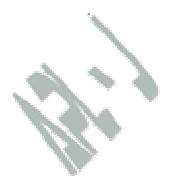
Gestion de gros fichiers binaires (images) en APL*PLUS III

par Gérard A. Langlet



L un des nombreux problèmes que nous avons rencontrés en APL concerne l'impression d'images, constituées en fait de grosses matrices binaires.

Une image de 512⍟512 points (pixels, soit noirs soit blancs) comporte déjà plus de 250.000 points. Pourtant, avec une imprimante-laser de résolution 600 points au pouce, cette image ne représentera qu'un petit carré d'environ 2 cm de côté.

Il vaut mieux disposer d'une implantation d'APL qui gère les bits en bits, c'est pourquoi nous opérerons cette fois en APL*PLUS III. (L'adaptation en Dyalog.APL ne doit pas poser problème).

Nous traiterons en parallèle deux cas, celui des fichiers *.BMP (en anglais « BitMaP ») et celui des fichiers *.TIF (« Tagged Image Format »). Ces deux formats permettent d'accéder aux différents éditeurs graphiques (« Brush », « Hpbrush », etc...) disponibles sur le marché sous **Windows**, et dont les fichiers peuvent être insérés sous forme d'images directement sous **Word** (2 ou 6).

Plus généralement, un fichier-image contient des caractères; il est constitué d'un en-tête (appelé en jargon un bandeau) et du codage de l'image proprement dit.

Dans le cas des fichiers *.TIF, l'en-tête se compose de la variable TIF, un vecteur de 198 caractères dont les indices successifs dans ⎕AV le vecteur atomique en origine 0 sont par exemple les suivants :

```
73 73 42 0 24  0 0 0 44 1   0  0 1   0  0 0 44  1 0   0 1 0   0  0 1 4   0 254 0 4 0 1 0 0
 0  0 0 0  0  0 0 1 3  0 1   0  0 0 0 200  0 0  0 1 1   3 0 1   0  0 0 233   0 0 0 2 1 3 0
 1  0 0 0  1  0 0 0  3 1   3  0 1   0  0 0 0  1 0 0   0 6 1   3 0 1   0   0 0 0 0 0 0 7
 1  3 0 1  0  0 0 1  0 0   0 1 7 1   4 0 1  0  0 0 1 9 8 0 0   0 2 1   1   3   0 1 0 0 0 1 0
 0  0 2 2  1  4 0 1  0 0 0 2 3 3  0 0   0 2 3 1  4 0 1   0 0 0 1 7 0 2 3   0   0 2 6 1 5 0 1 0 0
 0  8 0 0  0 2 7 1  5 0 1   0  0 0 0 16  0 0  0 4 0 1   3 0 1   0 0 0   2   0 0 0 0 0 0 0
```

Dans le cas des fichiers *.BMP, l'en-tête se compose de la variable BMP, un vecteur de 118 caractères dont les indices successifs dans ⎕AV en origine 0 sont les suivants :

```
66 77 118 150   0 0   0   0 0 0 118   0   0 0  40   0   0 0  64   1   0 0 240   0
 0  0   1   0   4 0   0   0 0 0   0   0   0 0   0   0   0 0   0   0   0 0   0   0
 0  0   0   0   0 0   0   0 0 0   0   0 191 0   0 191   0 0   0 191 191 0 191   0
 0  0 191   0 191 0 191 191 0 0 192 192 192 0 128 128 128 0   0   0 255 0   0 255
```

```
 0   0    0 255 255 0 255    0 0 0 255    0 255 0 255 255    0 0 255 255 255
```

Si maintenant on désire transformer une matrice binaire en fichier-image, il s'agit de compléter de manière adéquate l'une ou l'autre de ces variables qui servira d'en-tête.

Programmes généraux d'écriture et de lecture de fichiers natifs

(pour APL*PLUS III; également valables pour APL*PLUS PC et APL*PLUS II à condition que la taille de fichier n'excède pas la longueur d'un segment de mémoire).

```
    ∇ R←a dansfic f;K;⎕ELX;e
[1]   ⍎'R←¯1◊e←''a←a rempltxt qS,qR,qS,qR,qL''◊⎕ELX←''→1+⎕LC''◊ f⎕ntie R◊ f⎕neras
e R'
[2]   ⍎'⎕ELX←''⎕DM''◊ f⎕ncreate R◊ K←~qL∈a◊ K←K∧qR∈a◊ K←K∧1=⍴⍴a◊ ⍎K/e'
[3]   ⍎'a⎕nappend R◊ ⎕nuntie R◊R←0 0⍴a'
    ∇
```

a est un vecteur de caractères et f un nom de fichier natif [1].

rempltxt est la fonction txtrepl de la zone ASMFNS, avec des arguments commutés (le vecteur dans lequel s'effectuent les remplacements à gauche).

Les variables qS, qR et qL sont des scalaires-caractères, respectivement la barre oblique (en anglais « Slash »), le Retour du chariot (en APL*PLUS ŒTCNL : « Terminal Character New Line »), et le saut de Ligne (en APL*PLUS Œ TCLF: « Terminal Character Line Feed »).

La ligne [1] a pour but **a)** de remplacer les retours du chariot simples (suffisants pour passer également à la ligne en APL) par des retours du chariot et des passages à la ligne (obligatoires pour les fichiers natifs DOS si vous voulez pouvoir les lire, par exemple à l'aide d'une simple instruction **type** sous DOS), **b)** de détruire un éventuel fichier portant déjà le même nom que celui contenu dans l'argument f.

Le cas d'un fichier en tableau n'est pas traité ici, car les logiciels décrits ci-après fabriquent des vecteurs de caractères. Le résultat R est normalement un objet vide; si on obtient ¯1 c'est qu'il s'est produit quelque chose d'anormal.

```
      ∇ R←a depuisfic f;⎕elx
[1]   ⎕nuntie R←¯1◊⎕elx←'⎕dm'◊ f⎕ntie R◊ f←⎕nread R,82,⎕nsize R◊ ⎕nuntie R◊R←f
[2]   a←0=⎕NC'a'◊ ⍎a/'R←R~qL'◊ a←-'→'=¯1↑R ◊ R←a↓R
     ∇
```

Comme pour l'écriture de fichier, le nom f est le nom de fichier; a est un argument facultatif, dont la présence entraîne l'élimination des sauts de ligne (en APL, comme déjà mentionné, le retour du chariot assure en plus la passage à la ligne suivante). L'argument de sortie R contient alors le contenu du fichier, un vecteur de caractères; en

outre, si le dernier caractère du fichier est un caractère EOF « End Of File », ce qui correspond, en APL, au symbole de branchement « … », ce caractère, parfois gênant, se trouve ôté du vecteur-résultat.

Nous sommes maintenant en mesure de fabriquer des fichiers *.TIF ou *.BMP.

Fabrication de fichiers binaires (images) proprement dits

```
    ∇ R←A FABTIF M;C;D;F;J;K;L;S;⎕IO
[1]   ⎕IO←0⋄ 'A←0'do 0=⎕NC'A'⋄ L←11=⎕DR M⋄ L←L∧2=ρD←ρM⋄ F←'C:\APL.TIF'⋄ J←1↓K←3ρ2
56
[2]   ⍎L/'''D←ρM←M,0''do A' ⍝ ajoute une lignes vide (A) en bas
[3]   ⍎L/'''D[1]←C⋄M←D↑M''do D[1]≠C←8×⌈.125×S←D[1]⋄ D←ρM←82⎕DR M⋄M←,M⋄ R←⍕ρM←TIF,
M'
[4]   ⍎L/'M[154+⍳3]←qAF K⊤×/D⋄ M[47 46]←qAF J⊤S⋄ M[143 142]←M[59 58]←qAF J⊤D[0]'
[5]   ⍎L/'M dansfic F⋄ R←R,F'
     ∇
```

Cette fonction FABTIF fabrique un fichier *.TIF à partir d'une matrice binaire M (non nécessairement carrée). Le nom du fichier s'appelle ici 'C:\APL.TIF' mais on peut le changer à volonté.

Comme d'habitude, la sous-fonction do exécute l'expression APL contenue dans son argument gauche, si la condition contenue dans l'argument droit est vraie.

Si l'argument facultatif A existe, une ligne de zéros sera ajoutée en bas de la matrice M : on conseille de toujours utiliser cette option lorsque le nombre de lignes de M est impair.

La fonction qAF équivaut à la fonction ⎕AF d'APL2. Elle établit une correspondance biunivoque entre les entiers de 0 à 255 et les caractères de ⎕AV .

La mise en fichier d'une matrice binaire est immédiate.

```
  ∇ R←FABBMP M;C;D;E;F;J;K;L;S;⎕IO
[1] ⎕IO←0⋄ L←11=⎕DR M⋄ L←L∧2=ρD←ρM ⋄ F←' C:\APL.BMP' ⋄ J←2↓K←4ρ256⋄ M←⊖M⋄ M←~M
[2] ⍎L/'E←BMP⋄E[18+⍳8]←qAF,⌽⌽⍎K⊤D⋄D[1]←2×S←4×⌈.125×D[1]⋄M←D↑15×M⋄D←×/2 2ρD[0],S,2
 1'
[3] ⍎L/'E[2 3]←qAF⍎,J⊤ρM←qAF 16⊥⍉DρM⋄M←E,M⋄ ''R←⍕ρM⋄R←R,F''do~¯1∈M dansfic F'
   ∇
```

Dans la fonction FABBMP , le nom du fichier F peut se changer également.

Dans les deux fonctions, la mise en fichier n'a lieu que si M est bien une matrice **binaire**, c'est-à-dire un objet de type 11 (« Data Representation »). Pour faire fonctionner ce logiciel par exemple en APL*PLUS PC, implantation qui ne possède pas de codage binaire, il convient de remplacer cette condition par une vérification que tous les items de M font bien partie de l'ensemble 0 1.

Nous laisserons le lecteur écrire, à titre d'exercice, les fonctions inverses, qui, à partir d'un fichier *.TIF ou *.BMP, reconstituent la matrice M.

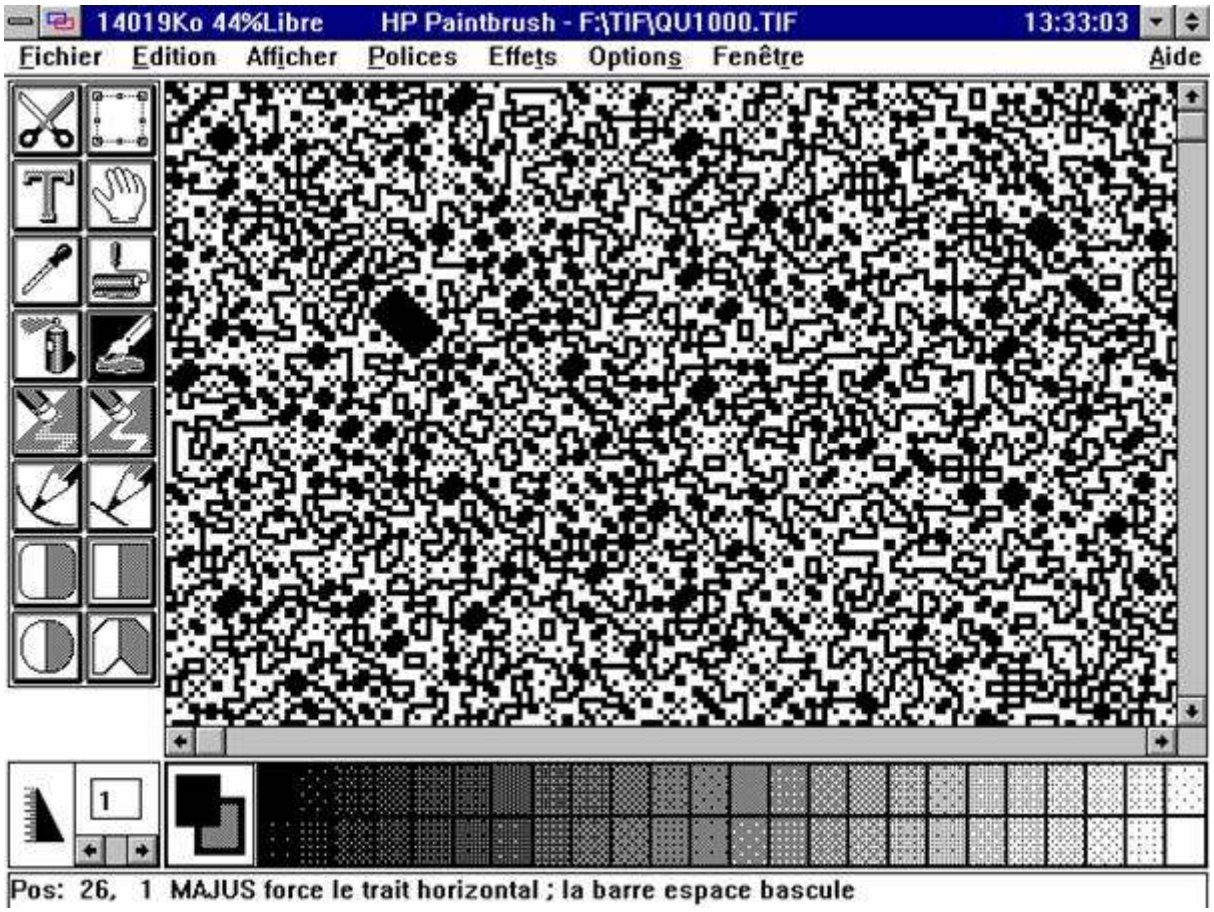
Il est tout à fait possible de créer des fichiers *.BMP en couleurs. Dans les listages ci-dessus, on n'a choisi que deux couleurs, le noir et le blanc.

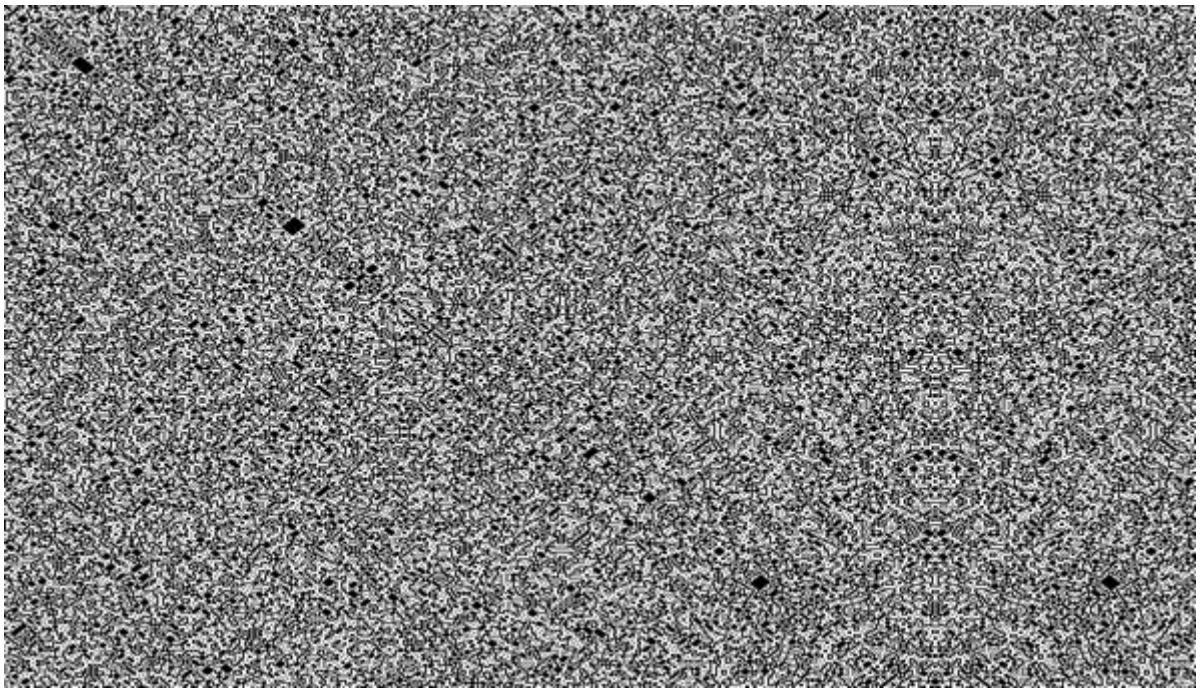
Les fichiers *.TIF occupent moins de place sur disque que les fichiers *.BMP (à cause, justement, du codage possible de la couleur dans ces derniers).

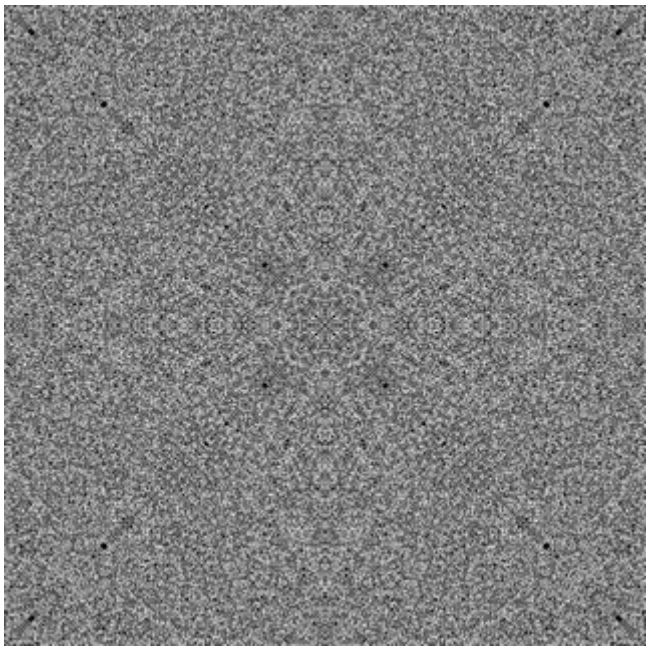
A titre d'application maintenant, voici le résultat du QUINTO 1000, une matrice d'un **million de bits** , insérée sous « WORD » en tant qu'image, à partir d'un fichier *.TIF. La vue suivante est une recopie d'écran du fichier tel qu'on peut la voir sous le logiciel « HP-Paintbrush » avec une magnification (zoom) d'un facteur 3; seule la partie située en haut et à gauche (c'est-à-dire les premières lignes et les premières colonnes) de la matrice-résultat se trouve affichée. La totalité du fichier *.TIF occupe en mémoire une place de 125198 octets exactement. La solution du QUINTO 1000 est unique et présente, de ce fait, la symétrie du carré. (Pour l'obtenir, on a utilisé la méthode de Langlet-Mironov , amplement décrite dans les numéros précédents de ce journal; nous sommes bien loin des résultats que permettait d'obtenir, selon M. Dumontier, le domino d'APL, lequel déclare déjà forfait pour le pauvre QUINTO 18 !).

La seconde figure ne présente elle aussi qu'une partie du résultat, sans « zoom », telle qu'on peut l'afficher en plein écran également sous « HP-Paintbrush ».

N.B. Les figures originales sont en taille double : ne pas oublier que les pages des Nouvelles d'APL sont réduites d'un facteur linéaire de 2.


14019Ko 44%Libre
HP Paintbrush - F:\TIF\QU1000.TIF
13:33:03
Fichier Edition Afficher Polices Effets Options Fenêtre Aide
1
Pos: 26, 1 MAJUS force le trait horizontal ; la barre espace bascule





Voici ce que donne l'image complète d'un million de points à l'échelle 1, avec une résolution de 600 points au pouce (la nuit, tous les chats sont gris), après importation sous Word6 du fichier *.TIF correspondant.

Une page complète pourrait théoriquement contenir 6 millions de points; encore faut-il pouvoir l'imprimer, (votre imprimante-laser doit disposer de mémoire supplémentaire).

Attention ! En ce qui concerne la mise en fichier *.BMP , ne pas oublier que le type entier occupe, en APL*PLUS III, 4 octets par pixel (si le blanc est codé 0, le noir sera codé 15 et non pas 1), de sorte qu'il vous faudra une mémoire centrale colossale pour transférer une simple image d'un million de points !

Passons alors à l'un des « top models » les plus utilisés en traitement d'image. Il s'agit de la jolie Léna, célèbre de par le monde, car elle fait les gorges chaudes de nombreux articles consacrés à la transformée de Fourier et à la compression d'images dans les milieux spécialisés [4].

La figure suivante est obtenue à partir d'une matrice LENA de dimension 512 512 , passée au scanneur à partie d'une photographie, transférée par la fonction FABTIF en fichier *.TIF, puis affichée en plein écran par le logiciel « HP-Paintbrush » et transférée par le presse-papiers sous WORD6; ceci permet d'obtenir sans autre forme de procès, une image environ 4 fois plus grande linéairement que si on l'avait importée sous Word directement depuis le fichier *.TIF. Le fichier possède une taille de 32966 octets. On peut donc mettre sur disquette HD près de 50 fichiers de ce type, de quoi conserver le trombinoscope-palmarès des adhérents d'**AFAPL** ayant réglé leur cotisation 1996 avant le printemps.

; ; 

Voici, importés directement sous Word sous forme d'image, deux agrandissements différents d'un fichier *.TIF obtenu à l'aide de l'instruction :

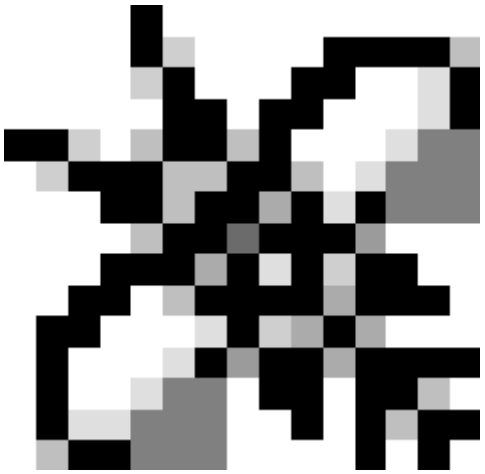
FABTIF 2/2/LENA

Ce qui fonctionne sur de gros fichiers fonctionne aussi, évidemment, sur de petits. Voici ce que donne une insertion de fichier sous « Word6 » :

Une matrice 15⊖15 est affichable (fichier *.BMP) comme:

Le fichier *.TIF correspondant s’affiche comme:

Ensuite, on peut dupliquer et agrandir en cliquant deux fois pour appeler l’éditeur d’images de Word:



Bien que la qualité d’impression des Nouvelles d’APL n’arrive pas encore à la cheville de celle de la formule du nouvel **APL-Quote-Quad** (2000 pixels au pouce), on voit très bien sur ces reproductions qu’il

n’est absolument pas nécessaire de considérer plus de deux niveaux d’intensité pour décrire au mieux toutes les nuances de gris d’une image (photographie) en noir et blanc. Les pixels (bits) eux-mêmes de l’image définissent leur propre amplitude ou intensité, puis que le carré de 0 vaut 0 et que le carré de 1 vaut 1.

Pourquoi donc se complique-t-on la vie en traitement d’images classique, alors que le binaire pur permet d’effectuer la totalité des manipulations d’image directement en algèbre logique, sans aucun risque d’erreur ni de troncature d’information, et en outre, par des transformations le plus souvent **linéaires** ? La réponse fait intervenir la *force de l’habitude* [5] ; les physiciens et autres spécialistes du traitement d’images ont toujours procédé comme cela; en fait, il s’agit d’une **erreur historique**, due à la méconnaissance de l’algèbre vectorielle et matricielle modulo 2, telle qu’on peut la pratiquer en APL avec un peu de jugeotte et une banque d’idiomes du plus bel effet, esthétique comme pratique.

---

[1] Il n’est bien sûr pas nécessaire d’inclure chaque ligne dans un vecteur de caractères exécutable : –' ' . Si nous le faisons, c’est parce qu’il s’agit d’un moyen de supprimer les espaces blancs inutiles, et aussi parce que dans certaines implantations d’APL (notamment APL.68000), les fonctions occupent beaucoup moins de place en mémoire.

[2] Le **Quinto 1000** est tout à fait soluble en APL*PLUS III. Sur un 486 à 33 MHz, il faut environ 80 minutes de patience !

[3] Rappelons que la méthode de Langlet a été considérablement accélérée par le jeune Ilya Mironov , et que, sans ce dernier, la solution du Quinto 1000 exigerait plusieurs jours d’ordinateur et une mémoire immense !

[4] ce dont nous reparlerons à l’occasion d’un prochain article.

[5] Cette force, inconnue des manuels de Physique, existe pourtant bel et bien. La plupart des physiciens utilisent des méthodes mathématiques mises au point essentiellement aux XVIIe et XVIIIe siècles, et encore enseignées comme le nec plus ultra dans les Grandes Ecoles et les Facultés, alors qu’il conviendrait d’innover quelque peu, c’est-à-dire de bâtir des algorithmes mettant en jeu le langage propre de l’ordinateur, qui ne sait parler QUE le binaire, langue dans laquelle il excelle savamment.