UNIVERSITÉ CADI AYYAD
FACULTÉ DES SCIENCES
SEMLALIA - MARRAKECH
UFR DE CALCUL FORMEL & INFORMATIQUE
DÉPARTEMENT DE MATHÉMATIQUES

N° d'ordre

MÉMOIRE
présenté à la Faculté, pour obtenir
le Diplôme des Études Supérieures Approfondies
en Informatique

REF 06/01

TITRE

ÉTUDE ET RÉALISATION D'UNE FONTE POUR L'ÉCRITURE ARABE

Par
Mohamed ELYAAKOUBI

Soutenu le 22 novembre 2003 devant la commission d'examen :

Président :

Abdelilah KANDI-RODY	P.E.S Université Cadi Ayyad de Marrakech

Examinateurs :

Salah-Eddine LABHALLA	P.E.S Université Cadi Ayyad de Marrakech
Azzeddine LAZREK	P.E.S Université Cadi Ayyad de Marrakech
Khalid SAMI	P.E.S Université Cadi Ayyad de Marrakech

Remerciement

Je tiens à exprimer ma gratitude au professeur Azzeddine LAZREK, qui a dirigé ce travail avec beaucoup de patience et de disponibilité. Ses explications et ses idées ont été indispensables pour l'avancement de cette étude.

J'adresse mes remerciements sincères et spéciaux à Monsieur Abdelilah KANDR-RODY, le chef du Laboratoire de Calcul Formel et Informatique.

J'adresse mes remerciements chaleureux à Monsieur Mostafa BANOUNI professeur au Département de Physique à la Faculté des Sciences d'Agadir, pour son accueil chaleureux et ses orientations.

Je tiens à remercier Monsieur Salah-eddine LABHALA pour l'intérêt qu'il a porté à ce travail en acceptant de faire partie du jury.

Je remercie également tout les membres de l'équipe de composition du e-document scientifique multilingue, et plus particulièrement Monsieur Khalid SAMI pour la lecture qui a fait à ce rapport, son aide, sa collaboration et ses orientations.

Merci également à tout le personnel du Laboratoire pour son efficacité et sa gentillesse, M. El Kahoui, B. Sadik, D. Bouziane et H. Izelgue.

Je suis particulièrement sensible à la très bonne ambiance du Laboratoire Calcul Formel & Informatique. Je remercie tous mes collègues du Laboratoire Said, Mohsine, Toufiq, Mustapha, Abdellah, Abderrahim et Hassan.

Enfin, j'exprime toute ma sympathie à tous les membres de ma famille ainsi que mon ami Abderrahim pour leur soutien moral.

Table des matières

Table des figures

Chapitre 1

Introduction

Sujet vaste et complexe, celui des rapports entre ordinateur et typographie. Historiquement l'initiation dans ce domaine revient à Macintosh d'`Apple`, prévu à l'origine pour fonctionner en mode graphique, qui est une condition indispensable à la qualité typographique.

Dans ce mémoire, nous proposons une étude et une réalisation d'une fonte pour l'écriture arabe. Nous verrons d'abord dans les deux premiers chapitres une introduction à la typographie ainsi qu'aux technologies inventées pour gérer les polices de caractères. Le troisième et quatrième chapitre expliquent plus précisément de quoi est faite une police de type `TrueType` et `OpenType` et explorent les coulisses de leurs fichiers, des secrets qui étaient jalousement gardés par les spécialistes. Dans cette étude, nous nous sommes basés essentiellement sur deux manuels de références proposés en ligne par `Apple` [3] et `Adobe` [5].

Créer une police de caractères pour l'arabe, qui soit cohérente et belle et qui respecte un ensemble minimum de règles calligraphiques en vigueur, est une tâche à la fois technique et artistique complexe. Ce qui rend la confection plus difficile est l'exigence d'une analyse contextuelle : le processus par lequel les lettres arabes acquièrent des formes différentes selon leurs positions dans le mot, et selon les lettres environnantes, ainsi que la présence des voyelles et des signes diacritiques qui se placent au-dessus ou au-dessous des lettres et qui ne doivent pas affecter la contextualité. L'écriture latine possède depuis l'époque romaine deux styles fondamentaux : l'écriture manuscrite cursive et les caractères individuels utilisés par les graveurs pour les inscriptions lapidaires qui ont été utilisés par la suite pour confectionner les caractères mobiles de la typographie. Par contre, l'écriture arabe est une. L'imprimerie ainsi que l'outil informatique, jusqu'à présent, se contentent de reproduire fidèlement la structure manuscrite et cursive de l'écriture, et qui par conséquent engendre différents difficultés lors de son introduction aux nouvelles technologies de la typographie numérique. Cet aspect des choses que nous allons examiner dans le dernier chapitre de ce mémoire où nous verrons deux programmes de typographie appliquée sur ordinateur `Font Creator` et `Visual OpenType Layout Tool`. Grâce à ces outils on a pu confectionner notre police pour l'arabe. Notre

confection se restreindra a une fonte de style Naskh نسخ, le style adopté depuis les premières tentatives d'informatisation et de normalisation de la calligraphie arabe. Tous les styles d'écriture arabes ne sont pas facilement transposable au caractères d'imprimerie, le style Naskh, Koufi كوفي, Ruqaa رقعة, et plus difficilement Farsi فارسي ont aujourd'hui des équivalents en lettres d'imprimerie. Il en va autrement pour les autres styles comme le Diwani ديواني ou Thuluth ثلث par exemple qui ne se prête pas à une telle simplicité dans la mesure où les lettres n'ont pas de ligne de base commune, en particulier.

Avec le développement des techniques de la typographie numérique, et l'apparition des formats de fontes vectorielles TrueType et OpenType, différentes sociétés ont développé leurs propres fontes arabes. Citons ainsi la fonte *Traditionnel Arabic* de Microsoft, où on trouve une place pour des ligatures esthétiques qu'on ne trouve pas dans la fonte *Times New Roman* de la même société. Dernièrement des formes de fonte ont apparu pour la conception de textes coraniques avec des différents allographes (variantes du même glyphe) [19].

Des nouveaux concepts d'entrée et de sortie sont introduits sous Unicode. L'arabe classique et coranique requiert plus de caractères que le clavier de saisie arabe ne peut fournir. Les sociétés Basic et Docotype ont développé en commun un EME (Editeur de Méthode d'Entrée) [12] pour saisir de l'arabe sophistiqué, et transcrire l'imprimerie traditionnelle avec les allongements de la Kashida d'une manière curviligne, et l'allongement des signes diacritiques suivant la langueur de la Kashida.

En ce qui concerne une fonte mathématique, à notre connaissance et jusqu'à présent les tentatives de réalisation d'un système de traitement du document mathématique arabe sont très limités. Le système RyDArab [13] de Azzeddine Lazrek est la première extension du système TEX et de l'extension ArabTEX ou Ω pour générer des expressions mathématiques arabes utilisant des symboles spécifiques et se déroulant de droit à gauche. À part la tantative de Fayes Alhargan [18], une fonte mathématique arabe TrueType ou OpenType utilisée dans un concept de WYSIWYG est quasi inexistante.

Chapitre 2

Introduction à la typographie

La typographie numérique est l'art de composition de textes qui comprend la conception, le style, l'apparence, l'impression, l'affichage et l'utilisation des polices de caractères, plus généralement, le traitement des documents textuels assisté par ordinateur.

2.1 Terminologie

Cette section propose un glossaire restreint de termes fréquemment employés en typographie.

- **Caractère :**
  On désigne par caractère une entité abstraite décrivant un signe typographique (ex. une lettre, un chiffre, une ponctuation, ...), indépendamment de la forme qu'il peut prendre.
  Ainsi le caractère "lettre latine majuscule A" décrit une lettre majuscule, objet linguistique qui fait partie d'un ensemble bien connu : l'alphabet latin.
- **Glyphe :**
  Le glyphe est la représentation visuelle du caractère, en d'autres termes, l'image qui s'inscrit à l'écran ou sur papier.
  À un même caractère, ils correspondent plusieurs glyphes qui représentent ses différentes formes, (ex. A, A, ...), on peut avoir des caractères qui ne correspondent à aucun glyphe, c'est le cas des caractères de contrôles qui correspondent à des touches de contrôles sur le clavier (ex. le retour chariot, la tabulation, ...).
  De plus, un même glyphe peut correspondre à plusieurs caractères successifs ; c'est le cas des ligatures (ex. œ, fi, ...).
- **Police :**
  Le terme police désigne un ensemble complet de caractères, d'une même famille, d'un même dessin, et d'une même graisse.
  Exemples : Times New Roman, Arial, Courier, ... sont parmi les polices les plus connues.

- **Fonte :**
  Le terme *fonte* issu de l'imprimerie traditionnelle souvent utilisé en informatique, provient du mot anglais *font*, qui désigne une police dans une certaine taille avec un attribut. Ainsi Times New Roman 10 gras est une fonte et Arial 10 italique est une autre.
  Ce terme est très fréquemment utilisé comme un synonyme de *police* et vice-versa.
- **Attribut :**
  À l'intérieur de la même famille, plusieurs effets peuvent être introduits pour mettre en évidence des parties d'un texte.
  En général, on trouve des variantes en gras, en italique ou une combinaison entre les deux.
  D'autres styles d'accompagnement peuvent être utilisés, comme la mise en petites capitales ou le soulignement...
  On distinguera des polices avec empattement, on dit aussi sérif, (ex. Times New Roman) et des polices dites sans empattement ou sans sérif (ex. Arial).

2.2 Métriqures des fontes

Parler des caractères, c'est d'abord parler de leur anatomie et expliquer les différentes propriétés métriques pour en caractériser la forme. Un caractère à l'origine est un parallélépipède de métal destiné à l'impression, sur lequel est gravé le dessin proprement dit de la lettre.

De nos jours, les caractères au plomb ne sont plus guère utilisés que pour les éditions de l'art et de travaux spécifiques ; mais, il est difficile, voire impossible, de parler de caractères numériques, ni surtout des propriétés métriques sans faire allusion au plomb. Le figure 2.1 montre en clair les différentes notions physiques liées au dimensions du caractère, les cadres noirs correspondent au bords du caractère et la ligne pointillée est une ligne fictive, appelée *ligne de base*, sur laquelle s'appuie la base des lettres. Nous donnerons par la suite quelques concepts de base.

- Œil :
  L'œil d'un caractère est sa partie imprimable ; le signe en relief qui reçoit l'encre.
- Bonding Box :
  Le Bonding Box est la plus petite boite qui contient l'œil d'un caractère.
- Talus :
  Le talus est l'espace existant en haut de la surface de l'œil. Entre la partie supérieure de l'œil et le bord supérieur du caractère, on parle de talus de tête. Entre la partie inférieure de l'œil et le bord inférieur du caractère, on parle de talus de pied.
- Approche :
  L'approche est le blanc gauche et droit nécessaire entre deux lettres pour qu'ils ne se touchent pas.

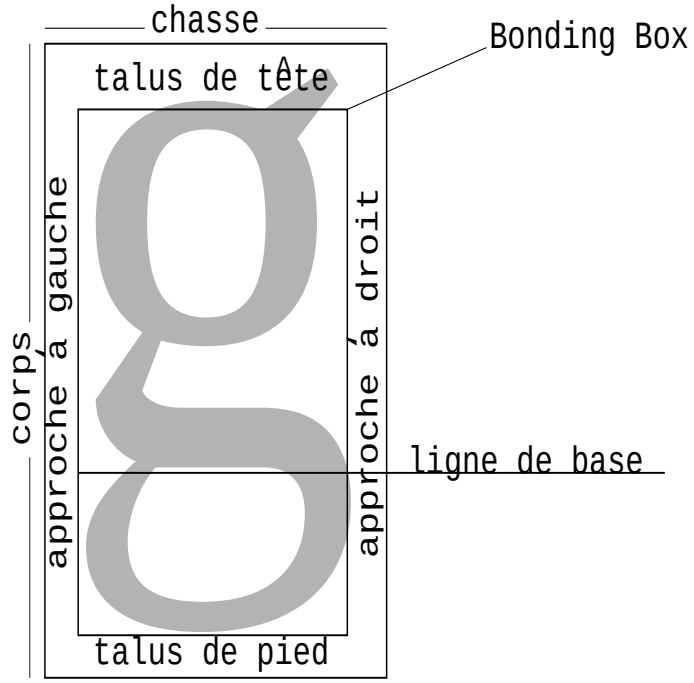

FIG. 2.1 – *Métriques des fontes en plomb*

- Corps :
  Le corps définit la taille verticale des caractères d'une même police exprimée en points, un point vaut 1/72 pouce, c'est-à-dire 0.04 *cm*, soit par exemple à peu prés 0.56 *cm* de hauteur dans le corps 14.
- Chasse :
  La chasse désigne l'encombrement total horizontal du signe typographique, c'est la largeur de l'œil plus les approches. À l'intérieur d'une même fonte, la chasse peut prendre des valeurs différentes : étroite, normale, large, ..., ainsi le "m" est plus large que le "i".
  Un caractère carré dont la chasse est égale au corps s'appelle un cadratin, c'est souvent le cas d'un "M" ; c'est pourquoi on emploie le terme *carré em*. Un caractère dont la chasse est la moitié du corps s'appelle un demi cadratin ou *en*, dans la terminologie anglaise, le quart de cadratin est souvent appelé *fine*.
  Le cadratin servait, par exemple, à la valeur de retrait usuel en début d'un paragraphe. Le demi cadratin correspond à la chasse des chiffres et permet alors d'aligner des colonnes des nombres. La fine est l'espace que l'on met, par exemple, devant le point virgule ou le point d'exclamation[1].
- Crénage :

[1]par contre en typographie française devant les deux points on met un espace normal.

Bien que les approches étaient calculées pour que les espaces entre lettres aient des valeurs convenables, certains cas de voisinage pouvaient produire un blanc très grand. Ainsi, par exemple, l'espace créé par la pente du "A" et du "V" dans le mot `AVIS` peut laisser croire qu'il y a deux mots. Pour éviter ceci, on utilisait souvent des caractères crénés.

- Ligature :
  Une ligature est une seule graphie résultant de la fusion de deux caractères, comme "æ" ou "Æ". On peut classifier les ligatures typographiques en trois types [9] :
  1. les ligatures linguistiques, ce sont celles obligatoires pour l'écriture d'une langue donnée, et obéissent à des règles grammaticales. L'exemple le plus proche est la ligature arabe LAM-ALEF ﻻ ;
  2. les ligatures esthétiques, ce sont celles qui ne sont pas indispensables et qu'on peut remplacer par leurs composantes non liées, telle que ﷺ ou ﷻ ;
  3. les ligatures contextuelles, ce sont les variantes de positions dans un processus de composition de texte. L'écriture latine actuelle n'utilise pas des ligatures contextuelles, par contre l'arabe qui est une écriture cursive, est l'exemple par excellence d'une écriture riche en ligatures contextuelles. Exemple :

ا+ل+خ+ط+ +ا+ل+ع+ر+ب+ي ⟵ الخط العربي

On retient en particulier que la chasse est l'espace horizontal qu'occupe un caractère. À l'intérieur de la chasse, les approches gauche et droite, correspondent à la partie blanche servant à séparer deux caractères, et les talus, de tête, et de pied, permettant aux lettres de ne pas se toucher d'une ligne à l'autre.

Regardons maintenant le figure 2.2, on remarque que les minuscules sont appelées bas de casse. Cette appellation provient du fait que déjà les anciens caractères en plomb étaient rangés dans un plateau comportant un grand nombre de cases, les minuscules étant placés en bas. On peut noter que le dessin de "a" descend légèrement de la ligne de base, tandis que celui de "h" dépasse un peu le haut des majuscules.

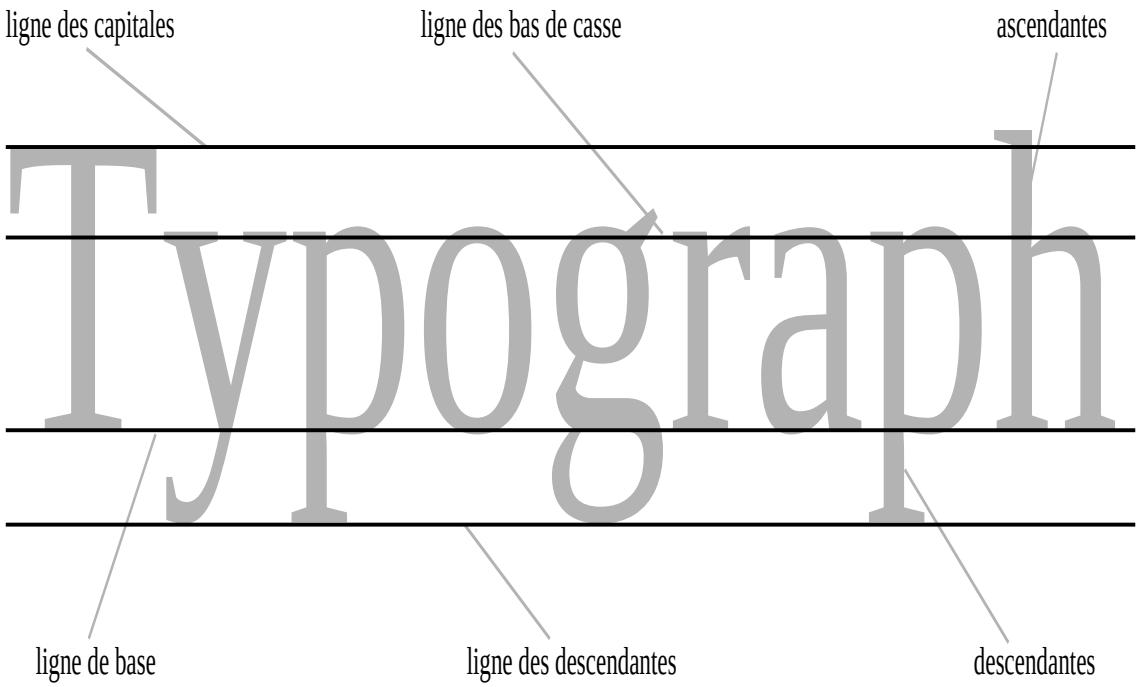

FIG. 2.2 – *Lignes de repère des caractères*

Chapitre 3

Technologie de fonte

En informatique, on distingue entre deux grands types de polices, les polices bitmap ou point par point et les polices vectorielles.

3.1 Police bitmap

Une police bitmap est un ensemble de caractères décrits par des matrices binaires, des tableaux de points (pixels) noirs ou blancs. Plus les caractères sont grands plus il faut plus de points pour les décrire. Un fichier décrivant la police doit exister non seulement pour chaque style mais encore pour chaque corps. C'est pourquoi on ne dispose que d'un nombre limité de valeurs. Une autre particularité des polices bitmap est qu'il est difficile de les utiliser dans des tailles autres que celles disponibles. Lorsqu'on tente d'afficher ou d'imprimer cette police dans un corps pour lequel elle n'est pas prévue, il apparaît un *effet d'escaliers* désagréable. En effet, lors de l'agrandissement, l'application se contente d'augmenter la taille des points constituant le caractère en les remplaçant par des petits carrés.

3.2 Police vectorielle

Il existe aujourd'hui un type de police dont le principe consiste non plus à donner le plan de bits des caractères mais plutôt ses descriptions géométriques. Ces polices dites vectorielles, ne dessinent pas le caractère proprement dit, mais son contour en utilisant des lignes droites et des arcs, suivant des algorithmes de gestion de courbes, inventés par le mathématicien Pierre Bézier (voir section 3.3). Tous les corps d'un même caractère sont créés à partir d'une description unique de ce caractère, au quelle on applique des coefficients de déformation. Des traitements seront réalisés avant tout affichage ou impression :

- mise à l'échelle des contours du caractère pour la taille voulue ;

- conversion en bitmap : processus appelé aussi *rastérisation*, qui consiste à traduire le contour d'un caractère en une image en mode points (bitmap).

3.3 Courbes de Bézier

Les courbes de Bézier sont des courbes polynomiales décrites pour la première fois en 1972 par l'ingénieur Pierre Bézier alors qu'il travaillait chez Renault sur des programmes de dessin assisté par ordinateur de capots de voitures.

En général, si on a $n+1$ points $P_0, P_1, ..., P_n$, tels que P_0 et P_n deux points de la courbe, et $P_1, ...P_{n-1}$ soient des points de contrôle ; la courbe de Bézier générée à partir de ces points a l'équation polynomiale de degré n suivante :

$$P(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} P_i \quad avec \quad t \in [0,1]$$

Les plus importantes des courbes de Bézier sont les cubiques. Ils sont utilisées pour le graphisme et dans multiple systèmes de traitement d'image, comme PostScript, Metafont et GIMP pour dessiner des courbes *lisses* à partir de points de contrôle. Les cubiques sont définies par les équations paramétriques :

$$x(t) = a_x + b_x.t + c_x.t^2 + d_x.t^3$$

$$y(t) = a_y + b_y.t + c_y.t^2 + d_y.t^3$$

Celles de Bézier se ramènent au polynôme :

$$P(t) = (1-t)^3.P_0 + 3.t(1-t)^2.P_1 + 3.t^2(1-t).P_2 + t^3.P_3 \quad avec \quad t \in [0,1]$$

Voici quelques propriétés de ces courbes :

- étant donné 4 points P_0, P_1, P_2, P_3, il existe une seule courbe de Bézier passant en P_0 et P_3, ayant $P_0 - P_1$ et $P_2 - P_3$ comme directions de leurs tangentes ; P_1 et P_2 étant les points de contrôle ;
- les points P_0, P_1, P_2, P_3 donnent déjà une idée de la forme de la courbe dont ils sont enveloppe ;
- selon la position relative de ces points, on peut avoir des formes variées (figure 3.1). En déplaçant un seul point, on obtient des variations subtiles ;
- il existe une méthode simple de construction, dérivée du théorème de Casteljou (voir figure 3.2).
  Calculant successivement les coordonnées des points :

$$P_1^{'} = (P_0 + P_1)/2$$

$$A = (P_1 + P_2)/2$$

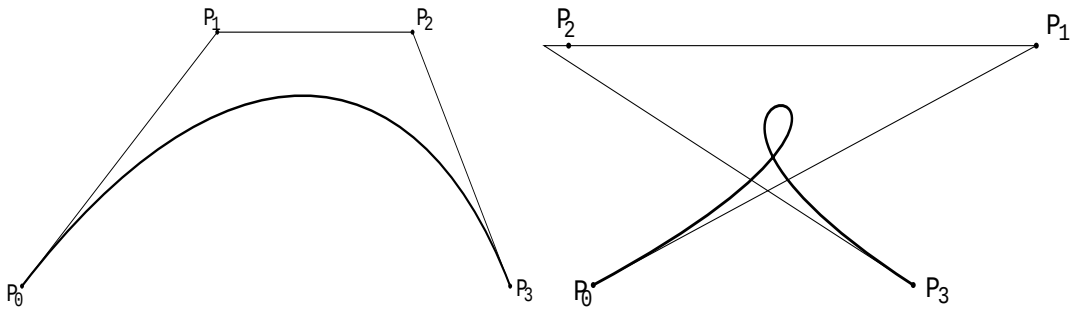

FIG. 3.1 – *Différentes formes de courbes de Bézier cubiques*

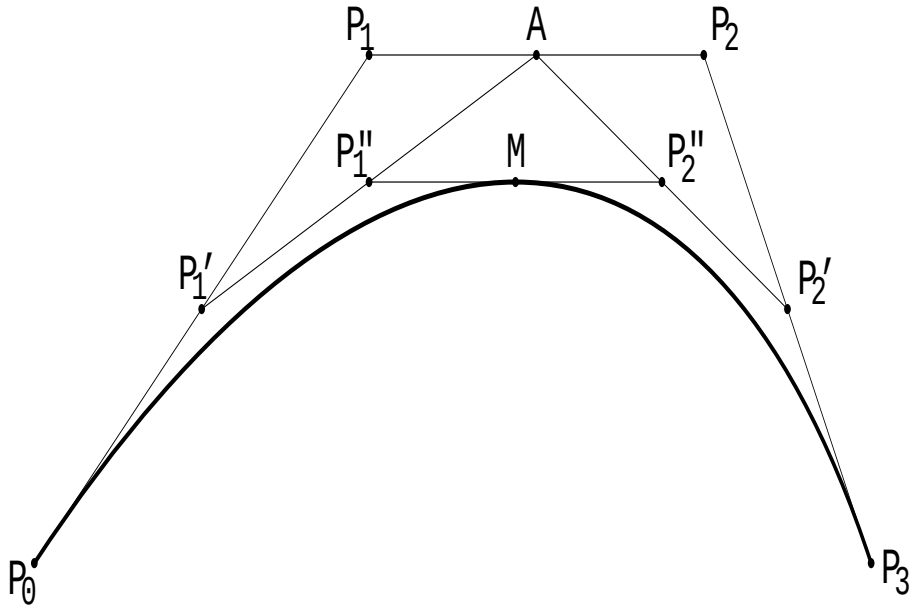

FIG. 3.2 – *Construction récursive de la courbe de Bézier*

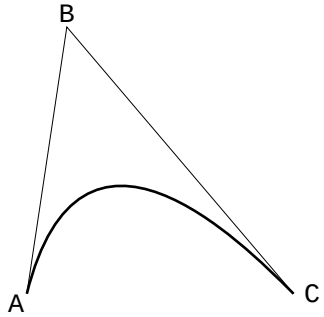

FIG. 3.3 – *Construction de la courbe de Bézier quadratique*

$$P_2^{'} = (P_2 + P_3)/2$$
$$P_1^{"} = (P_1^{'} + A)/2$$
$$P_2^{"} = (A + P_2^{'})/2$$
$$M = (P_1^{"} + P_2^{"})/2$$

et en continuant récursivement sur chacune des moitiés ; basée sur des divisions par deux. Cette méthode est très rapide, ce qui rend l'emploi des courbes de Bézier très efficace.

Les courbes de Bézier quadratiques sont plus simples (voir figure 3.3), elle se construit seulement à partir de trois points. L'équation de la courbe est donc :

$$P(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$$

Ces courbes sont utilisées pour la construction des glyphes d'une fonte au format TrueType.

3.4 Unités particulières

3.4.1 Carré em

Tout caractère est positionné dans un carré em, qui définit un système de coordonnées en deux dimensions dont l'axe x correspond aux déplacements horizontaux et l'axe y aux déplacements verticaux. En pratique, l'emplacement de l'origine (0,0) de ce système de coordonnées répond le plus souvent à certaines conventions liées à la nature de la fonte. Ainsi dans les caractères de type Roman, l'axe x est normalement pris sur la ligne de base, par contre, on le prend au centre pour les caractères qui présentent une certaine symétrie.

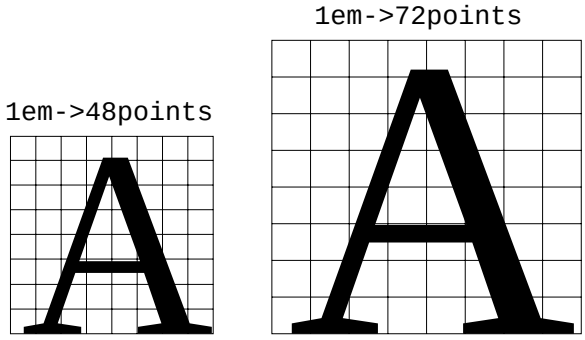

FIG. 3.4 – *Variation de la valeur des unités*

3.4.2 FUnit

Les coordonnées des points de contrôle, déterminant les formes de glyphes, sont définies à l'aide des unités particulières, les FUnits. Une valeur de 1 FUnit correspond à la plus petite unité mesurable dans un carré em, elle a donc une taille variable, relative. En effet, le nombre d'unités dans un carré em (upem), ne change jamais, et ce, quel que soit le corps dans lequel on demande l'affichage ou l'impression des caractères. Lors d'un changement d'échelle, c'est donc la taille des éléments de la grille qui est recalculée, et non le nombre de ces éléments. Soit l'exemple de la figure 3.4, supposons que le carré em soit défini à partir d'un quadrillage de 8 éléments, lorsque le caractère "A" est mis à l'échelle de façon à s'afficher dans un corps 48, ce caractère va occuper un certain espace. Si le corps demandé est maintenant 72, le caractère sera plus grand mais toujours avec un quadrillage de 8 sur 8. De cette façon, les coordonnées exprimées en FUnits sont toujours les mêmes quelle que soit la taille dans laquelle la police doit être rendu.

3.4.3 Mise à l'échelle d'un glyphe

L'étape qui suit, une fois déterminée l'échelle adaptée au corps choisi, consiste à transformer les valeurs relatives (les FUnits) en valeurs absolues en fonction de la résolution du dispositif de sortie utilisé, donc en pixels. Comme les résultats des opérations à effectuer doivent être entiers, les valeurs décimales seront arrondies à l'entier le plus proche. Ainsi, pour convertir un FUnit en pixels l'interpréteur utilise le coefficient suivant :

$$\frac{\text{corps\_en\_points} \times \text{résolution}}{72\ \text{points\_par\_pouce} \times \text{unités\_par\_em}}$$

Le corps est celui qui est demandé par l'application. La résolution est celle du dispositif de sortie en points par pouce. Le nombre 72 est un paramètre fixe qui vient du fait qu'un pouce contient 72 points. Le résultat obtenu donne un facteur d'échelle par lequelle la dimension, exprimée en FUnits, sera multipliée.

3.5 Algorithme de conversion ponctuelle

Supposons, dans un premier temps, que les caractères soient tous identiques (proportionnellement), c'est-à-dire qu'un "e" en corps 10 soit identique à un "e" en corps 1000, mais 100 fois plus petit et que la définition de la grille n'ait pas d'influence sur le dessin définitif du caractère. Le problème est alors : ayant la description d'un caractère par contours, étant donné la grille de l'écran ou de l'imprimante, déterminer quels en sont les points à noircir. On emploi pour cela un programme de conversion ponctuelle (scan conversion algorithm). Nous donnons par la suite les grandes lignes de ce processus :

1. digitalisation et enregistrement du contour en faisant appel au système de coordonnées particulier FUnits ;
2. conversion des unités internes en points (pixels) afin de mettre le contour à l'échelle voulue en fonction du corps à afficher ou à imprimer ;
3. obtention d'une nouvelle description du contour dans une grille dont les cordonnées sont exprimées en points ;
4. détermination des points intérieurs, et des points extérieurs du contour en tenant compte de la direction des courbes ;
5. construction du plan de bit.

Après avoir placé le glyphe sur la grille, l'interpréteur à besoin de connaître la valeur de la chasse, c'est-à-dire de combien il faut avancer la valeur du point courant pour placer le glyphe suivant sans qu'il touche au premier.

L'algorithme de conversion ponctuelle est effectivement appelé pour chaque glyphe à afficher ou à imprimer. Mais comme cet algorithme est coûteux en temps, certains interpréteurs ont prévu un mécanisme de sauvegarde des plans de bits : on ne calcule le plan de bit d'un glyphe donné (ex. un "a" de Times New Roman en corps 12 orienté à 30 degrés) qu'une seule fois puis on le sauvegarde dans une mémoire cache pour qu'il soit utilisé chaque fois qu'on fait appel à ce glyphe dans cette taille.

3.6 Algorithme de hints

En pratique, on ne traite pas le dessin d'un caractère de la même façon dans toutes les situations. Lorsqu'un caractère doit être affiché ou pixellisé dans une petite taille, la forme du caractère n'est pas respectée en raison de la faible résolution du dispositif de sortie, surtout, lorsqu'il s'agit d'une police à empattements qui sont alors grossis.

Pour éviter ces mauvaises adaptations à des petites tailles, les polices numériques comportent des instructions hints qui activent ou désactivent certains pixels pour que le contour de la police soit le plus fidèlement reproduit à l'écran ou lors d'une pixellisation.

3.7 Codage

Chaque caractère disponible sur ordinateur est associé à une valeur numérique, son code. Le code ASCII (American Standard Code for Information Interchange) est l'un des systèmes de codage les plus utilisés en informatique. Il a été défini en 1963 aux États-Unis puis repris par les organismes de normalisation des transmissions internationales de données. Ce système de codage à 7 bits, appelé aussi ASCII standard, autorise la codification de 128 symboles ou commandes. Les ordinateurs ne manipulaient alors que les 26 lettres de l'alphabet anglais. L'ASCII standard ne permettait même pas de reconnaître les lettres accentuées du français par exemple. C'est pourquoi il sera au bout de quelques années étendu sur 8 bits de façon à pouvoir intégrer 256 signes différents répartis de la façon suivante :

- de 0 à 31 : ce sont les codes de contrôles, ils ne correspondent pas à des caractères affichables mais à des commandes servant à indiquer des événements système (retour chariot, tabulation, ...) ;
- de 32 à 127 : on trouve les principaux caractères, majuscules et minuscules, les chiffres ainsi que quelques signes de ponctuation ;
- de 128 à 256 : cette seconde moitié, ou jeu étendu, est nettement moins standardisée. On y trouve normalement les caractères accentués français ainsi que d'autres caractères spécifiques à certains scripts.

En fait, Windows travaille avec un autre code appelé ANSI (American National Standard Institute) qui reprend en grande partie le code ASCII et propose des extensions suivant des codes pages.

Qu'il s'agisse de l'ASCII ou de l'ANSI, les limites sont toujours les mêmes, puisqu'ils ne permettent pas de définir plus de 256 signes. Or, l'extension des outils informatiques et des besoins des utilisateurs pose bien des problèmes :

- les codes des caractères étendus provenant de normes nationales et industrielles contradictoires, ne figurent pas dans les mêmes places, cela a posé des problèmes d'importation et d'exportation de textes ;
- l'espace de 7 bits de l'ASCII ou celui de 8 bits de ses successeurs sont totalement inadéquats pour couvrir toutes les langues du monde entier.

C'est pourquoi le besoin d'un système de codification plus large, complet et normalisé. Actuellement, Unicode est un code universel à 16 bits, il définit un mécanisme de codage des textes multilingues et facilite l'échange des données textuelles à l'échelle planétaire. L'utilisation d'un codage à 16 bits signifie qu'il existe des valeurs de code pour plus de 65000 caractères. Bien que ce nombre soit insuffisant pour coder directement tous les caractères utilisés dans les langues principales du monde, les standards Unicode et ISO/CEI 10646 définissent un mécanisme d'extension appelé UTF-16.

Afin de rendre les jeux de caractères les plus indépendants possibles des systèmes informatiques, la société Adobe a conçue une règle pour ses polices PostScript : elle attribut à chaque caractère un nom donné en clair, par exemple, le symbole monétaire

japonais reçoit l'appellation *yen* en PostScript, mais son code interne dans la police peut être aussi bien 157 en ASCII que 165 en ANSI.

Chapitre 4

Fontes TrueType

Il existe plusieurs types de police dont les principales sont les suivantes :

◇MetaFont de D. Knuth [16] ;

◇PostScript d'Adobe [17] ;

◇TrueType de MicroSoft [3] ;

◇OpenType [5] : nouvelle norme relative aux polices de caractères numériques, développée conjointement par Adobe et Microsoft. Les polices OpenType peuvent contenir des données vectorielles PostScript ou TrueType dans une enveloppe commune. Ces polices sont de type vectoriel.

TrueType est une technologie de fontes numériques créée par *Apple Computer* en 1990, et dont les droits de licence ont été acquis ultérieurement par *Microsoft Corporation*. Les deux sociétés ont indépendamment apporté des enrichissement à cette norme, utilisée aussi bien sous Macintosh que sous Windows depuis sa version 3.1 en 1992. Une police TrueType est composée d'un fichier d'extension TTF, notons qu'il faut bien distinguer le nom de la fonte, et celui du fichier qui le contient, ainsi la fonte Times New Roman a comme nom de fichier times.ttf.

4.1 Structures des fichiers TTF

Un fichier TTF est structuré en tables [3]. Il peut contenir jusqu'à une vingtaine de tables contenants la totalité des données nécessaires à l'affichage et à l'impression des caractères. Certaines de ces tables sont obligatoires et d'autres pas. C'est à leur étude que nous allons nous attacher dans le reste de ce chapitre.

4.1.1 Types de données

Nous nous contentons d'utiliser des outils pour accéder à ces tables ; nous allons retrouver TTFDUMP à plusieurs reprises, il va nous permettre de tirer une grande quantité d'informations. Le tableau qui suit indique les différents types de données qu'on rencontre

dans la description des divers tables.

Types de données	Description
BYTE	entier non signé sur un octet
CHAR	entier signé sur un octet
USHORT	entier non signé sur deux octets
SHORT	entier signé sur deux octets
ULONG	entier non signé sur quatre octets
LONG	entier signé sur quatre octets
FIXED	valeur décimale signé sur deux plus deux octets
FUnit	la plus petite unité mesurable dans un carré em
FWORD	entier signé sur deux octets décrivant une quantité en FUnits
UFWORD	entier non signé sur deux octets décrivant une quantité en FUnit

4.1.2 Structure des tables

Le fichier débute par différentes informations permettant en particulier de situer les tables de définition qui y sont enregistrées. Sa structure est la suivante telle qu'on le retrouve dans les exemples, et telle qu'elle est définie dans le standard TrueType.

Types de données	Nom	Description
FIXED	sfnt version	numéro de version
USHORT	num Tables	nombre de tables

Donc, à partir de l'octet 6, on trouve la liste des informations permettant de retrouver chacune des tables :

Types de données	Nom	Description
ULONG	tag	nom de la table
ULONG	chechsum	somme de contrôle
ULONG	offset	décalage de la table par rapport au début du fichier
ULONG	len	longueur de la table

Le programme TTFDUMP peut nous donner directement ces renseignements en l'appelant par la commande : ttfdump nom-du-fichier.ttf | more. La sortie produite par ttfdump (voir figure 4.1) nous apprend déjà que le fichier times.ttf, par exemple, comporte plus de 320 ko, on peut constater aussi que la première table à accéder est la table *head*, elle se trouve à la position 0000017c en hexadécimal, soit à 380 octets du début, et elle a une longueur de 54 octets.

4.1.3 Liste des tables

Les tables qui sont obligatoires doivent apparaître dans n'importe quel fichier TrueType[1]. Ces tables et leur description sont montrés dans le tableau suivant.

[1] Certaines écritures qui nécessite un traitement contextuel, comme l'arabe et différentes écritures du sous-continent indien, ont besoin d'autres tables tels que GPOS et GSUB qui seront l'objet du chapitre qui suit.

sfnt version :
number of tables : 23

0.	'DSIG'	- chechsum = 0x132244b4,	offset = 0x000539a4,	len =	5552
1.	'GDEF'	- chechsum = 0x514451bc,	offset = 0x00053608,	len =	154
2.	'GSUB'	- chechsum = 0xc8f7ad5b,	offset = 0x000536a4,	len =	734
3.	'JSTF'	- chechsum = 0x6d2a6906,	offset = 0x00053984,	len =	30
4.	'LTSH'	- chechsum = 0xba874f11,	offset = 0x00005920,	len =	1324
5.	'OS/2'	- chechsum = 0x10732def,	offset = 0x000001f8,	len =	86
6.	'PCLT'	- chechsum = 0x59d381e3,	offset = 0x000535d0,	len =	54
7.	'VDMX'	- chechsum = 0x4e236882,	offset = 0x00005e4c,	len =	4500
8.	'cmap'	- chechsum = 0xc1c13a73,	offset = 0x0000313c,	len =	4926
9.	'cvt '	- chechsum = 0xf34dda81,	offset = 0x00008514,	len =	1988
10.	'fpgm'	- chechsum = 0xf534884d,	offset = 0x00007ef4,	len =	1565
11.	'gasp'	- chechsum = 0x00180009,	offset = 0x00000250,	len =	16
12.	'glyf'	- chechsum = 0x2970c245,	offset = 0x00011da0,	len =	250256
13.	'hdmx'	- chechsum = 0x11e0a83b,	offset = 0x0000a178,	len =	31784
14.	'head'	- chechsum = 0xcc11edec,	offset = 0x0000017c,	len =	54
15.	'hhea'	- chechsum = 0x132911c0,	offset = 0x000001b4,	len =	36
16.	'hmtx'	- chechsum = 0x6d8332fd,	offset = 0x00008cd8,	len =	5280
17.	'kern'	- chechsum = 0xa677acd1,	offset = 0x0005216c,	len =	5220
18.	'loca'	- chechsum = 0x0b58504a,	offset = 0x0000447c,	len =	5284
19.	'maxp'	- chechsum = 0x0b9d10a7,	offset = 0x000001d8,	len =	32
20.	'name'	- chechsum = 0xa01fc45f,	offset = 0x00000260,	len =	11995
21.	'post'	- chechsum = 0xd6484739,	offset = 0x0004ef30,	len =	12859
22.	'prep'	- chechsum = 0xe43894a4,	offset = 0x00006fe0,	len =	3859

FIG. 4.1 – *L'entête du fichier times.ttf avec TTFDUMP*

Nom	Description
cmap	correspondance entre les caractères et la définition de leurs contours
glyf	définition des contours de chaque caractère
head	en-tête
hhea	définitions générales
hmtx	données métriques
loca	index permettant de situer les définitions des contours
maxp	profil maximum
name	table des noms
post	informations pour correspondance PostScript

Nous allons maintenant voir le contenu de ces différentes tables - du moins les principales - et nous nous aiderons du programme TTFDUMP avec la commande :

ttfdump -t nom_de_la_table nom_de_la_fonte.ttf | more.

La table *head*

Cette table fournit des informations d'ordre général sur l'ensemble de la fonte. En particulier, il décrit : la date de création de la fonte, le carré em et le nombre des FUnits le définissant. Voyons donc sa structure :

Type de donnée	Nom	Description
FIXED	table version	numéro de version
FIXED	fontReversion	défini par le fondeur
ULONG	checkSumAdjustment	somme de contrôle
ULONG	magicNumbe	utilisé pour le Macintosh
USHORT	flags	drapeaux (voir ci-aprés)
USHORT	unitsPerEm	le nombre d'unités par em
LONG	created	date de création de la fonte
LONG	modified	date de la dernière mise à jour
FWORD	xMin	coordonnée en x minimale
FWORD	yMin	coordonnée en y minimale
FWORD	xMax	coordonnée en x maximale
FWORD	yMax	coordonnée en y maximale
USHORT	macStyle	indique le style
USHORT	lowestRecPPEM	plus petite taille lisible en points
SHORT	fontDirectionHint	sert à l'indication des "hints"
SHORT	indexToLocFormat	utilisé dans les calculs des adresses
SHORT	glyphDataFormat	0 à l'heure actuelle

Les drapeaux sont des octets dans lesquels chacun des bits joue son rôle, dans la table head. L'indication flags se décompose de la façon suivante :

- bit 0 : la ligne de base pour la police a pour coordonnée y=0 ;
- bit 1 : la ligne de gauche a pour coordonnée x=0 ;
- bit 2 : les instructions de programme peuvent dépendre du corps demandé ;

- bit 3 : force les calculs internes pour la mise à l'échelle des contours à n'utiliser que les valeurs entières ;
- bit 4 : le calcul de la chasse peut être modifié par programme.

Les valeurs (xMin,yMin) et (xMax,yMax) sont en FUint, ils décrivent le carré em.

La table *cmap*

Cette table permet d'établir une correspondance entre les codes des caractères et la définition de leurs contours dans la table glyf. Elle peut contenir plusieurs sous-tables servant à spécifier de nombreux schémas d'encodage. La table cmap débute de la façon suivante :

Types de donnée	**Description**
USHORT	numéro de version (0)
USHORT	nombre de tables d'encodage

On trouve ensuite autant de sous-tables que de schémas d'encodage :

Types de donnée	**Nom**	**Description**
USHORT	PlatformID	identification de la plate-forme
USHORT	EcodingID	type d'encodage pour cette plate-forme
USHORT	'cmap' Offset	décalage de la sous-table par rapport à la table cmap
USHORT	Length	longueur de la table
USHORT	Format	numéro de format
USHORT	segCount	sert au calcul de la correspondance entre code de caractère et encodage

On trouve par la suite la sous-table de correpondance dont voici un extrait de la fonte times :

Char	0x0041	->	Index	65
Char	0x0042	->	Index	66
Char	0x0043	->	Index	67
Char	0x0044	->	Index	68
Char	0x0045	->	Index	69
Char	0x0046	->	Index	70

La table d'index nous apprend que le A majuscule (code 0x0041 ou décimal 65) est associé au contour de rang 65. Un index nul signifie simplement que le caractère possédant ce code n'est pas défini dans cette fonte.

La table *glyf*

C'est la table qui contient la totalité des descriptions du dessin des caractères, ou plus exactement de leurs contours, elle possède autant d'entées que de caractères définis. Nous venons de voir que l'on y accédait à l'aide d'un système d'index à partir de la table cmap. Nous allons simplement donner le format de base sans entrer dans les détailles.

En effet, nous allons décrire par la suite un logiciel spécialisé tel que **Font Creator** qui va nous permettre de y accéder et la modifier.

Les informations décrivant chaque contour sont structurées de la façon suivante :

Type de donnée	**Nom**	**Description**
SHORT	numberOfContours	nombre de contours utilisés pour décrire le glyphe, une valeur négative indique un caractère composite
FWORD	xMin	coordonnée minimale en x
FWORD	yMin	coordonnée minimale en y
FWORD	xMax	coordonnée maximale en x
FWORD	yMax	coordonnée maximale en y

Les valeurs (xMin,yMin) et (xMax,yMax) sont bien entendu en FUnits, ils définissent le rectangle délimitant le caractère. Cet en-tête est en suite suivi d'un ensemble de données :

Type de donnée	**Nom**	**Description**
USHORT	EndPoints	tableau indiquant le nombre de points terminaux sur chaque contour
USHORT	instructionlength	le nombre d'instructions qui suivent
BYTE	flags[n]	ensemble de n drapeaux, chacun étant associé à un point terminal

La table des drapeaux (flags) est une table de 8 bits, où chaque bit porte une information :

Bit	**Drapeau**	**Description**
0	On Curve	si 1, le point est sur la courbe, sinon en dehors (On/Off)
1	X-Short	si 1, la coordonnée en X est de longueur 1 octet, sinon elle est de longueur 2 octets
2	Y-Short	même chose que X-Short pour la cordonnée en y
3	Repeat	si 1, l'octet suivant spécifie le nombre de fois où ce groupe de drapeaux doit être répété, ce qui permet de définir moins de drapeaux que de points
4	X-Short	indique le signe de la coodonnée en X
5	X-Short	Idem pour la coordonnée en Y
6,7	Reserved	reservés (égaux à 0)

La table *name*

Cette table contient toutes les informations sur le nom de la fonte, sa famille, le nom du fabriquant, et ainsi de suite.

La table *loca*

Cette table contient les décalages à appliquer à l'adresse de la table glyf pour retrouver le début de la déscription de chaque contour.

La table *maxp*

Cette table définit les besoins techniques en mémoire pour la police.

La table *hhea* et *hmtx*

Ces deux tables contiennent l'ensemble des informations métriques indispensables pour le rendu de la fonte, tels que la dimension des ascendants au dessus de la ligne de base, et celle des descendants, ainsi que la valeur de la chasse pour chaque caractère.

La table *post*

Dernière table obligatoire, contient des informations nécessaires à l'utilisation des fontes TrueType sur des imprimantes PostScript.

Les autres tables ne sont obligatoires que dans le sens de définir au mieux le comportement de la police, notamment dans les cas extrêmes. Il s'agit des tables *cvt, fpgm* et *prep*. D'autres sont destinées à améliorer le rendu en fonction du terminal de sortie ou des indications de hints (tables *hdmx, LTSH, VDMX*). La table *kern* sert à définir les paires crénés, autrement dit, les couples de caractères qui doivent être rapprochés ou éloignés.

Chapitre 5

Fontes OpenType

OpenType est un format de police "ouvert", utilisable sur tous les systèmes d'exploitation, développé conjointement par Adobe et Microsoft. C'est une extension du format TrueType, qui combine les deux technologies, TrueType et PostScript et offre des possibilités typographiques importantes.

OpenType peut utiliser soit des contours TrueType, soit des contours PostScript. Nous n'allons nous attacher qu'au premier cas. Le fichier de la fonte a donc une structure de tables concaténées :

- les tables nécessaires à l'affichage et ceux qui améliorent le rendu de la fonte sur le dispositif de sortie sont toujours ceux cités dans le chapitre précédent ;
- il y a également d'autres tables appelés OTL (*OpenType Layout tables*) contenant des informations supplémentaires concernant les ligatures, la substitution de glyphes, le positionnement des signes diacritiques, la justification, et bien d'autres informations qui soutiennent d'autres fonctions typographiques avancées :

Étiquette	**Description**
BASE	ligne de base
GDEF	définition de glyphes
GPOS	positionnement de glyphes
GSUB	substitution de glyphes
JSTF	justification

5.1 La table GSUB

La table de substitution de glyphes, GSUB, contient des informations portées sur la substitution contextuelle de glyphes et les ligatures. Cela donne la possibilité de gérer des systèmes d'écriture complexes comme l'arabe et l'hébreu. En effet, dans un processus de composition de texte arabe chaque lettre peut être écrite de plusiaurs manière différente suivant sa position dans le mot : initiale, médiane, finale, isolée.

La table GSUB décrit six types de substitutions, qui sont largement répandues dans

la typographie internationale :

- substitution simple : remplace un glyphe simple par un autre, ceci est employé, par exemple, pour rendre les variantes de position des caractères dans un texte arabe (ex. ت ← ت) ;
- substitution multiple : remplace un glyphe par plusieurs glyphes, ceci est employé pour décomposer les ligatures (ex. œ → oe) ;
- ligature : remplace une suite de glyphes par un seul, ceci est employé pour rendre les ligatures linguistiques (ex. la paire LAM-ALEF ne se produit pas par une simple concaténation du LAM initial et ALEF final) ;
- substitution contextuelle (ex. voir section 6.4.4) ;
- substitution enchaînée ;
- substitution alternée.

5.2 La table GPOS

La table de positionnement de glyphes, GPOS, contient des informations sur le positionnement en X et en Y de glyphes par rapport à d'autres. Cela permet de placer les signes diacritiques par rapport à des glyphes de base. Il permet aussi de gérer des systèmes d'écriture qui requièrent à la fois un positionnement et une substitution contextuels comme le persane.

La table de GPOS soutient huit types d'actions pour placer et attacher des glyphes :

- ajustement simple : permet de placer les indices supérieurs et inférieurs ;
- ajustement de paire : pour placer deux glyphes l'un par rapport à l'autre ;
- positionnement d'un signe diacritique par rapport à un glyphe de base ;
- positionnement d'un signe diacritique par rapport à une ligature ;
- positionnement d'un signe diacritique par rapport à un autre ;
- attachement cursif : permet de gérer des écritures comme le persane où les glyphes sont attachés le long d'une ligne descendante ;
- positionnement contextuel : permet de positionner un ou plusieurs glyphes suivant un contexte ;
- substitution contextuelle enchaîné : permet de placer un glyphe suivant un contexte enchaîné.

5.3 Organisation de tables

Les deux tables GSUB et GPOS emploient la même structure arborescente de données pour décrire leurs fonctionnalités typographiques. On trouve donc la liste d'écritures (ScriptList), la liste de langues (LanguageList), la liste de rubriques (FeatureList) puis la liste de consultations (LookupList) (voir figure 5.1) :

- script : spécifie l'écriture ou le type de l'alphabet (ex. arabe, latine, ...) ;

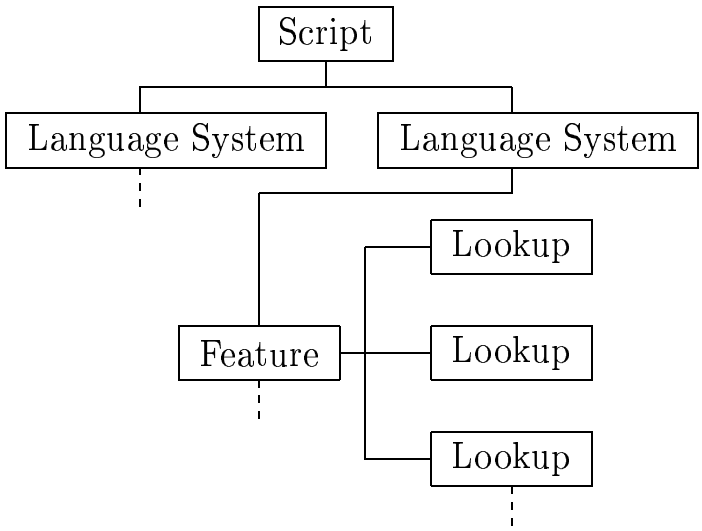

FIG. 5.1 – *Structure arborescente des tables GSUB et GPOS*

- language : définit la langue (ex. arabe, français, ...)[1] ;
- feature : décrit le type de fonctionnalité typographique à employer (ex. placement des signes diacritiques, ...) ;
- lookup : contient les données requises pour la mise ou point de la substitution ou du positionnement.

[1]une écriture peut définir plusieurs langue, par exemple l'alphabet latin est utilisé pour le rendu de la langue française, anglaise, ...

Chapitre 6

Création d'une fonte arabe

Dans ce chapitre, nous allons décrire la procédure utilisée pour créer une fonte arabe au format TrueType. Nous avons choisi le style calligraphique Naskh نسخ, qui est le style le plus adopté pour la confection des textes arabes.
Normalement, la création d'une police arabe commence par une longue phase de dessin effectué à la main par un calligraphe ou à l'aide d'un outil informatique. Dans le premier cas, il faudra ensuite passer les modèles au scanner, puis les transformer en objets vectoriels. Dans le deuxième cas, les outils informatiques semblent non satisfaisants.

Une fois la conception terminée, les caractères doivent être traités et analysés à l'aide d'un programme particulier afin de créer le fichier définissant la police.

6.1 Écriture arabe

L'arabe est l'une des langues sémitiques les plus importantes de l'histoire de l'humanité. Et la plus conservatrice depuis sa mise au point par le Saint Coran.

L'écriture arabe est plus qu'un moyen de communication, c'est un art de calligraphie très raffiné. Les différences entre le système alphabétique arabe et latin sont nombreuses. Citons par exemple :

- l'alphabet arabe est composé de 28 lettres :

  أ ب ت ث ج ح خ د ذ ر ز س ش ص ط ض ظ ع غ ف ق ك ل م ن ه و ي

- le système des signes diacritiques des points joue un rôle de premier ordre, dans la mesure où certaines lettres ne se distinguent que par leur présence, leur nombre et leur position ;
- l'arabe s'écrit et se lit de droite à gauche alors que les systèmes romans se déroulent dans le sens inverse ;
- la cursivité de l'écriture ;
- les lettres prennent différentes formes suivant leur position dans le mot : initiale, médiane, finale ou isolée. L'analyse contextuelle permet de déterminer ces variantes :

ي و م ← يوم

- l'écriture arabe est riche en matière de ligatures linguistiques :

ل ا ← لا

- la justification de la ligne dans un paragraphe se fait avec la Kashida (allongement des lettres) contrairement à l'utilisation des blancs entre les mots, et la césure dans l'écriture romane ;
- Les formes minuscules et majuscules des lettres sont inexistantes ;
- les pays arabes font usage de deux types de graphies pour représenter les chiffres ;
- les lettres de l'alphabet arabe sont toutes des consonnes, la voyellisation du texte se fait à l'aide de signes diacritiques placés au-dessus ou au-dessous de la lettre pour une voyellisation courte, ou à l'aide de l'une des trois lettres consonnes ALEF, WAW ou YAH placées après la lettre à voyeller pour une voyellisation longue.

6.2 Réalisation

La richesse de l'imagination des calligraphes a conduit à rendre très coûteuse l'imprimerie des textes arabes à l'aide des techniques traditionnelles, les cases pouvaient contenir jusqu'à 1000 caractères [14]. En informatique, les ligatures rendent difficile la reconnaissance, l'affichage et l'impression des caractères.

Longtemps les typographes essaient d'imiter l'écriture manuscrite dans ses liaisons les plus complexes. On abandonna ces ligatures et on adopta le principe de séparation des caractères à fin de limiter le nombre et donc les frais d'impression, tout en essayant de produire fidèlement l'écriture manuscrite avec ses ligatures et ses lettres dont la forme varie selon leur place dans le mot.

Si l'on veut être minimaliste, pour écrire en arabe il suffit de connaître les quatre formes de chaque lettre et de savoir qu'on a des ligatures linguistiques et des ligatures contextuelles, comme ensemble minimal de règles. Mais la calligraphie arabe va beaucoup plus loin de ces règles du premier degré, qui font partie de la grammaire de base. On a aussi les ligatures esthétiques qui font l'élégance de l'écriture arabe.

L'écriture arabe connaît trois types de ligatures :

- les ligatures linguistiques ; ce sont celles qui sont indispensables pour l'écriture d'une langue donnée, et obéissent à des règles grammaticales (la grammaire étant liée à la langue et non pas à l'écriture) ;
- les ligatures contextuelles ; ce sont des graphies qui se produisent sous certaines conditions, par exemple en début de mot. Elles sont obligatoires dans le contexte d'une écriture donnée, indépendamment des langues utilisant cette écriture ;
- les ligatures esthétiques ; ce sont des graphies optionnelles qui existent pour des raisons esthétiques, de lisibilité, et/ou de tradition. Elles sont utilisées dans la plaéographie et la typographie de qualité. En arabe, on distingue entre des ligatures esthétiques de deuxième (deux lettres) et de troisième degré (trois lettres).

On désire confectionner une fonte arabe respectant les règles calligraphiques de style

Naskh. Les différentes méthodes existantes sont :

- dessiner les caractères à l'aide des outils informatiques ;
- scanner des modèles dessiner à la main par un calligraphe.

Notre conception est faite via une scannarisation de modèles dessinés à la main, puis améliorés à l'aide des outils informatiques. On s'est basé sur deux logiciels de typographie assistée par ordinateur **Font Creator** et **Visual OpenType Layout Tool**. Le premier logiciel permet de transformer les images scannées en images vectorielles puis de sauvegarder la police au format TrueType pour Windows et Macintosh. Le deuxième logiciel permet d'ajouter les fonctionnalités typographiques requises pour le rendu correct d'un fonte arabe. Dans la suite on utilise la notation {<بج>} pour représenter la famille de tous les caractères crées à partir du même glyphe de base بج, ainsi {<بج>}={بج,تج,ثج,نج}.

Les glyphes de base utilisés dans notre fonte sont donc :

- les 28 lettres de la langue arabe ;
- les différentes variantes de positions : isolée, initiale, médiane et finale ;
- L'analyse contextuelle est utilisée pour déterminer la forme des lettres, il peut être utiliser pour déterminer la présence de ligatures. Mais elle ne peut être aussi utilisée pour déterminer quand est ce que le signe HAMZA ء combiné avec une voyelle longue, est au-dessus ou au-dessous du ALEF ou sur la ligne. En effet, cela nécessite que le texte soit voyellisé.
  Cette correspondance utilise deux caractères au lieu d'un seul. En fait les écritures HAMZA ON ALEF أ ـأ, HAMZA UNDER ALEF إ ـإ, HAMZA ON WAW ؤ ـؤ, HAMZA ON YEH ئ ـئ ئـ ـئـ, correspondent à la superposition de deux caractères : le caractère HAMZA et l'un des caractères ALEF, WAW ou YEH ;
- les chiffres arabes utilisés au Machrek ٠١٢٣٤٥٦٧٨٩ ;
- la forme médiane de la lettre BEH {<ـبـ>}, et des autres lettres de sa famille, peut prendre la forme {<ـبـ>}, selon le contexte, même chose à dire pour la forme initiale de la lettre BEH ;
- les ligatures linguistiques de type : {<لا>} et {<ـلا>} ;
- des ligatures esthétiques de deuxiéme degré, de type : {<بح>} {<بم>} {<بي>} {<في>} {<لح>} {<لم>} {<بن>} {<بى>} {<تر>} {<بج>} {<بم>} {<جم>} {<سم>} {<لج>} {<مخ>} {<مم>} {<هم>} {<لا>} {<ـلا>}, on obtient ainsi le mot نهر au lieu de نهر ;
- des ligatures esthétiques de troisiéme degré, de type : {<لله>} {<محم>} {<لح>}, le nom "Mohammed" s'écrira donc محمد au lieu de محمد ;
- un emplacement pour la kashida ;
- les différentes signes diacritiques ًٌٍَُِّْ ;
- les signes diacritiques de type ٌّ s'obtient en combinant les deux marques ٌ et ّ.

Afin de réduire le coût de la confection et de diminuer la taille de la fonte, on profite de l'outil de création de caractères composites disponible par *Font Creator* pour créer les glyphes voyellés avec des points ou des accents placées au dessus ou au dessous comme

ر et ز provient du même caractère de base ر.

Pour préciser la prononciation du texte purement consonantique, les voyelles brèves et les signes orthographiques sont ajoutés au-dessus ou au-dessous des consonnes. Le texte arabe peut être complètement, partiellement ou non voyellé. La voyellisation pose le problème de la position des signes de voyelle par rapport à la lettre de base. Après étude du système de placement des marques diacritiques des fontes TrueType et OpenType existantes, on constate les choses suivantes :

- dans la fonte Times New Roman, les signes diacritiques prennent la même place et la même hauteur par rapport à la ligne de base est ce quelque soit le glyphe de base ;
- la fonte Traditional Arabic utilise deux glyphes pour chaque signe diacritique, chacun a une hauteur différente pour marquer deux hauteurs selon le caractère de base.

Les signes de voyellisation arabes prennent des hauteurs différentes, non seulement, suivant le glyphe de base mais aussi suivant le contexte. On peut considérer la lettre arabe comme un aimant de la marque diacritique. On va utiliser les fonctionnalités de la table GSUB pour placer correctement les signes diacritiques avec des hauteurs variantes.

6.3 Font Creator

Font Creator proposé par la société High Logic est un logiciel à interface graphique permettant la création des polices TrueType ou la modification des polices TrueType existantes. Il faudrait plus d'un chapitre pour avoir une vision complète de cet outil. Nous proposons la consultation de la page Web de High Logic sur laquelle est proposée un manuel de référence plus détaillé [6]. Citons simplement quelques une de ses caractéristiques principales :

- l'édition des caractères, avec évidemment la possibilité de travailler sur les contours grâce aux courbes de Bézier ;
- la possibilité d'utiliser la barre d'outils, pour dessiner des caractères d'une façon WYSIWYG ;
- la transformation automatique d'une image d'un caractère en mode bitmap de format bmp - par exemple passée au scanner - en une image vectorielle ;
- la création des caractères composites ;
- les informations concernant le crénage peuvent être lues, modifiées ou créées d'une manière souple.

Voyons donc à quoi ressemble l'interface de ce logiciel. Lançons **Font Creator**, puis ouvrons une police TrueType existante. Alors le jeu de caractères qu'elle contient s'affiche sous forme d'un tableau où chaque cellule a une légende et une partie qui contient le dessin du caractère (voir figure 6.1). Pour distinguer les types des glyphes, les légendes ont différents couleurs :

- gris : caractère vide ;

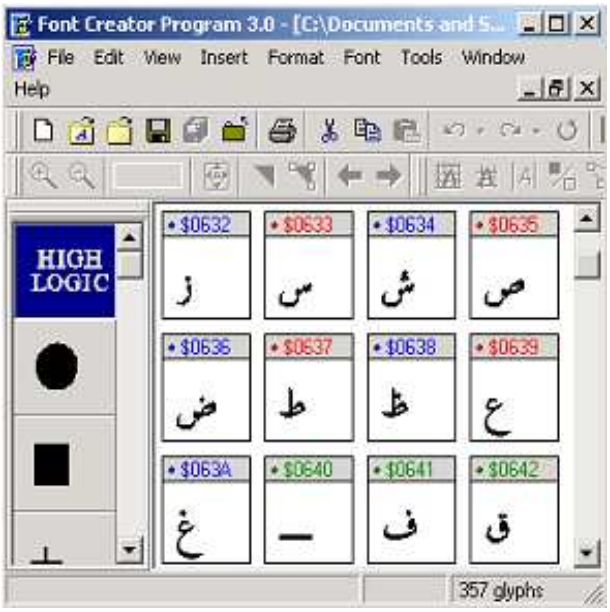

FIG. 6.1 – *Ouverture de la police Arabic font avec Font Creator*

- rose : caractère vide, utilisé par d'autre(s) glyphe(s) composé(s) ;
- vert : caractère simple ;
- rouge : caractère simple utilisé par d'autre(s) caractère(s) composé(s) ;
- bleu : caractère composé d'autres caractères simples ou composés ;
- pourpre : caractère composé utilisé par d'autre(s) caractère(s) composé(s).

6.3.1 Édition d'une fonte

Après avoir choisi **File→New**, on obtient un tableau de caractères vide. Il est important dans un premier temps de nommer la nouvelle fonte. Comme on l'a déjà noté, il faut distinguer le nom de la fonte du nom du fichier qui la contient.

Format→Naming donne la possibilité de donner un nom utilisable par Windows et un autre par Macintosh.

6.3.2 Édition des glyphes

Font Creator est doté de fonctions de dessin vectoriel et accompagné d'une bibliothèque de modèles, faits que l'on puisse les utiliser pour agrémenter les compositions. Un double clic sur une cellule du tableau, ramène vers une grille sur laquelle on peut dessiner notre caractère. La fenêtre d'édition propose des différents outils simples à manipuler. Sélectionner **Contour** dans le menu **Insert**, un clic gauche de la souris sur la grille crée un point sur le contour (on-curve), et un clic droit crée un point de contrôle

employé pour céer des courbes de Bézier. L'idée d'utiliser les outils graphiques disponibles par Font Creator pour produire des glyphes arabes est loin d'être satisfaisante de point de vue calligraphique, alors on a opté pour une réalisation via scannarisation. En effet, Font Creator possède une option de vectorisation d'images bitmap en sélectionnant **Import Image** du menu **Tools**. Des retouches sur l'image vectorielle importée sont indispensables.

Nous a aussi profité des glyphes déjà existants dans une autre police que nous avons copié dans la nouvelle fonte. Toutes les formes de toutes les variantes des lettres ainsi que les ligatures doivent être créer une par une.

6.3.3 Compatibilité avec Unicode

Font Creator est compatible avec le standard de codage universelle Unicode, ce qui donne la possibilité de faire la correspondance entre l'emplacement des caractères au clavier et la définition de leurs contours dans le tableau des glyphes.

Pour donner un code à un caractère, cliquer droite sur la cellule qui contient le dessin du caractère puis sélectionner **Properties→Mappings**.

En Unicode l'arabe occupe principalement l'intervalle des codes [0600,06FF] en hexadécimal. Un autre intervalle est réservé pour des ligatures et des formes contextuelles.

6.3.4 Caractères composites

Les caractères composites sont simplement une combinaison de deux autres glyphes. Généralement, un glyphe de base et un ou plusieurs signes diacritiques (points, accents, ...) placées au-dessous ou au-dessus du glyphe de base.

On peut créer des caractères composites, en cliquant à droite dans la fenêtre d'édition de glyphes, en sélectionnant **Add Composite Glyph Member**, dans la fenêtre **Composite Glyph Properties**. On peut modifier les positons des caractères composantes, les redimensionner... Nous avons utilisé cette option pour créer des caractères composites, par exemple la lettre KHAH est composée de la lettre HAH et d'un point placé au-dessus :

ح+ ˙ ← خ

6.3.5 Limites de Font Creator

Bien sûr les explications qui précèdent sont un peu optimistes, il faut compter plusieurs jours de travail pour créer un jeu de caractères complet, en tenant compte des différentes retouches. La situation se complique quand il s'agit d'une fonte arabe, du fait qu'un texte arabe est plus qu'un ensemble de caractères concaténés.

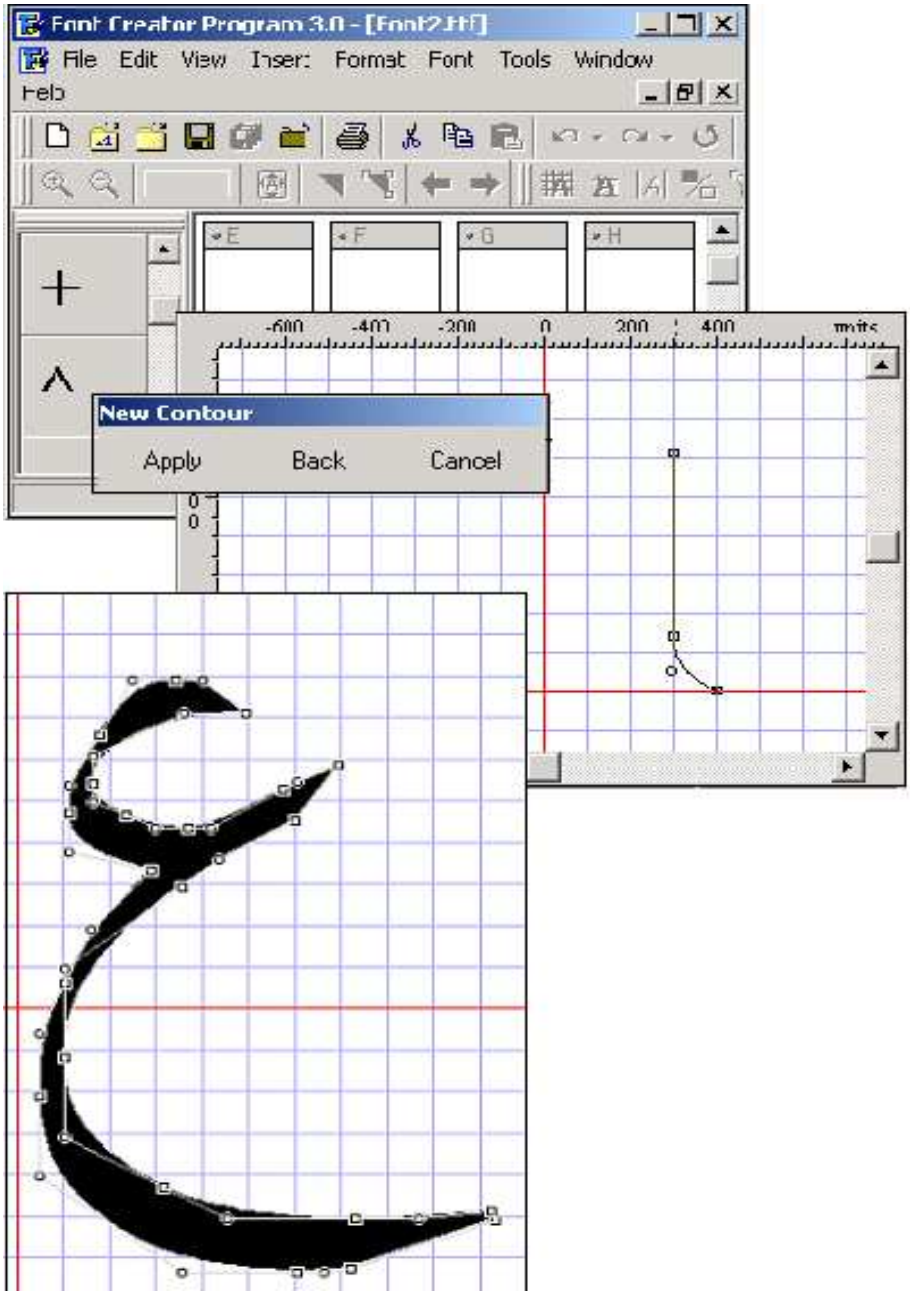

FIG. 6.2 – *Profiter des outils graphiques pour dessiner des glyphes*

L'outil dont on dispose ne nous permet pas de gérer ni la table de substitution de glyphes GSUB, ni la table de positionnement de glyphes GPOS. Bien sûr, Il nous faudra un autre utilitaire pour accomplir le travail, c'est ce qui fera l'objet de la section suivante.

6.4 Visual OpenType Layout Tool

Visual OpenType Layout Tool (VOLT) de Microsoft, fournit une interface utilisateur, libre, facile à utiliser, pour ajouter des tables OTL (OpenType Layout Tables)[1] à des polices TrueType. Citons les caractéristiques principales de VOLT :

- éditer la structure arborescente de GSUB et GPOS d'une manière flexible ;
- établir la correspondance entre les glyphes et leur noms ainsi que leur codes ;
- définir des goupes de glyphes qui subissent le même traitement contextuel ;
- définir les paires de crénage.

6.4.1 Mise au point des ligatures

Lançons VOLT et ouvrons la fonte déjà créer par Font Creator. En cliquant sur le bouton **Edit Glyphs**, on retrouve notre jeu de caractères où chaque glyphe a un nom, un code et un type[2].

Commençons par éditer l'ensemble des substitutions et des ligatures requises pour le rendu correcte de notre fonte, autrement dit, les différents nœuds de la table GSUB. le fichier nommé TAGT.txt est disponible sur le site de Microsoft [8], contient la liste des "Scripts", des "Languages" et des "Lookups" reconnus par VOLT, ainsi que leurs étiquettes. La consultation de ce fichier est indispensable pour se rendre compte bien de ce qu'il faut mettre. On procède alors comme suit :

- **Add Script** : à l'aide de ce bouton, on place la racine de la structure arborescente de notre table ; soit l'alphabet arabe. Après consultation du fichier TAGT.txt, on se rend compte qu'il faut mettre "Arabic" ;
- **Add Language** : l'arabe est la langue par defaut de l'alphabet "Arabic" ;
- **Add Feature** : les rubriques indispensables pour le rendu des différentes variantes de positions sont : Initial Forms, Medial Forms et Terminal Forms.
  Pour la construction des différentes ligatures, on a besoin du rubrique "Standard Ligatures" ;
- **Add Substitution** : nommer le "lookup" concerné puis valider en appuyant sur la touche Entrer ;
- **Edit Lookup** : sélectionner le "Lookup" à éditer puis appuyer sur le bouton **Edit Lookup**, dans la fenêtre d'édition de substitution qui apparaîtra, saisir les données requises : les glyphes en question, le sens de déroulement et le contexte.

[1] voir le chapitre précédent.
[2] Simple, Mark, Ligature, Component.

6.4.2 Mise au point des signes diacritiques

Les signes diacritiques jouent un rôle très important dans un texte arabe, ils marquent les voyelles brèves. Les voyelles longues sont marquées à l'aide de l'une des trois lettres consonnes ALEF, WAW ou YEH placées aprés la lettre à voyeller.

Pour placer les signes diacritiques avec Visual OpenType Layout Tool, on introduit dans l'arbre déjà structuré le rubrique "Mark Positioning" concernant le positionnement de glyphes.

6.4.3 Groupes de glyphes

Pour accélérer l'ensemble des traitements, VOLT fournit un outil permettant de construire des groupes de glyphes qui subissent le même traitement contextuel. Par exemple, pour placer les signes diacritiques ; au lieu de faire un traitement pour chaque signe, on construit deux groupes de glyphes :

1. le groupe "AboveMarks", contenant les marques placées au-dessus ;
2. le groupe "BelowMarks", contenant les marques placées au-dessous.

6.4.4 Substitution contextuelle

Pour écrire en arabe, il ne suffit pas de connaître les quatre formes de chaque lettre et de savoir les différentes ligatures linguistiques. Les règles de la calligraphie vont beaucoup plus loin. Certaines lettres peuvent prendre différentes formes suivant le contexte, ainsi la lettre médiane BEH {<ـبـ>} doit avoir un allure un peu plus aiguë {<ـبـ>} dans le cas où elle interpose deux lettres aiguës.

Chapitre 7

Conclusion

Actuellement, on dispose d'une fonte arabe au format TrueType de style Naskh basée sur une fonte déjà existante. Les contributions principales apportées à cette fonte, à la lumière des règles calligraphiques du style Naskh, sont :

- des corrections sur des ligatures de deuxième degré ;
- l'ajout des ligatures de troisième degré ;
- le placement approprié des signes diacritiques.

L'avancement des étapes de la confection de cette fonte a été l'aboutissement des travaux suivants :

- une élaboration des règles de la typographie numérique ;
- une étude des règles de la calligraphie arabe du style Naskh en vue de leur formalisation ;
- une étude des différents formats de polices TrueType, OpenType et PostScript ;
- une étude comparative des fontes TrueType et OpenType déjà existantes ;
- la prise en main des logiciels de la typographie numérique, en particulier, **Font Creator** et **Visual OpenType Layout Tool**.

Rappelons que les exemples de textes introuduits dans ce rapport, sont tous composés à partir de notre fonte confectionnée. Comme nous pouvons le constater, les règles de la calligraphie arabe sont loin d'être tout à fait respectées. Le développement est encore ouvert sur d'autres travaux :

- la multi présentation, pour donner à l'utilisateur plus de liberté à choisir le type et le degré de ligature voulus ;
- la justification des textes en utilisant la Kashida d'une façon curviligne ;
- l'allongement dynamique des signes diacritiques suivant la longueur de la Kashida ;
- le positionnement des signes diacritiques suivant la hauteur des caractères ;
- la réalisation d'une fonte pour les mathématiques arabes.

Bibliographie

[1] Jacque André, *Métriques des fontes en typographie traditionnelle*, Cahiers GUTenberg n°4, 1989.

[2] Jacque André, *Caractères numériques : introduction*, Cahiers GUTenberg n°26, 1997.

[3] *http ://developer.apple.com/fonts/TTRefMan/.*

[4] Daniel Rougé, *Police TrueType pour Windows*, SYBEX. 1994.

[5] *http ://partners.adobe.com/asn/tech/type/opentype/index.jsp.*

[6] *http ://www.high-logic.com/manual/index.html. The Font Creator Programm Manual.*

[7] Azzeddine Lazrek, *Vers un système de traitement du document scientifique arabe,* Thèse de Doctorat d'État, Université Cadi Ayyad, Faculté des Sciences Marrakech, 2002.

[8] *http ://www.microsoft.com/typography/devlopers/volt/defaut.htm.*

[9] Yannis Haralambous, *Tour du monde des ligatures*, Cahiers GUTenberg, n° 22, 1995, pp. 69-70.

[10] Azzeddine Lazrek, *Aspects de la problématique de la confection d'une fonte pour les mathématiques arabes*, Cahiers GUTemberg n° 39-40, 2001.

[11] Yannis Haralambous, *Une police mathématique pour la société Mathématique de France : le SMF Baskerville*, Cahier GUTemburg n° 39, 1999, pp. 5-19.

[12] Thomas Milo, *ALI-BABA and the 40 Unicode Characters - Towards the Ideal Arabic Working Environment*, TUGBoot, Volume 24, 2003.

[13] *http ://www.ucam.ac.ma/fssm/rydarab/.*

[14] Rachid Zghibi, *Le codage informatique de l'écriture arabe : d'ASMO à Unicode et ISO/CEI 10646*, Université Paris 8.

[15] Charles Bigelow et Kris Holmes. *Création d'une police Unicode,* Cahiers GUTemberg n°20, 1995.

[16] Donald Ervin Knuth, *The METAFONTbook*, Addison-Wesley, 1986.

[17] Adobe Systems Incorporated, *Postscript Language Reference Manual*, Addison-Wesly, 1984.

[18] *http ://www.linux.org.sa/.*

[19] محمد زكي و محمد خضر. الحروف العربية والحاسوب. مجمع اللغة العربية الأردني. 1996.
Mohamed Zaki, Mohamed Khader, Institut jordanien de la langue arabe, 1996.