉ — Oui, mais par texte il faut préciser que l'on peut indexer un mot, une section, un paragraphe, un chapitre entier etc. Ce qui veut dire un marqueur de début et un marqueur de fin d'index !

- Possibilité d'avoir des index à plusieurs niveaux
- Possibilité d'avoir plusieurs index
- Indexation d'un même mot sous plusieurs entrées
- Indexation d'un mot sous une entrée radicalement différente de lui-même
- Tri alphabétique correct pour le français (avec la sympathique question des chiffres, tant qu'à faire…)
- L'index produit doit être un texte exactement comme les autres, donc que l'on puisse mettre où l'on veut et que l'on puisse modifier à sa guise.

> J. ANDRÉ — On doit donc pouvoir éventuellement indexer l'index…
>
> > E. CURIS — Ce qui me fait penser à quelque chose : on doit pouvoir faire « automatiquement » des renvois dans l'index… De façon, par exemple, à ce que l'indexation automatique de « Mme de La Fayette » soit comptée aussi pour « La Fayette (Mme) » et place un renvoi à « Mme de… » *s'il en trouve* (exemple bidon, c'est juste pour l'idée).

Ce qui semble le mieux, en fait, c'est que l'index soit fait à partir de codes dans le texte, qui contiennent l'entrée (éventuellement la sous-entrée) de l'index qu'il faut enregistrer à cet endroit du texte. Les codes doivent alors pouvoir être entrés à la main ou dans le cadre d'un chercher/remplacer (indexation « automatique » d'une liste de mots).

Il me semble que c'est tout…

Ce qui existe

- Calamus. En théorie : code de contrôle sur ce principe, tout bon si ce n'est qu'on est limité à un index par texte et un seul niveau. En pratique, il n'y a pas moyen d'obtenir les numéros de page (!). Il y a donc un module annexe qui fait de la recherche/indexation à partir d'une liste, mais on a obligatoirement mot de la liste = entrée :-(Bref, à revoir. En plus, les accents sont mal triés :-(

- TeX. Tout ce qu'il faut, si je ne m'abuse — si ce n'est que mes essais montrent que les accents sont mal triés (il paraît que c'est corrigé depuis) et qu'il paraît que la modif'. de l'index est une horreur.
- Xpress. Rien de base, mais des Xtensions — si j'ai tout bien suivi.
- Word. Dans sa dernière version, il semble y avoir pas mal de choses pour faire ça qui paraissent puissantes. Mais accents en majuscules mal triés.

Voilà pour ceux que je connais un peu.

c) Tableautage

O. RANDIER — Le tableautage est un des gros soucis de la mise en pages. Les solutions actuelles ne sont pas satisfaisantes, parce qu'elles appliquent soit la logique tableur (cellules), soit la logique texte (tabulations) de façon rigide. Or il faudrait une logique tableur pour le flux de texte, mais conserver le fonctionnement global de mise en pages pour le reste. Ce qui serait bien, c'est un outil de chaînage « tableau » qui transforme un groupe de bloc en cellules, chaque bloc restant gérable indépendamment et par colonnes/lignes comme des blocs ordinaires. On doit pouvoir, par contre, sélectionner du texte par colonnes/lignes. L'ensemble doit pouvoir être « ancré » dans un flux de texte.

Ceci n'empêchant pas la gestion de tableaux à l'ancienne mode d'XPress, qui permet de gérer plus facilement certaines choses comme les points de conduite.

II – TRAITEMENT DE TEXTES

II.1 – Fonctions de base

d) Gestion des notes

E. CURIS — Voici une petite base de travail pour ce qui est de la gestion des notes de bas de page dans le futur logiciel idéal à venir. C'est le fruit d'une collaboration avec Alain Hurtig.

Le logiciel doit pouvoir numéroter les notes automatiquement — si possible en temps réel — de la façon que l'on veut (lettres/chiffres/symboles ; en commençant à 1 à chaque page/texte/chapitre/où l'on veut). Il serait sympathique aussi d'avoir une possibilité d'appel de note lié à un numéro de ligne — donc certaines valeurs pouvant disparaître. Ça implique accessoirement un compteur de ligne, qui pourrait avoir une utilité certaine pour certaines mises en pages.

T. BOUCHE — Euh, ça consiste en quoi, une numérotation avec des symboles, m'sieur ?

E. CURIS — Numérotation * puis ** puis *** puis… J'ai déjà vu ça quelque part — mais ça suppose bien sûr qu'il y ait très peu de notes par page et que l'on recommence à un sur chaque page…

E. CURIS — Le texte de la note doit pouvoir être modifié, mis en page, traité, … tout ce qu'on veut exactement comme du texte normal. Si possible, existence quand même de paramètres par défaut (comme la position des filets).

A. HURTIG — Ce qui peut sembler évident et facile à mettre en œuvre pour des notes en fin de chapitre ou de volume — on a alors affaire à un simple flot de texte, seuls étant spécifiques les numéros de notes qui sont gérés par le logiciel et automatiquement remis à jour en cas d'ajout ou de retrait d'une note — pose un redoutable problème pour les notes disposées sur la même page que leur appel.

Dans le premier cas, l'intervention du typo se borne, finalement, à la gestion des veuves et des orphelines, aux fausses coupes, aux lézardes, etc. : l'agencement du texte n'est pas fondamentalement modifié.

Dans le second, le domaine d'intervention est beaucoup plus vaste. Rien ne garantit, en premier lieu, que les notes viennent en bas de page : on peut avoir un système de gloses marginales. Il faut donc que l'opérateur intervienne pour positionner ses notes correctement, ce que le programme ne saurait faire qu'en première approximation (ça peut demander un assez long travail, surtout quand les notes sont en habillage, une

partie « enfoncée » dans la colonne de texte et l'autre dans le grand fond : j'en parle en connaissance de cause).

Mais même dans le cas beaucoup plus fréquent des notes en bas de pages, il est indispensable que l'opérateur-metteur en pages garde une maîtrise totale sur le rendu de sa page. Si le programme doit venir placer correctement les notes sous leur appel (ce que fait TeX, ce que fait Word, ce que ne fait pas XPress !), il ne doit pouvoir le faire qu'en première instance. C'est au metteur en pages qu'il revient de savoir si la note ne va pas venir « déborder » sur la page suivante (quand le logiciel juge, lui, que « ça rentre »… mais avec un chausse-pied et en forçant !). C'est lui qui sait si la note toute entière doit se trouver sur la page suivante (pour conserver le gris d'une page par exemple, ou pour toute autre raison souvent liée au sens). C'est lui qui juge s'il convient d'augmenter ou de diminuer l'espace entre la colonne de texte et la note sur une page donnée (quitte à bouleverser temporairement l'équilibre des pages suivantes), simplement pour éviter une veuve malencontreuse ou une césure incorrecte, etc.

Bref : une note doit être considérée comme une portion de texte « normal », au style près, et à la gestion du numéro de note près. Cela pose bien entendu le problème de l'automatisation de la gestion des appels de notes : que se passe-t-il lorsque, sur un travail achevé, l'auteur a la lubie de rajouter ou de supprimer une note ? Le programme *doit* gérer ce cas d'espèces, hélas fréquent ! Il est probable qu'alors, la mise en pages et le placement des notes doivent être refaits à la main. Ce serait un moindre mal !

E. CURIS — Lors de la répartition du texte des notes, si une note doit être répartie sur plusieurs pages, possibilité d'insertion automatique de renvois [(suite de la note … de la page …)] — le texte de ce renvoi étant bien sûr entièrement définissable.

Possibilité de définir et mélanger plusieurs familles de notes, avec donc numérotations distinctes et tout et tout. Les notes doivent pouvoir être placées n'importe où quelle que soit leur famille (texte, notes, légendes de dessins, …) ou au contraire limitées à un chaînage de texte.

Éventuellement, possibilité d'insérer des renvois entre notes précisant à quelle page se rapportent les notes qui suivent — pour les regroupements en fin de volume.

Ce qui existe

Ce qui suit est très succinct mais doit donner une idée de ce que l'on peut faire. Par ordre alphabétique de logiciel.

- Calamus : O.K. pour la mise en page [c'est du texte exactement comme les autres placé dans des cadres/blocs définis par l'utilisateur]. À mon avis, O.K. pour le principe de fonctionnement [code de contrôle dans le texte qui contient le texte de la note], mais trop limité en pratique : une seule famille de notes par chaînage ; numérotation obligatoirement continue pour le chaînage et liée au chaînage ; impossible d'utiliser des notes appelées par un symbole ; aucune insertion de renvois. Peut-être bien aussi une

limitation à 65 536 caractères par note, mais je n'ai encore jamais atteint cette longueur alors je ne sais pas trop…

- TeX : sauf erreur, tout cela doit pouvoir être fait à condition de se plonger dans le langage…
- Word : O.K. pour la numérotation et le fonctionnement mais trop peu de liberté pour la mise en page.
- Xpress : apparemment rien à la base, mais il y aurait des X-tensions ?

II – TRAITEMENT DE TEXTES

II.2 – Utilitaires textuels

a) Vérificateurs orthographiques/grammaticaux

O.RANDIER — La vérification orthographique est un vaste débat, dans lequel nous ne rentrerons pas. La qualité généralement médiocres des vérificateurs des différents logiciels de P.A.O. a suscité de nombreuses solutions de tierce partie, dont il serait utile de s'inspirer.

Au sujet des langues, par contre, il serait utile, dans le cadre évoqué à propos du support multilingue, de pouvoir désigner le vérificateur (éventuellement de tierce partie) à employer en fonction de la langue (il ne sert à rien de contrôler un texte en anglais avec le vérificateur français !) et de pouvoir lancer une vérification globale des diverses langues.

b) Moteur de recherche

O.RANDIER — La recherche ne doit pas se limiter, comme c'est actuellement le cas dans XPress, aux caractères courants. On doit pouvoir rechercher/remplacer n'importe quoi, un texte, un attribut de style, une feuille de style, une balise, la couleur d'un bloc, le x^{e} mot, lettre, etc. d'un style donné, etc. (liste non limitative).

c) Éditeur de texte intégré

O.RANDIER — On a déjà évoqué, à propos d'autres sujets, la nécessité d'un éditeur de textes intégré, afin de permettre de faire des corrections dans un contexte d'affichage simplifié et surtout pour pouvoir « mettre les mains dans le cambouis » pour des contrôles pointus. Bien entendu, il faudrait pouvoir basculer facilement entre l'affichage avec et sans balises (bien distinctes, les balises). Et l'éditeur doit pouvoir s'utiliser seul, le secrétaire de rédaction n'a pas besoin de lancer un logiciel lourd pour saisir ses textes.

Surtout, TRÈS IMPORTANT, il faudrait pouvoir enregistrer le texte en externe au fichier et pouvoir y abonner plusieurs fichiers indépendamment de l'enrichissement. J'explique : je fais de la pub dans le médical (beurk). Très souvent, je fais des séries d'annonces pour des plans (sic) média. Ces annonces comportent forcément les mentions légales du produit (un délire hallucinant du ministère de la Santé), enrichies différemment en fonction du contexte (taille de l'annonce, couleurs, texte plus ou moins maltraité *pour que ça rentre*. Invariablement, les fêlés du ciboulot des laboratoires qui se tripotent sur leur texte (que personne ne lira, vu que c'est en corps 6 et que tout est déjà dans le Vidal) me donnent des corrections APRÈS que j'ai enrichi 15 annonces. Corrections que je suis obligé de faire 15 fois, donc. Si je pouvais abonner mes 15 annonces à un unique texte sans foutre en l'air mon enrichissement, mon travail ressemblerait un peu moins à un enfer. D'autant que je n'aurais pas besoin d'ouvrir 25 fichiers pour retrouver celui où se trouve la dernière version de ces @#&$* de M.L. de m…

III – FONCTIONS DE MONTAGE

III.2 – Calques

O.RANDIER — À ce sujet, que dire sinon : « On veut des calques ! » On pourrait juste ajouter, au niveau de la conception, que les maquettes pourraient être une extension du concept de calques. Comme sur Illustrator, on devrait, bien sûr, pouvoir masquer des calques, les verrouiller, etc. Rien de très nouveau, donc.

III.3 – Montage des pages

b) pages multiformat

O.RANDIER — Il serait extrêmement pratique de pouvoir assembler des pages de formats différents, pour réaliser des couvertures avec des dos carrés, des chemises avec rabats, etc. Bien sûr, dans ce cas, on aimerait que traits de coupe ou de pliure (au choix) se placent au bon endroit. Les pages assemblées de cette façon doivent rester imprimables séparément. Et on doit pouvoir assembler les pages aussi bien verticalement qu'horizontalement (rabats de pied d'une chemise). Afin d'éviter de se luxer les cervicales, il faudrait pouvoir composer les pages à l'horizontale et les tourner lors de l'assemblage. En fait, l'assemblage doit restituer dynamiquement les pages composées individuellement (je vois ça juste comme une bascule entre deux modes d'affichage).

c) pages non rectangulaires

Dans le contexte évoqué précédemment, il serait utile de pouvoir faire des pages non rectangulaires. Par exemple, les rabats d'une chemise et autres pattes de collages sont souvent trapézoïdaux. Et j'ai déjà eu à faire des mises en pages sur des pyramides (ah les enfoirés !). Ça m'aurait beaucoup aidé de pouvoir éviter d'essayer de me rappeler la trigonométrie… L'idéal serait de pouvoir désigner un tracé vectoriel comme format de page (avec placement des hirondelles et forme de découpe automatique).

d) pages pré-imposées

On devrait pouvoir faire des impositions simples directement, sans passer par des solutions professionnelles couteuses. Par exemple, une planche constituée de plusieurs exemplaires d'une même page, ou un chemin de fer à taille réelle, pour économiser du papier ou du film. Il faudrait aussi pouvoir enregistrer plusieurs impositions d'un même fichier et pouvoir basculer de l'une à l'autre à l'écran ou à l'impression (par exemple, entre pages à la suite et vraies doubles).

IV – FONCTIONS PRÉPRESSE

IV.1 – Trapping (tiens, on n'a pas de traduction correcte de ce terme)

a) Trapping Wysiwyg

O.RANDIER — Il serait très utile de pouvoir visualiser et modifier le trapping de n'importe quel objet. Un mode de prévisualisation et de travail dédié (où on ne verrait que les trappings, avec un code couleur pour les trapping positif, négatif, etc.) serait sans doute le plus simple. En tout cas, le trapping doit rester dynamique, comme il l'est sur XPress, et non « compilé » après coup, comme sur Illustrator, ce qui se révèle rapidement ingérable.

b) Logique de trapping

O.RANDIER — Si le système de trapping dynamique d'Xpress est insurpassable dans le principe, dans son utilisation pratique, il présente deux lacunes importantes :

— L'algorithme ne prend en compte que les blocs, et dans leur totalité, moyennant quoi d'une part, il calculera le débord d'un texte coloré, dont le bloc déborde sur une image, même si le texte ne déborde aucunement, d'autre part, il ne calculera pas le débord d'un texte sur un autre ou, pire, d'un bloc sur un cadre ! L'algorithme doit prendre en compte tous les objets séparément.

— Si du texte coloré se trouve sur une image, il ne peut calculer qu'un trapping positif ou nul, alors qu'il en faudrait généralement un négatif (maigri de la défonce dans l'image). Il paraît que c'est une limitation de Postscript. Il serait temps de la dépasser… D'une façon générale, il faudrait pouvoir tenir compte des imports dans le calcul de trapping (particulièrement des imports vectoriels).

IV.3 – Séparation

O.RANDIER — Pas grand-chose à dire sur la séparation, qui me paraît globalement bien fonctionner dans XPress. Une chose, quand même : il faudrait pouvoir supprimer *aussi* les encres quadri, quand on fait des documents mono-, bi- ou trichrome, ça éviterait bien des erreurs.

IV.4 – Gestion des images

a) Gestion 32 bits

O.RANDIER — S'il y a quelque chose qui ferait gagner du temps, de la place et de l'argent, ce serait une vraie gestion 32 bits des images, c'est-à-dire la possibilité d'enregistrer un masque de transparence en 8 bits dans le format d'image. Encore une limitation de Postscript, qu'on avait espéré résolue avec Postscript 3 (sigh)… En l'absence de gestion par Postscript, la gestion du format Photoshop natif pourrait faire l'affaire, à la rigueur.

On aimerait pouvoir choisir d'imprimer la prévisualisation à la place de l'image, image par image, et avec le tracé de détourage (pour les épreuves de lecture).

Une bonne gestion des liens avec les bases de données images est indispensable.

VI – ARCHITECTURE LOGICIELLE

VI.6 – Interface

a) Gestion des unités de mesure

P. CAZAUX — Tiens, un truc que j'avais oublié […] et qui me gonfle en ce moment : avoir des unités de mesures variables dans les différents dialogues, et qu'on puisse en changer dans un dialogue sans que ça change partout. Je m'explique, parce que je vois bien que je ne suis pas clair : je veux avoir dans le dialogue Style – Format d'Xpress des points dans les espaces inter-paragraphes (directement, pas en tapant des points et en ayant une conversion en mm) sans être obligé d'aller changer le choix des unités de mesure dans les prefs, parce que je veux que les règles restent en mm. Ca va, là ? Je verrais bien un petit menu déroulant proposant les différentes unités en face de chaque case dans laquelle on tape une mesure, avec la possibilité de fixer l'unité de chaque case par défaut.

> A. HURTIG — Pour les non-XPressiens, je tente une explication plus claire (voyons si c'est *vraiment* plus clair !) : dans les boîtes de dialogue de XPress, on peut rentrer des valeurs avec n'importe quelle unité (des points Postscript, des picas, des cm, des mm, il y en a une palanquée).
>
> Disons qu'on peut demander un inter-paragraphe de 6 pt, ça ne gène pas le logiciel. Mais quand on rouvre la boîte de dialogue, XPress a converti la valeur en question en mm (ou en cm, ça dépend de la valeur indiquée par défaut dans les préférences) : c'est extrêment gênant, parce qu'on ne se souvient pas forcément de ce qu'on avait décidé de faire quand on a déterminé ses feuilles de style (et autres), et que faire la conversion « à l'envers » est évidemment fastidieux.
>
> Cette conversion automatique est valable partout, sauf pour l'interlignage et la force de corps, qui sont toujours en pt. Et montre à quel point ce logiciel n'est pas fait pour les gens de labeur, ou tout simplement pour tous ceux qui ont des maquettes rigoureuses à produire : c'est l'enfant adultérin de publicitaires et d'informaticiens.
>
> Moi je verrais bien qu'il garde les unités comme on les a rentrées (et qu'il convertisse à la demande, uniquement : on clique sur un bouton et hop ! il convertit… Comment le logiciel se débrouille en interne pour répondre à la commande n'est pas mon problème.)
>
> > O. RANDIER — Sans oublier qu'on aimerait pouvoir utiliser des unités relatives (fixer la valeur d'alinéa en cadratins, par exemple), ou fixer des unités personnalisées (pour utiliser le point millimétrique, par exemple). Par contre, la possibilité de faire des opérations en mélangeant les unités (et, d'une façon générale, de pouvoir faire des opérations dans tous les champs de valeurs), qu'on trouve dans XPress, est à conserver et à généraliser.