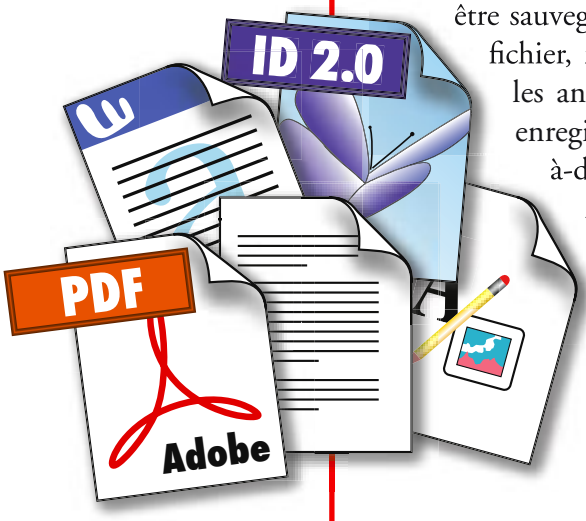
LES FICHIERS INFORMATIQUES

Introduction



L'ordinateur traite des informations de tous genres : textes, photos, illustrations, calculs scientifiques, images de synthèse, etc. Toutes ces données doivent être sauvegardées et surtout structurées. Lorsqu'une application ouvre un fichier, il faut qu'elle sache à quoi elle a à faire et comment elle doit les analyser. Chaque application lors de la sauvegarde d'un fichier enregistre les données du document dans un ordre bien précis, c'est-à-dire un format de fichier. Ainsi, lors de l'ouverture du fichier, le logiciel, auquel il est destiné, peut lire les données et les traiter convenablement. Chaque fichier est accompagné d'un suffixe (.txt, .psd, .doc, etc.), ce qui permet une reconnaissance de l'appartenance du fichier à une application avant même la lecture de ce dernier. Le système d'exploitation associe également une icône aux différents types de fichier pour une identification visuelle et rapide par l'utilisateur.

En réalité, un fichier enregistré sur un support informatique, quel qu'il soit, correspond à une suite d'états d'information/non information. Par exemple, un lecteur de CD-Rom (ou de DVD), par le biais de son laser, va scruter la surface d'une galette en plastique (le CD-Rom lui-même). Cette surface possède des trous. S'il y a présence d'un trou, il y a information. S'il n'y a pas de trou, il n'y a pas d'information. Electriquement, ces présences ou absences de trous vont activer des circuits électroniques pour organiser et structurer les informations recueillies (fig. 1). En réalité la détection du « 1 » se fait lorsqu'il y a passage d'un creux à une surface plane ou inversement. Mais cela ne gêne en rien la démonstration. Comme vu dans le chapitre précédent, pour une meilleure compréhension par l'être humain, on codifie les informations en bits puis en octets. Ensuite, la gestion de ces données sera réalisée suivant des règles « inventées » par les informaticiens, l'ordinateur n'étant qu'une simple machine électrique obéissant à des ordres pré-programmés.

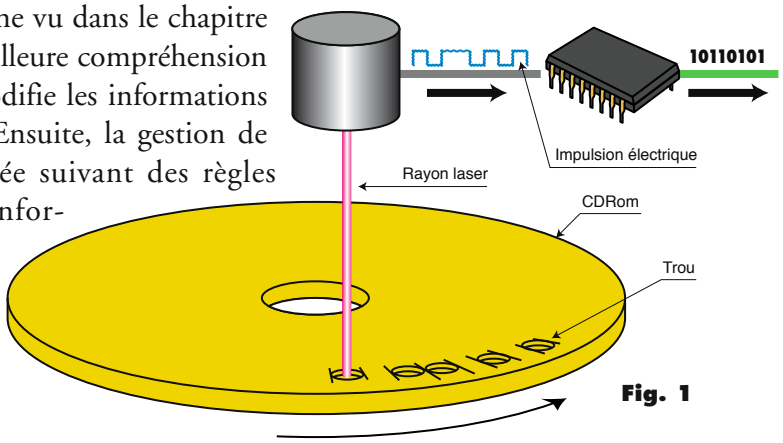


Fig. 1

Il existe 4 grands types de formats de fichiers : le texte, le bitmap, le vectoriel et les polices de caractères.

Fichiers texte

C'est le fichier de base par excellence. C'est même le tout premier fichier qui ait été inventé en informatique. Très vite, les premiers informaticiens ont eu besoin de dialoguer avec les circuits électroniques de leurs machines. Il leur fallait envoyer et visualiser des informations de façon naturelle. Le plus facile pour communiquer étant l'écrit, le clavier pour saisir des lettres et le moniteur (écran) pour visualiser des messages naquirent. Bien sûr, il fallut codifier ce texte au format informatique et, pour que tous les informaticiens du monde puissent s'entendre, le code ASCII (American Standard Code for Information Interchange) fut inventé. A chaque caractère, signe de ponctuation, action sur le clavier fut associée une valeur numérique (fig. 2), certaines étant réservées à la commande de périphériques ; imprimantes, modems. Le tableau T1 représente cette codification.

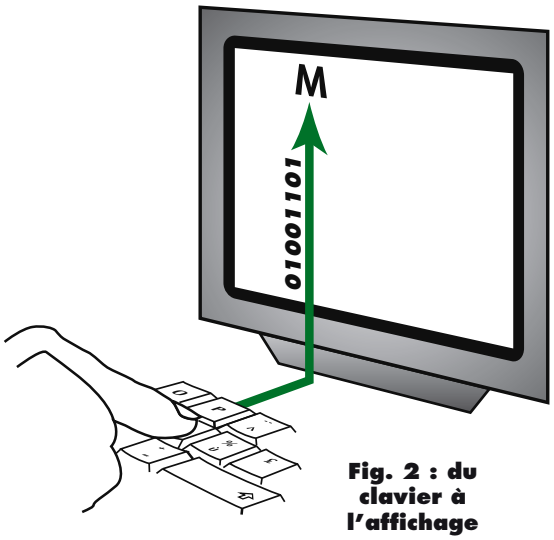


Fig. 2 : du clavier à l'affichage

Un fichier texte regroupe ces codes et les restitue après traitement sous forme de lettres, de chiffres, de signes de ponctuation, d'ordres comme la tabulation, le retour chariot, etc…

Il est à noter que le monde PC et le monde MAC n'utilisent pas exactement la même codification des caractères, notamment pour les caractères accentués (numérotés de 128 à 256) et qu'il est nécessaire de faire une conversion (parfois automatique).

0	Nul	16	Data Link Escape	32	Espace	48	0	64	@	80	P	96	`	112	p
1	Début d'entête	17	Contrôle périph.	33	!	49	1	65	A	81	Q	97	a	113	q
2	Début de texte	18	Contrôle périph.	34	"	50	2	66	B	82	R	98	b	114	r
3	Fin de texte	19	Contrôle périph.	35	#	51	3	67	C	83	S	99	c	115	s
4	Fin de transmission	20	Contrôle périph.	36	$	52	4	68	D	84	T	100	d	116	t
5	Demande	21	Acc. de réception	37	%	53	5	69	E	85	U	101	e	117	u
6	Accusé de réception	22	Synchronisation	38	&	54	6	70	F	86	V	102	f	118	v
7	Beep	23	Fin de transmission	39	'	55	7	71	G	87	W	103	g	119	w
8	Backspace	24	Annulation	40	(	56	8	72	H	88	X	104	h	120	x
9	Tab. horizontale	25	Fin de support	41	)	57	9	73	I	89	Y	105	i	121	y
10	Saut de ligne	26	Substitution	42	*	58	:	74	J	90	Z	106	j	122	z
11	Tab. verticale	27	Escape	43	+	59	;	75	K	91	[	107	k	123	{
12	Saut de page	28	Séparateur de fichier	44	,	60	<	76	L	92	\	108	l	124	|
13	Retour chariot	29	Séparateur de groupe	45	-	61	=	77	M	93	]	109	m	125	}
14	Fin d'extension	30	Séparateur d'enreg.	46	.	62	>	78	N	94	^	110	n	126	~
15	Début d'extension	31	Séparateur d'unité	47	/	63	?	79	O	95	_	111	o	127	Effacement

Code ASCII Caractère ou action correspondant

Tableau T1 : le code ASCII

Fichiers bitmap

Le mot bitmap veut dire **carte de bits** ou matrice de bits. L'écran est divisé suivant une grille. Chaque case de cette grille est associée à une information binaire et, en fonction de cette information, l'ordinateur, via sa carte vidéo, affiche un **point** sur l'écran, **appelé pixel** (fig. 3). Par extension, **toute image composée de pixels est appelée bitmap**. C'est le cas des images réalisées avec Photoshop™ ainsi que toutes celles possédant, par exemple, le suffixe jpg, bmp, png, gif, tif...

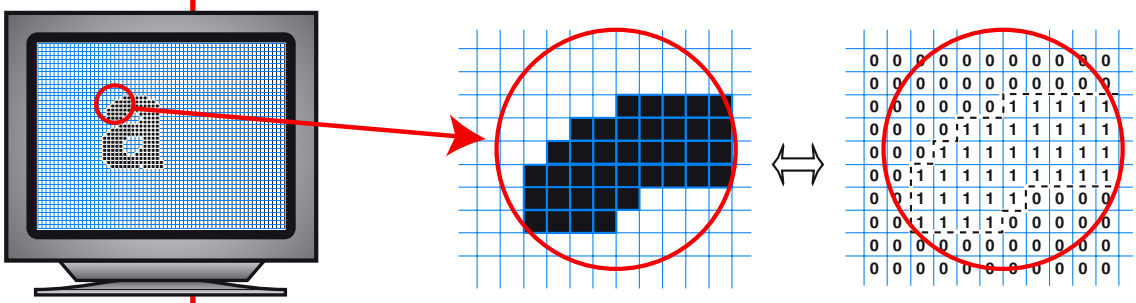


Fig. 3

Codage couleur des pixels

Si à l'origine **chaque pixel** était **codé** sur un bit, les besoins d'affichage en couleur ont amené les informaticiens à le coder **sur un ou plusieurs octets**.

Rappelons qu'un octet se compose de huit bits permettant de coder 256 valeurs différentes allant de 0 à 255. Si l'on associe ces valeurs à des couleurs prédéfinies dans une palette (pour le web par exemple) on peut afficher un pixel en couleur (fig. 4a) ou bien en gris si la palette correspondante se compose de différents gris ou niveaux de gris (fig. 4b).

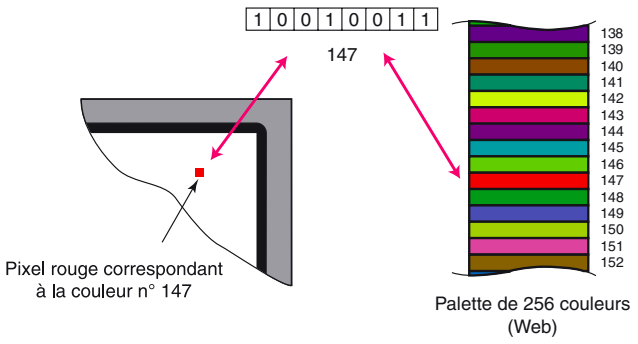


Fig. 4a

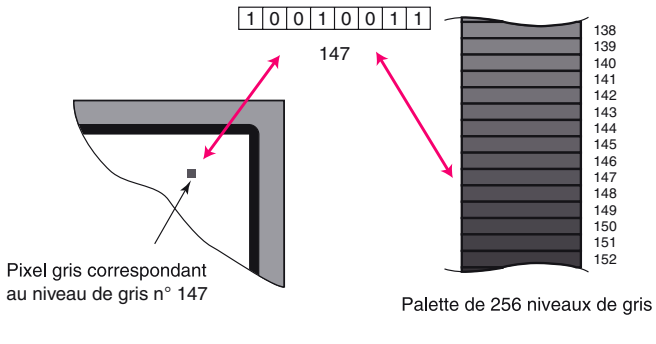


Fig. 4b

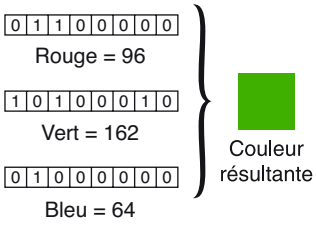


Fig. 5a

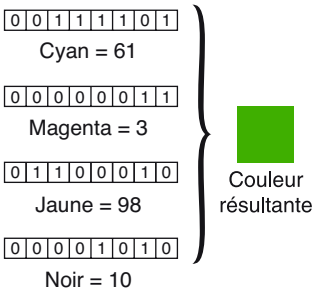


Fig. 5B

Un codage sur 2 octets permet d'accéder à près de 65000 couleurs (affichage en milliers de couleurs). Un codage sur 3 octets, soit 24 bits, autorise une palette de près de 16 millions de couleurs par le biais des couleurs rouge, verte et bleue à la base même de la colorimétrie (fig. 5a). Chacune de ces couleurs sera codée sur un octet, un octet pour le rouge, un octet pour le vert, un octet pour le bleu (RVB ou RGB). Un codage sur quatre octets permet de travailler en CMJN chaque couleur étant codée, également, sur un octet (fig. 5b). Cet affichage des pixels en fonction du nombre de bits est appelé « profondeur d'écran » (fig. 6).

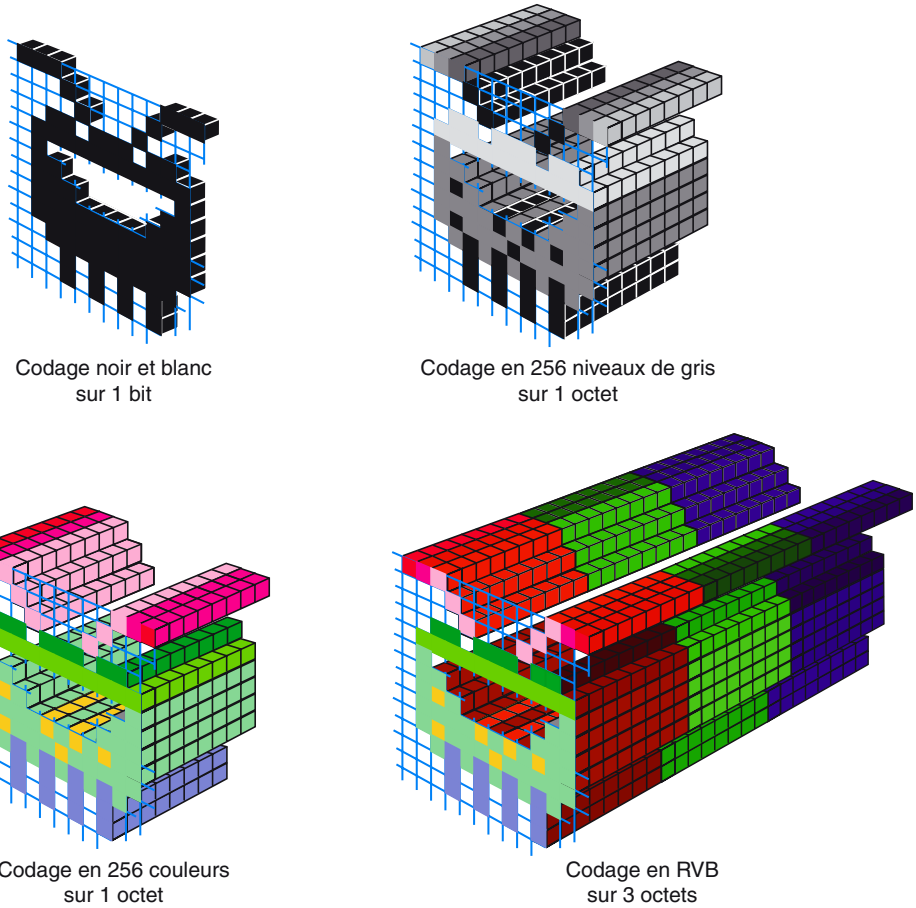


Fig. 6 : pronfondeurs d'écran

Résolution et définition d'un bitmap

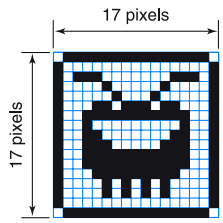
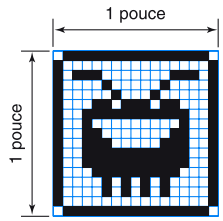


Fig. 7



Résolution de 17 dpi

Fig. 8

Ces deux termes sont parfois confondus bien qu'ils aient des sens très différents.

La définition exprime les dimensions d'une image, c'est-à-dire sa largeur et sa hauteur définies en pixels (fig. 7).

La résolution indique, quant à elle, **le nombre de pixels** existant **sur** une distance de **1 pouce** de cette image (fig. 8). Elle est exprimée en dot per inch en anglais (dpi) ou point par pouce en français (ppp). La taille d'un bitmap peut varier en fonction du périphérique auquel il est destiné, 72 dpi pour un écran, 600 dpi pour une imprimante laser, 1270 ou 2540 dpi pour une flasheuse. La figure 9 montre le même bitmap imprimé dans deux résolutions différentes. Sur le dessin de droite, il est deux fois plus petit puisque chaque point le constituant est deux fois plus petit que sur l'illustration de gauche.

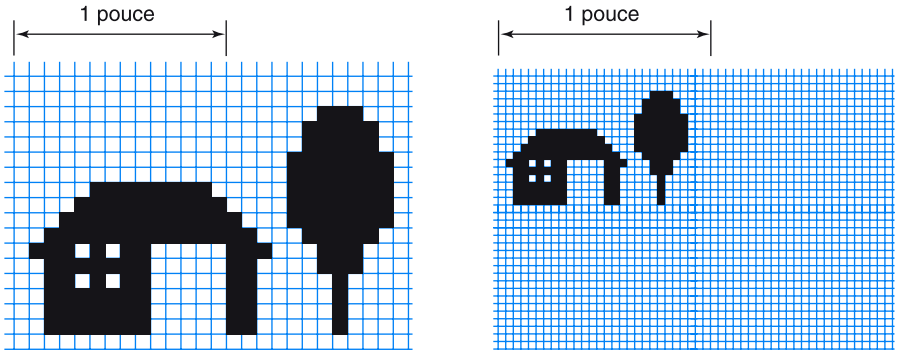


Fig. 9

Calcul du poids d'un bitmap

Le poids d'une image, c'est la quantité de mémoire utilisée par cette image sur un support informatique, donc le nombre d'octets constituant cette image exprimé en ko, Mo ou Go.

Il est important d'en avoir une idée précise pour des raisons de productivité évidente : les disques durs seront moins encombrés, les serveurs moins surchargés, les transferts sur réseau et les manipulations à l'écran plus rapides, les temps de flashage plus courts...

Même si dans la vie professionnelle quotidienne on calcule très rarement le poids d'un fichier bitmap, savoir déterminer le poids d'une image permet de mieux appréhender la structure d'un fichier bitmap.

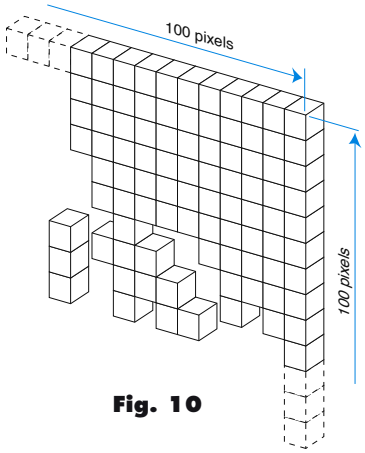


Fig. 10

Premier exemple : considérons une image de 100 pixels par 100 pixels codée sur 1 bit (fig. 10). Son poids est simple à calculer.

Nous avons, pour constituer ce bitmap : 100 x 100 = 10 000 pixels.

Chaque pixel est codé sur 1 bit, ce qui fait : 10 000 x 1 = 10 000 bits.

1 octet fait 8 bits. Il suffit de diviser par 8 soit : 10 000 ÷ 8 = 1250 octets (ou 1,22 ko).

Deuxième exemple : soit une image de 25,4 cm par 12,7 cm, codée en niveaux de gris, donc sur 1 octet, avec une résolution de 100 dpi (100 pixels par pouce).

Comme la résolution est exprimée en pouce (dpi), il faut ramener les dimensions de l'image en pouce. Il suffit de diviser par 2,54 (1 pouce = 2,54 cm), soit dans notre exemple :

25,4 ÷ 2,54 = 10 pouces pour la largeur et 12,7 ÷ 2,54 = 5 pouces pour la hauteur.

On sait qu'il y a 100 pixels sur un pouce (résolution = 100 dpi), nous avons :

10 x 100 = 1 000 pixels sur la largeur et 5 x 100 = 500 pixels sur la hauteur.

Soit un total de 1 000 x 500 = 500 000 pixels pour constituer l'image.

Chaque pixel étant codé sur 1 octet cela fait : 500 000 x 1 = 500 000 octets,

soit : 500 000 ÷ 1024 = 488,28 ko ou 488,28 ÷ 1024 = 0,47 Mo.

Rappelons qu'**un kilo-octet vaut 1 024 octets** et non pas 1 000, tout comme 1 méga-octet vaut 1 024 ko, etc.

Si le bitmap avait été codé sur 3 octets (RVB), il aurait suffit de multiplier par 3 le nombre de pixels obtenu ci-avant, soit :

500 000 x 3 = 1 500 000 octets ou 1 500 000 ÷ 1024 = 1464,84 ko ou 1464,84 ÷ 1024 = 1,43 Mo. Pour une image en CMJN nous aurions multiplié par 4.

On peut en déduire une formule :

$$\text{Poids} = \frac{\text{largeur} \times \text{hauteur} \times \text{résolution}^2 \times \text{profondeur d'écran}}{\text{1000 ou 1024}}$$

En pouce

En dpi

En nb. d'octets :

0,125 pour du bitmap
1 pour du niveau de gris
1 pour 256 couleurs
2 pour 65000 couleurs
3 pour du RVB
4 pour du CMJN

En kilo-octet, Mo, Go, ... (si l'on divise par 1000)
En kibioctets, Mio, Gio, ... (si l'on divise par 1024)

Diviser par 1000 ou 1024 autant de fois que nécessaire pour obtenir au maximun 3 chiffres significatifs avant la virgule
(ex : 12,39 Mo et non pas 12395,2 ko)

Un bitmap supporte très mal les changements d'échelle. Si un bitmap fait 4 pixels par 4 pixels, le doubler de taille augmente d'autant son nombre de pixels. Il va donc manquer des pixels pour le définir (fig. 11a). La première solution consiste donc à doubler la taille des pixels, ce qui rend le bitmap « grossier » (fig. 11b). On appelle ça la « pixellisation ». L'idéal serait de garder la même grosseur de pixel et d'inventer des pixels intermédiaires (fig. 11c). Il s'agit du rééchantillonage. Plusieurs algorithmes sont disponibles et suivant leur capacité, ils prennent plus ou moins en compte leur environnement. Cependant, le rééchantillonnage a ses limites : toutes les images ne le supportent pas et peuvent se retrouver floues (fig. 12).

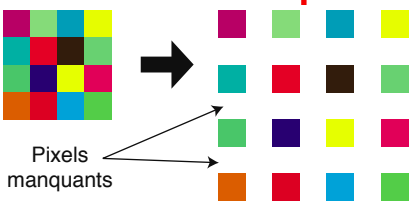


Fig. 11a

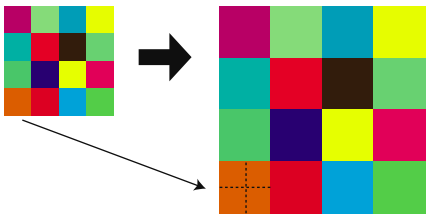


Fig. 11b

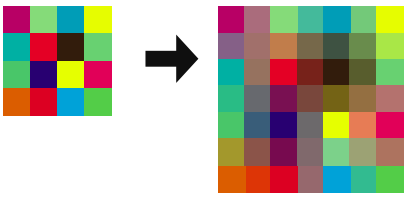


Fig. 11c

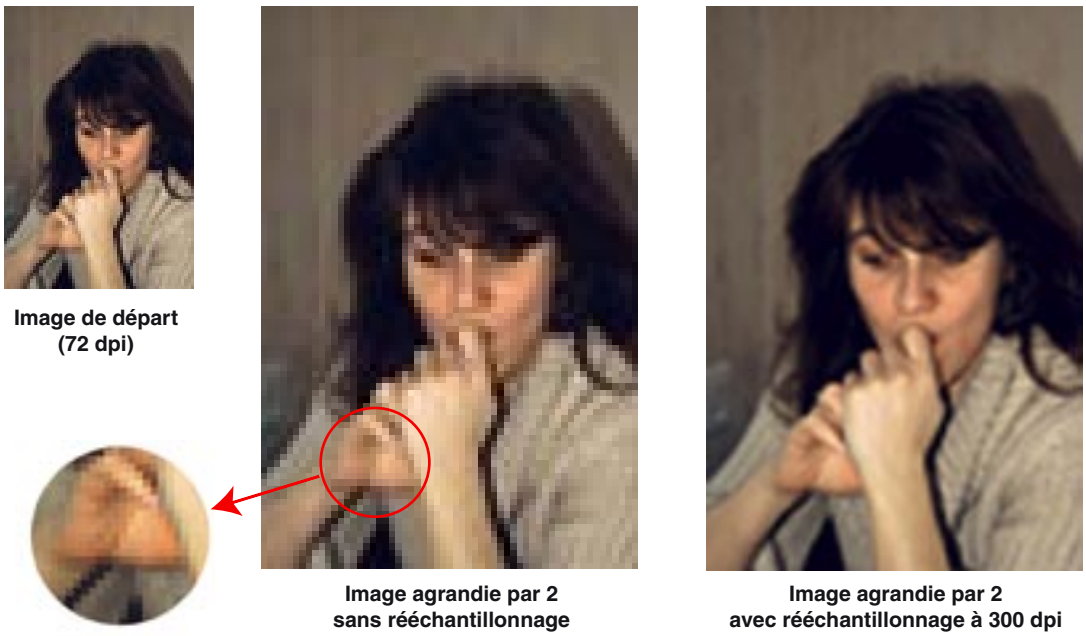


Image de départ (72 dpi)

Image agrandie par 2 sans rééchantillonnage d'où une pixellisation

Image agrandie par 2 avec rééchantillonnage à 300 dpi

Fig. 12

Fichiers vectoriels

Les fichiers bitmap sont gourmands en mémoire. La figure 13 représente une image en bitmap. Il s'agit d'un simple trait. Cependant, pour le définir, il faut l'inclure dans un carré de 100 pixels par 100 pixels, le trait étant représenté par des points noirs, tous les autres pixels par des points blancs. Ces points blancs existent bel et bien et sont codés de la même manière que les points noirs (sur 1 bit, 1, 2, 3 ou 4 octets) et donc utilisent autant de mémoire qu'eux. Pour définir ce simple trait, il faut coder 100 x 100 = 10 000 pixels et donc au moins autant d'informations.

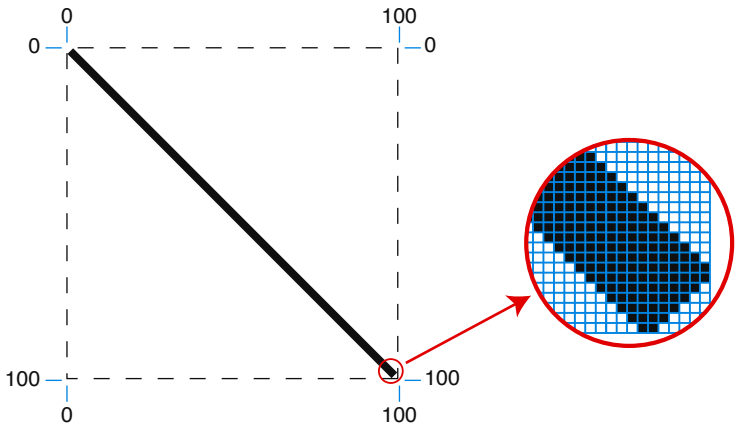


Fig. 13 : bitmap de 100 pixels par 100 pixels

L'idée du vectoriel est de **diminuer le nombre d'informations** pour définir un graphisme en indiquant uniquement des **coordonnées** et le **type de graphisme**. Par exemple, en vectoriel, le trait de la figure 13 serait défini par ses coordonnées (0,0), (100,100) et l'instruction « ligne ». Cela fait 5 informations contre 10 000 dans le cas d'un bitmap. Les **fichiers** s'en trouvent nettement **réduits** et il suffit d'avoir un **interpréteur**, c'est-à-dire un logiciel capable d'analyser les informations vectorielles et de les transformer en informations bitmap pour l'affichage sur l'écran (fig. 14). Rappelons que l'affichage à l'écran est forcément bitmap puisque l'écran est une matrice de pixels.

Travailler en vectoriel présente le très gros avantage de s'**adapter à la résolution des périphériques de sortie** (écran, imprimantes, flasheuse, CTP). La figure 9 de la page 27 montre que la taille d'affichage (ou d'impression) varie en fonction de la résolution, c'est-à-dire en fonction de la grosseur des pixels que restitue un périphérique. Cela veut dire que les documents réalisés sur ordinateur auraient des formats différents selon qu'ils seraient imprimés sur imprimante jet d'encre à 600 dpi ou sur une flasheuse à 2 500 dpi par exemple.

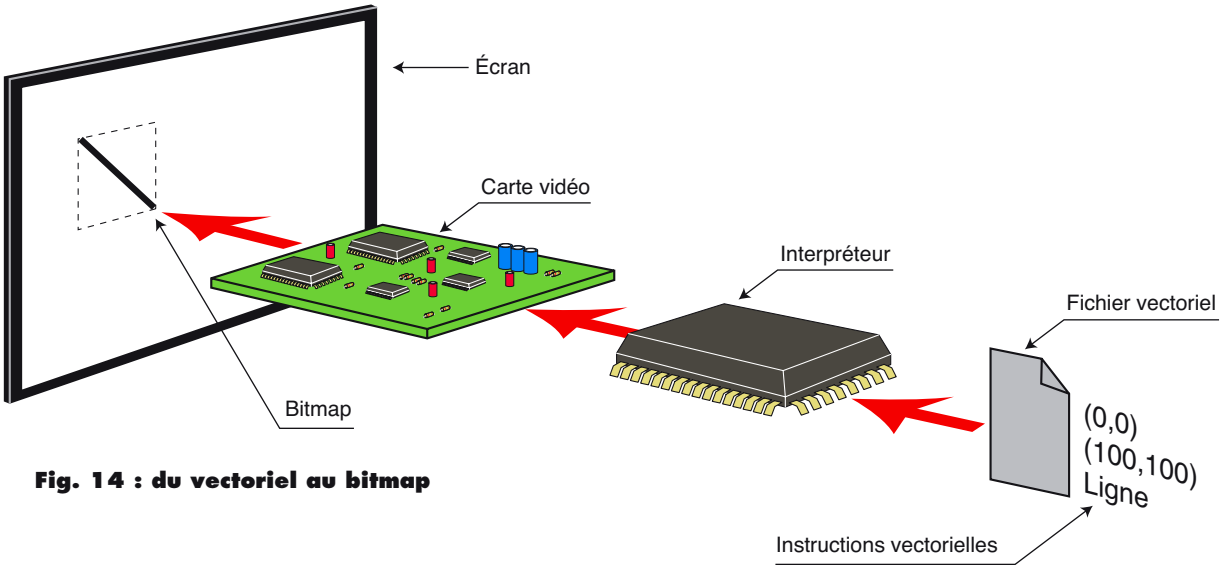


Fig. 14 : du vectoriel au bitmap

Les instructions vectorielles manipulent une page globalement et place les éléments constituant cette page dans un système de coordonnées. L'exemple de la figure 15 montre qu'une même ligne sera imprimée dans les même dimensions et positionnée au même endroit sur une feuille de papier quelle que soit la résolution de l'imprimante. L'interpréteur calcule le nombre de pixels correspondant à une dimension. Par exemple, la coordonnée « x » du point « P » sera positionnée sur le pixel n° 590 pour une résolution de 300 dpi et sur le pixel n° 1 180 pour une résolution de 600 dpi mais sera située à 50 mm du bord de la feuille dans les deux cas. Cette nature mathématique autorise toutes les manipulations, notamment les agrandissements. Un simple coefficient permet de recalculer les coordonnées d'un objet graphique et de redéfinir les pixels qui le composent. A contrario, un bitmap ne subit pas d'agrandissement sans pixellisation.

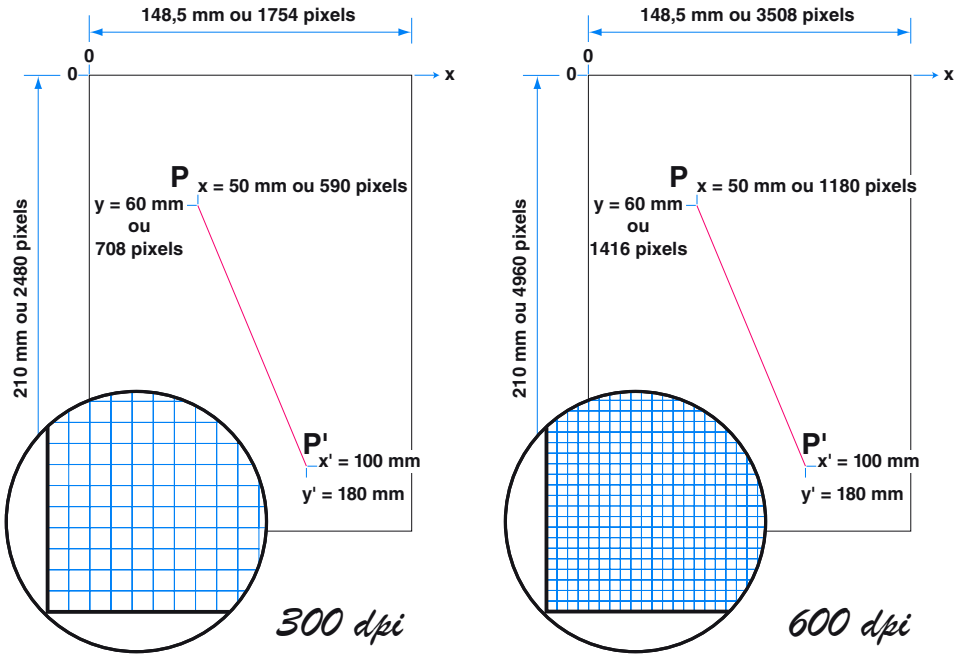


Fig. 15 : coordonnées globales et coordonnées en pixels

Le PostScript

Le PostScript est devenu un standard dans les industries graphiques et peut s'importer dans grand nombre de logiciels de PAO. Il a été inventé par la société américaine Adobe Système Inc, et c'est un langage de programmation au même titre que d'autres langages informatiques tels que le Basic, le C ou le Pascal. Son rôle est, avant tout, de **contrôler l'ensemble d'une page imprimée** en terme de qualité **indépendamment de la résolution de sortie** des périphériques d'impression. C'est donc **un langage de description de page** manipulant des objets graphiques de base comme les droites ou les courbes de Bézier (fig. 16), permettant de construire toute autre forme géométrique : rectangles, cercles, polices de caractères, etc. (fig. 17).

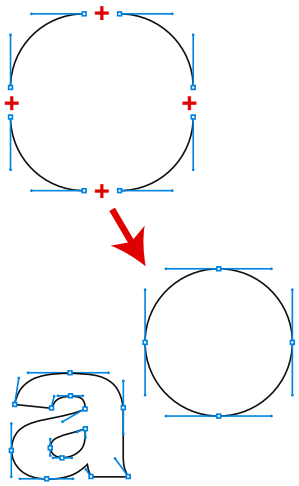


Fig. 17 : objets graphiques définis par des courbes de Bézier

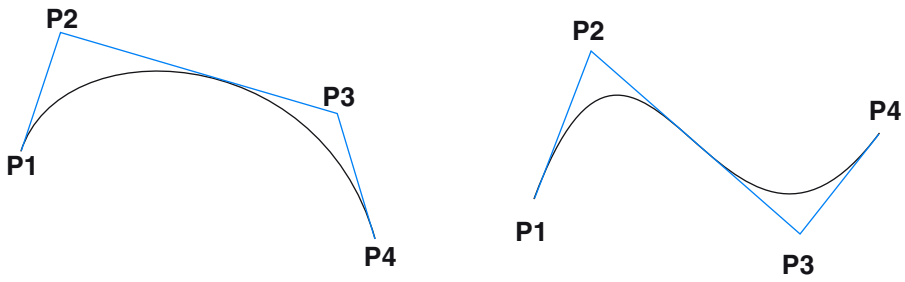


Fig. 16 : courbes de Bézier

À l'instar de monsieur Jourdain qui faisait de la prose sans le savoir, l'opérateur PAO programme en PostScript à travers son écran. Lorsqu'il dessine dans Illustrator, par exemple, le logiciel transforme les manipulations effectuées en **instructions PostScript** compréhensibles par n'importe quelle imprimante PostScript (fig. 18). Heureusement, car PostScript est un langage difficile demandant de solides connaissances en programmation.

Les courbes de Bézier ?

Inventées par un ingénieur français travaillant sur les flux aérodynamiques chez un grand constructeur automobile, elles sont devenues célèbres grâce à PostScript. Elles permettent de dessiner n'importe quelle courbe avec très peu d'informations, ce qui colle parfaitement à la philosophie de PostScript. En effet, seuls 4 points sont nécessaires quelle que soit la forme de la courbe.

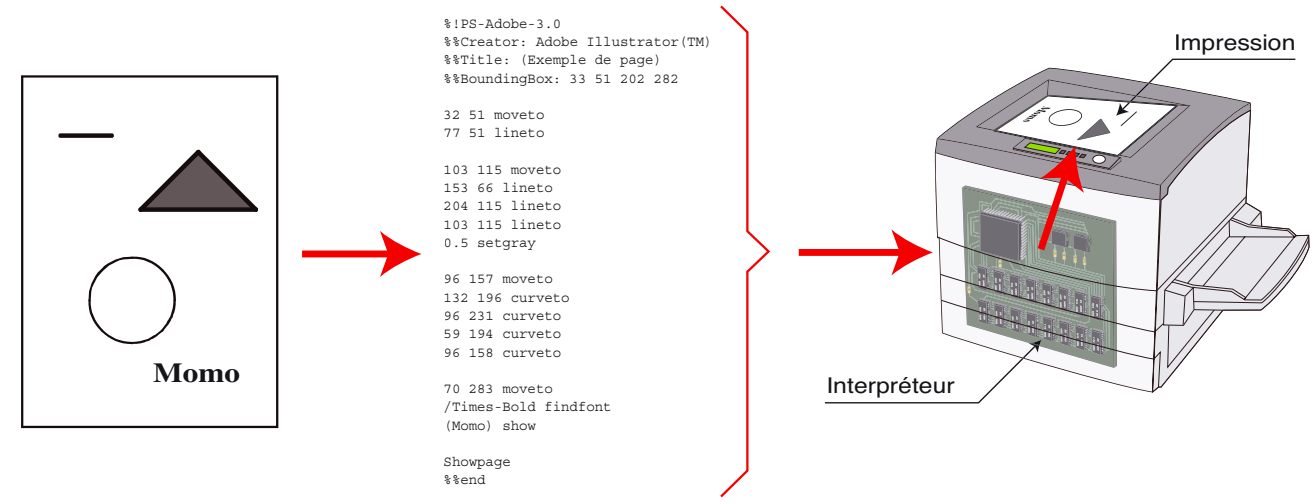


Fig. 18

Le format d'enregistrement du PostScript est l'EPS (de son vrai nom EPSF, Encapsulated PostScript File). **Un fichier EPS est constitué de quatre grandes parties** (fig. 19).

La première comprend l'en-tête où y figurent tous les commentaires concernant le document comme le nom du fichier, la version du logiciel qui a créé ce fichier, l'encombrement total de l'impression (bounding box), les couleurs utilisées, etc. (fig. 20 p. 34). Elles sont récupérées par des logiciels dédiés au traitement des documents. Puis viennent toutes **les instructions**, méthodes, procédures et fonctions nécessaires pour dessiner la page.

La fin de fichier regroupe des informations utiles à l'interpréteur du périphérique. L'en-tête et la fin du fichier sont communs à tous les fichiers PostScript alors que le corps est totalement différent d'un fichier à l'autre, en fonction des éléments à imprimer.

La dernière partie comprend **la visualisation en bitmap** du document à imprimer. On peut considérer cette visualisation comme une photo de la page. C'est cette image bitmap qui est en fait importée par les autres logiciels. Elle peut être considérée comme la basse résolution du fichier EPS (ne pas confondre avec la vignette qui est l'icône du fichier visible dans les fenêtres du bureau). Cette visualisation peut être enregistrée sous différents formats et profondeurs d'écrans. Un enregistrement en 1 bit fournira un aperçu en noir et blanc même si le document est en couleur. Cependant, cela n'altèrera en rien le fichier PostScript qui, de toutes façons, restituera les couleurs au moment de l'impression (fig. 21 p. 35).

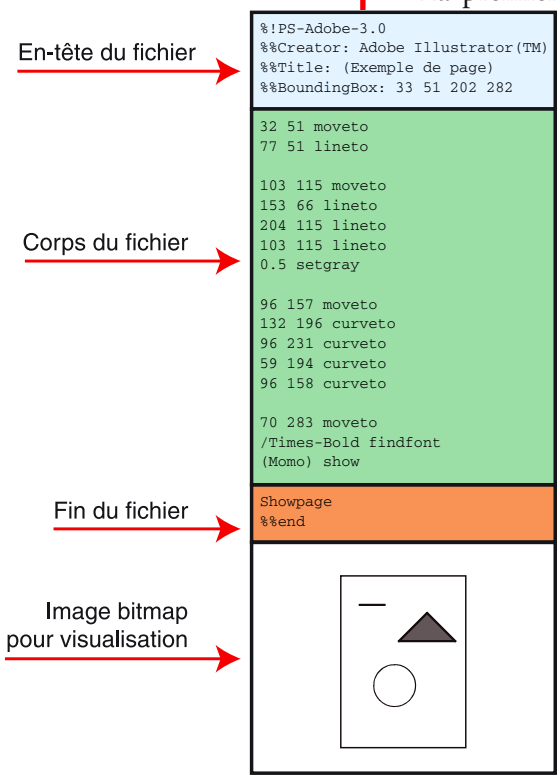


Fig. 19 : structure d'un fichier EPS

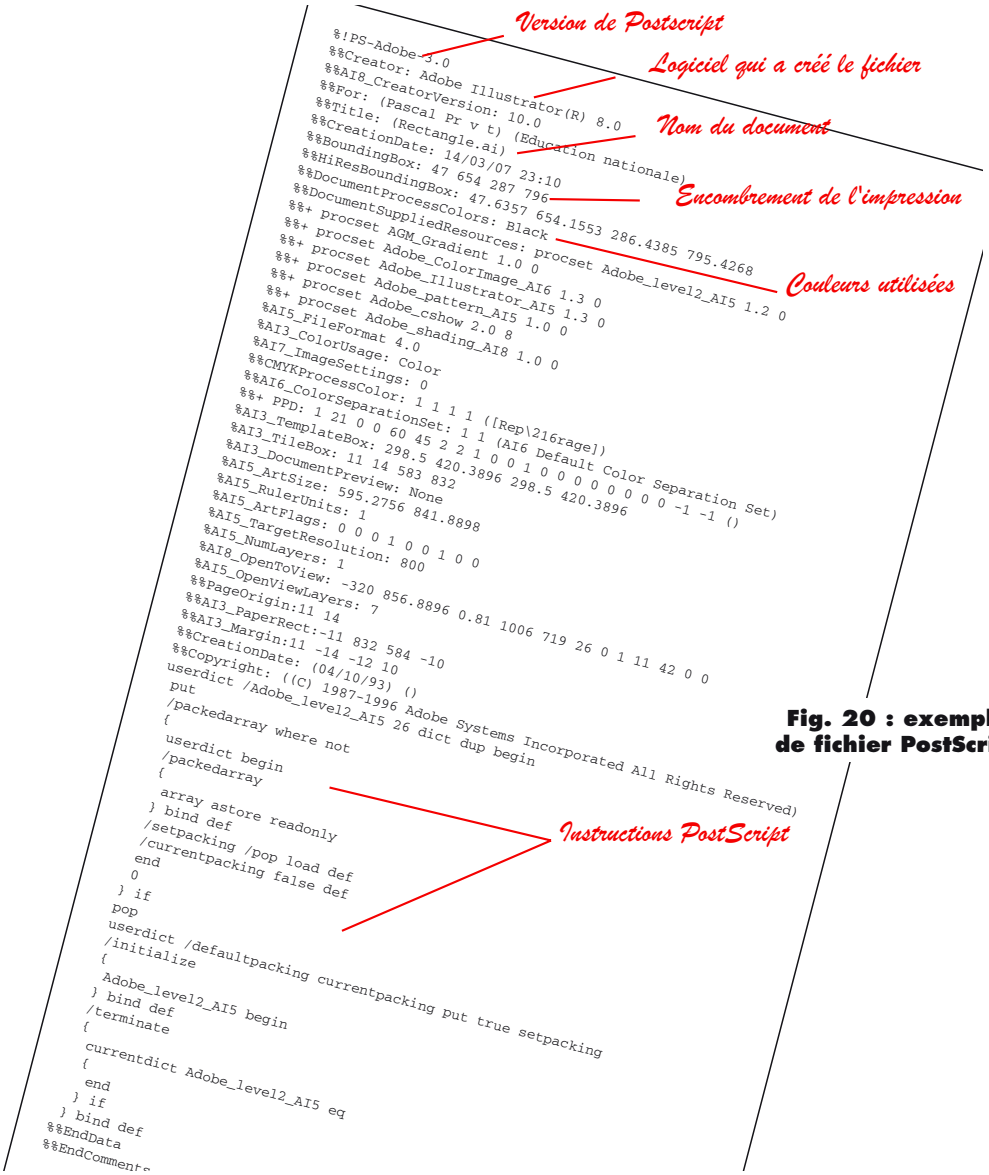


Fig. 20 : exemple de fichier PostScript

Un des autres avantages de PostScript est sa capacité à être transférable sur tous les systèmes d'exploitation. Cela s'explique tout simplement par le fait qu'un fichier PostScript est, en fait, du texte **codé** sur les 128 premiers caractères du code **ASCII**, ce qui évite tous les caractères spéciaux et accentués et rien n'est plus facile pour un ordinateur que de lire un **fichier texte**.

PostScript s'est donc vite imposé dans le processus de publication. Cependant, il a rapidement trouvé ses limites. Hormis le fait qu'il génère des fichiers volumineux, il n'enregistre pas les polices de caractère mais juste leurs attributs : nom, graisse, corps, etc. Si un fichier PS est envoyé à un intervenant tiers pour être exploité (sortie de films sur flasheuse notammment), la police utilisée risque de n'être pas exactement la même. Un "Garamont de Monotype" n'est pas construit identiquement à celui d'Adobe ou d'ITC par exemple. Que dire des polices, installées sur des systèmes d'exploitation divers. Le transfert d'un document Mac vers un PC (ou inversement) pose de gros problèmes, notamment de chasse et il est peu probable de retrouver la mise en page originelle.

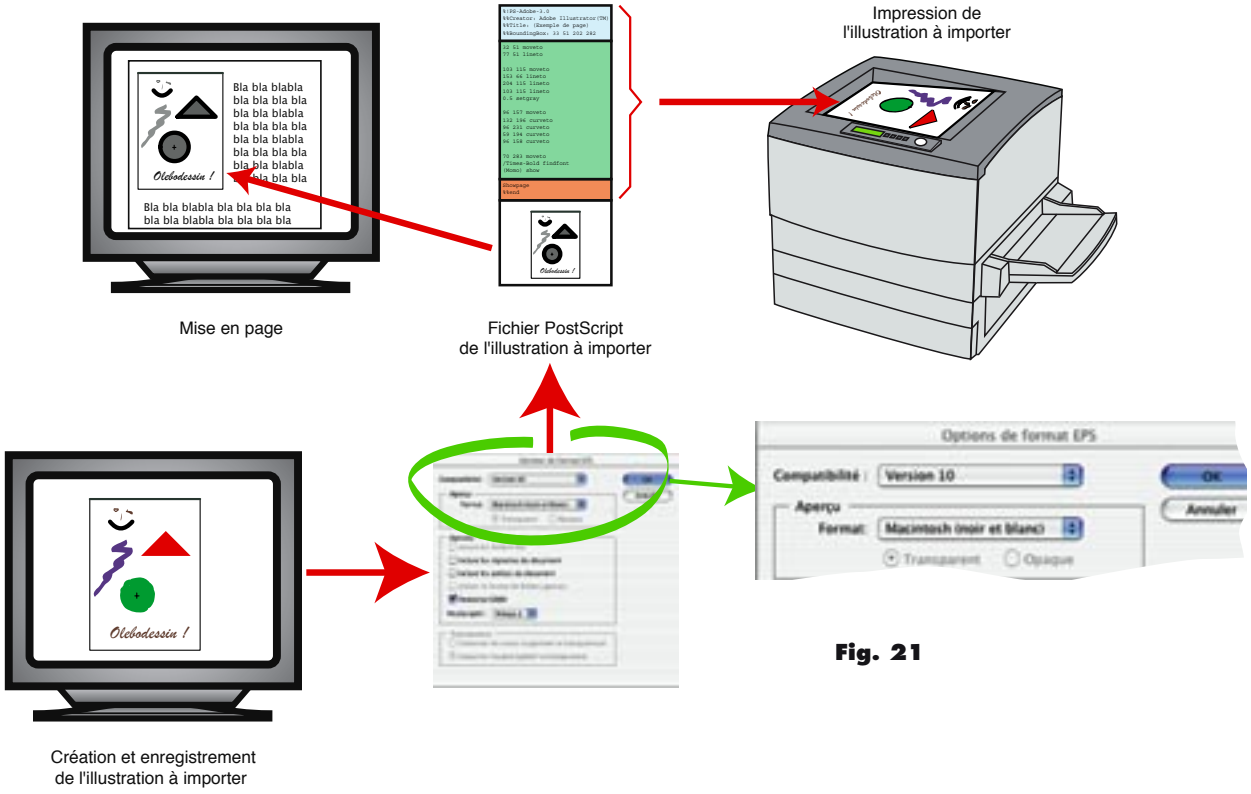


Fig. 21

Le saviez-vous ?

L'inventeur du PostScript, qui est aussi le créateur de l'entreprise Adobe Systems Inc, s'appelle John Warnock. Titulaire d'une maîtrise en mathématiques et d'un doctorat en informatique, il commença ses travaux d'association de visualisation-écran et de mémoire d'écran au début des années 70. Puis il rejoint le PARC (Palo Alto Research Center) de Xerox Corporation (où y fût inventée la souris et l'interface graphique) et développa, en 1978 avec Martin Newell, un premier langage de description de page appelé JAM. Ce langage donna naissance à Interpress qui n'arriva pas à s'imposer. En 1982, il développa, avec Charles Geshchke, le langage PostScript.

PostScript obtint ses lettres de noblesse, dans les arts graphiques, grâce à la sortie en 1984 du premier Macintosh et de l'imprimante Apple LaserWriter.

Le PDF

Le PDF est une évolution de PostScript. Il a été créé pour pallier les imperfections de ce dernier. Bien souvent un document s'imprime difficilement en raison d'erreurs PostScript. Cela est dû au fait que les interpréteurs PostScript ne « parlent » pas forcément tous, et exactement, le même langage PostScript. Les constructeurs de périphériques de sortie ou les éditeurs de logiciels implantent dans leurs produits leur propre interpréteur et donc le programment différemment (le PostScript est un langage de programmation !). Pour simplifier, on peut parler de « patois PostScript » et la restitution d'un rectangle, par exemple, ne s'exprimera pas exactement de la même manière chez un constructeur que chez un autre, même s'ils parlent tous les deux d'un rectangle.

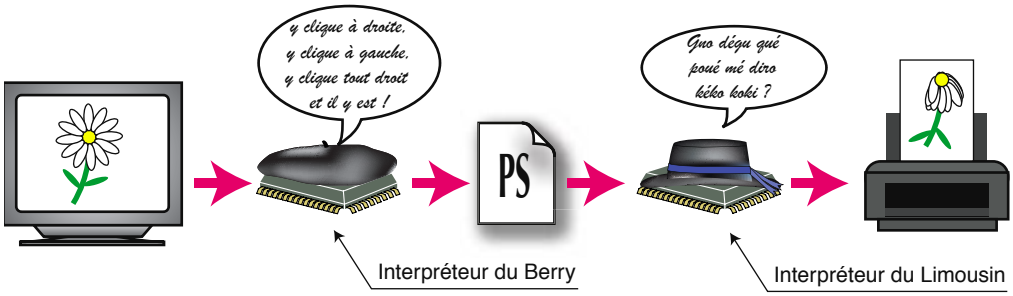


Fig. 22 : problème de communication en PostScript

Cela veut dire que si les instructions du patois PostScript n° 1 sont envoyées à l'interpréteur qui cause le patois PostScript n° 2, il peut y avoir des problèmes de communication et le rectangle demandé risque ne plus être tout à fait un rectangle. En réalité, cela ne peut pas se passer avec un motif géométrique aussi simple qu'un rectangle mais s'il s'agit d'un document complexe avec des dégradés, beaucoup de polices de caractères ou des tracés compliqués, la probabilité de ne pas se retrouver avec le résultat attendu est grande (fig. 22). D'où l'importance de travailler avec des interpréteurs de même marque lorsque cela est possible, ou bien avec un seul et unique interpréteur pour tous les périphériques de sortie. Cela évite de se retrouver avec des milliers d'exemplaires en sortie de presse offset différents du bon à tirer que le client a signé et validé.

Le PDF contourne ce problème en indiquant seulement des ordres généraux, des « objets graphiques ». En fait, un rectangle (pour reprendre notre exemple) sera codé « rectangle » et peu importe la manière dont un interpréteur le restituera puisqu'on lui demandera juste de réaliser un rectangle. Qu'il le fasse avec le patois PostScript bidule ou le patois PostScript machin, il fera un rectangle. C'est simple et efficace et cela préserve la mise en page (polices, illustrations, graphiques...) telle

que souhaitée par l'opérateur PAO. Le codage d'un fichier PDF se fait en ASCII comme PostScript. En partant de ce constat, le PDF devient un format de fichier ultra-portable, c'est-à-dire capable d'être lu avec certitude sur tous les systèmes d'exploitation de n'importe quel ordinateur au monde. D'ailleurs PDF veut dire : Portable Document Format (format de document portable). Adobe inc qui l'a créé a très bien compris l'intérêt d'un tel type de fichier et a fourni gratuitement à la planète entière un logiciel permettant de lire un document PDF sur tous les ordinateurs (Mac, PC, Linux). Il s'agit du fameux « Acrobat Reader ». Cette ouverture a fait le succès du PDF et son utilisation s'est très largement répandue dans de nombreux domaines de l'informatique (bureautique, organiseurs numériques personnels, téléphones portables, etc.).

Les fichiers PDF peuvent englober des photos (bitmap), être paramétrables, sécurisés et interactifs. On peut même y rechercher de l'information facilement. Ils sont donc optimisables pour leur utilisation finale (fig. 23).

Un document PDF concernant un manuel d'utilisation en ligne pourra comporter des signets, ainsi que des hyperliens pour y naviguer facilement de page en page. Un document destiné à l'affichage à l'écran pourra ne comporter que des images à 72 dpi (c'est souvent le cas des bons à tirer envoyés par e-mail). Un document destiné à l'impression sera paramétré avec des images en haute résolution et comprendra les traits de coupe, les repères de montage, les gammes, etc. Des styles pré-programmés (ebook, écran, impression, presse, styles personnalisés, fichiers légers, etc.) sont disponibles dans beaucoup d'applications destinées à la production de produits imprimés.

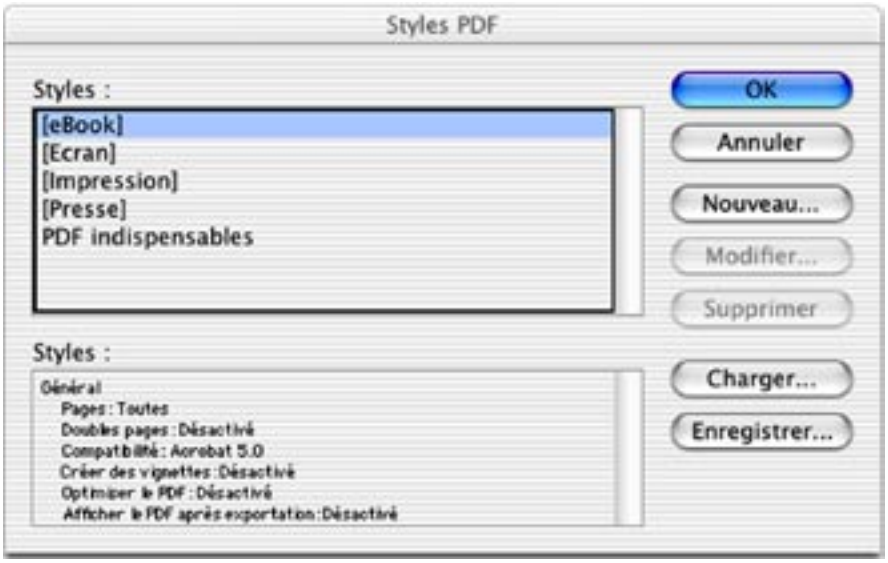


Fig. 23 : styles PDF

La sécurité d'accès aux documents n'est pas oubliée. On peut adjoindre aux fichiers PDF un mot de passe pour autoriser son ouverture et/ou sa modification, en interdire son impression ou en extraire des éléments (schéma, texte, etc.). Des standardisations du PDF comme le PDF-X, le PDF-A, etc., augmente encore la sécurité et la fiabilité des fichiers destinés à l'impression.

Les polices numériques

Il n'y a pas de communication écrite sans caractères d'imprimerie et nos bons vieux caractères en plomb sont maintenant avantageusement remplacés par les polices de caractères numériques. Une police est l'ensemble des déclinaisons d'un type de caractère d'imprimerie (romain, gras, italique, gras-italique, condensé, etc.). Les polices de caractères sont conçues par des spécialistes. La déclinaison en italique d'une lettre n'est pas seulement sa déformation penchée, tout comme son changement de corps (taille) n'est pas une simple réduction. Le dessin d'une lettre va donc changer en fonction de sa déclinaison ou de sa taille. Les graveurs et fabricants de caractères en plomb (les fondeurs) prenaient en compte ces variations pour pallier les impératifs techniques de l'impression, pour éviter des phénomènes de bouchage dans les contre-poinçons des caractères de petits corps par exemple (fig. 24). Un professionnel des industries graphiques se doit donc d'utiliser les polices à bon escient : ne pas pencher ses caractères pour en faire de l'italique, ne pas utiliser la fonction « gras » d'un logiciel de mise en page pour les caractères gras, ne pas étroitiser (réduction horizontale) pour les caractères « condensés », entre autres (fig. 25).

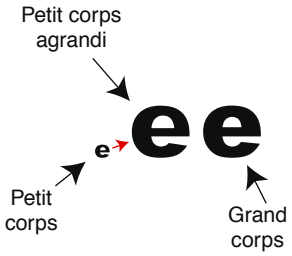


Fig. 24

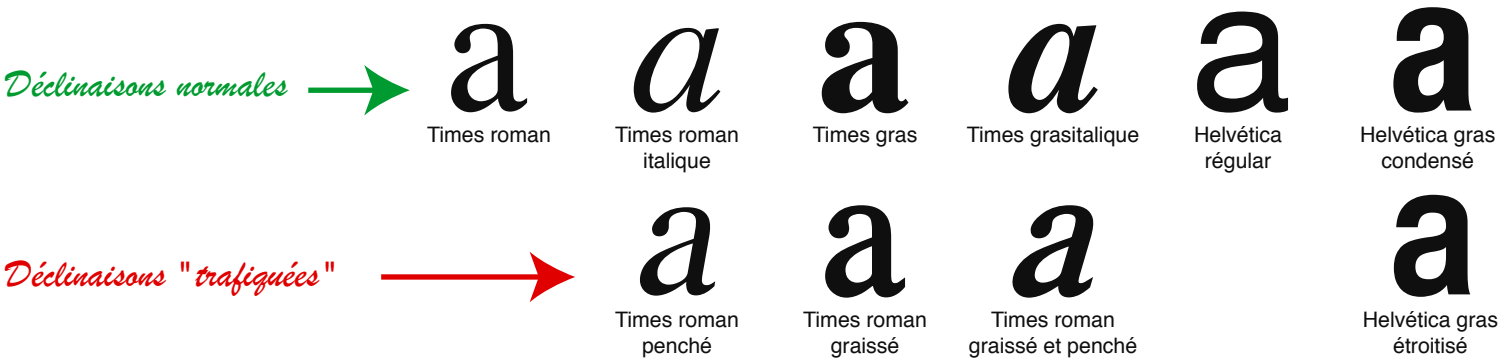


Fig. 25 : déclinaison des caractères

La nature même d'un caractère numérique pose des problèmes de reproduction. Ce caractère doit être restitué le plus fidèlement possible, c'est à dire reconstitué avec des pixels. Il faut pouvoir adapter le dessin du caractère à la résolution du périphérique de sortie. C'est particulièrement vrai pour les petits corps et les caractères possèdant des finesses. Par exemple, si une partie du caractère est définie à 2,5 pixels de large, le périphérique de sortie ne sera pas en mesure de l'afficher. En effet, 2,5 pixels deviendront 2 pixels ou 3 pixels. Pour de basses résolutions, cela peut avoir une incidence visuelle déplorable. Des algorythmes, appelés « hints » et incorporés au fichier de la police numérique, optimisent ces problèmes.

Fig. 26a : sans antialiasing

Fig. 26b : avec antialiasing

Les formats de fichier des polices de caractère

Il existe 3 grands types de formats couramment utilisés en industries graphiques : les polices PostScript type 1, les polices True Type et les polices Open Type (ou unicode).

PostScript type 1

Ce sont les plus anciennes, mais aussi les plus fiables qui hantent nos ordinateurs. Elles ont été développées par les inventeurs même de PostScript : Adobe. Leur particularité est d'être « constituées » de deux fichiers distincts : un fichier bitmap pour l'affichage à l'écran (dans les corps 8, 10, 12, 18, 24) et d'un fichier vectoriel pour l'impression. L'ensemble est regroupé dans des « valises » (« suitcase » en anglais), c'est-à-dire un type de dossier très particulier et reconnaissable par le système d'exploitation (fig. 27). Si l'agrandissement des caractères à l'impression ne pose aucun problème dû à leur nature vectorielle, l'affichage à l'écran des bitmaps crée une pixellisation. Ce problème est contourné grâce à des logiciels du type ATM intégrés dans le système comme MAC OS X par exemple. Ces logiciels utilisent une technique appelée « antialiasing » qui lisse les contours des caractères en utilisant des niveaux de gris (fig. 26b). les polices PostScript Type 1 sont généralement développées par les grands fondeurs et créateurs de logiciels. Elles sont viables et sans surprise. Elles sont à conseiller pour un usage professionnel.

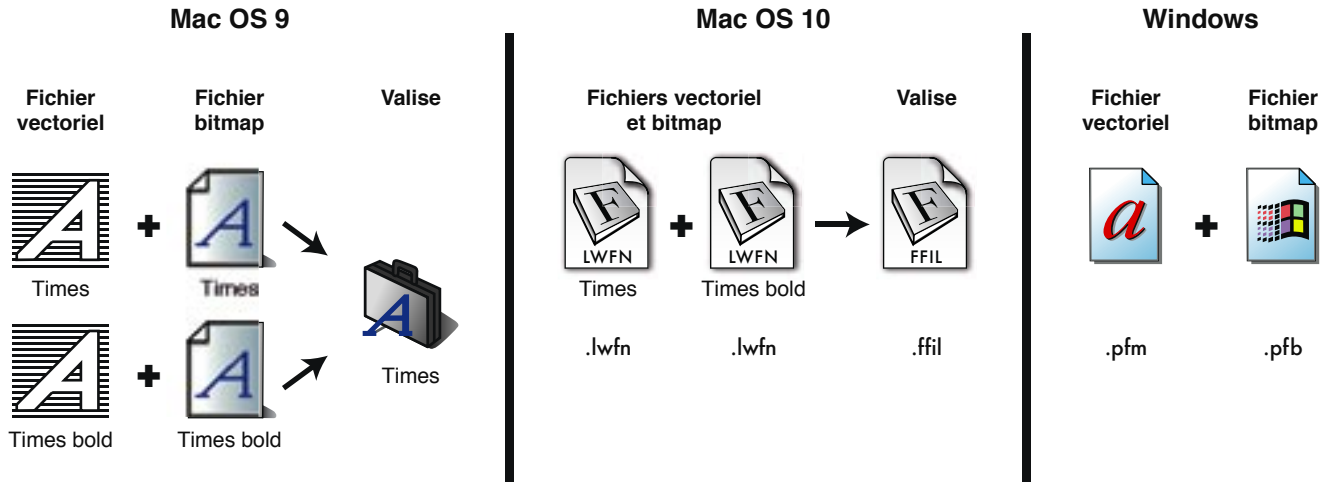


Fig. 27 : icônes des fichiers de polices PostScript type 1

Police True Type

Elles ont été développées par Apple et Microsoft. Un seul type de fichier constitue ces polices et il est vectoriel. Il sert autant à l'affichage écran qu'à l'impression. Le seul véritable inconvénient est que ce type de police de caractère n'est pas toujours totalement compatible avec les interpréteurs PostScript. Certains caractères n'apparaissent pas ou bien sont symbolisés par un affreux petit rectangle blanc bordé de noir. C'est bien évidemment du plus mauvais effet dans un document professionnel. Attention, ces polices peuvent être restituées correctement sur une imprimante laser ou jet d'encre mais pas lors de l'opération de flashage des films ou des plaques sur CTP. Leur usage est déconseillé en industries graphiques, à moins d'être certain de leur bon fonctionnement. Si vous travaillez avec des partenaires de la chaîne graphique, demandez-leur si leurs interpréteurs (les RIP) savent « digérer » les polices True Type. Au pire, si vous n'êtes pas certains, vectorisez vos caractères. La vectorisation, c'est la transformation en tracés vectoriels purs et durs. Vos caractères ne réagissent plus comme du texte éditable et transformable (texte actif), mais deviennent de simples dessins et il n'est plus possible de modifier ce texte facilement.

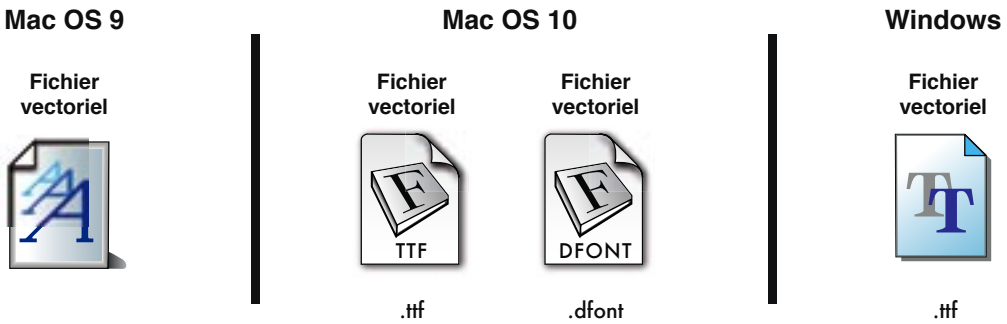


Fig. 28 : icônes des fichiers de polices True Type

Polices Open Type

La tendance à la compatibilité totale entre systèmes d'exploitations a poussé Adobe et Microsoft à développer un format de polices de caractères universel. Comme True Type, ce format n'utilise qu'un seul type de fichiers pour l'affichage à l'écran comme pour l'impression. En fait, un même fichier regroupe à la fois les polices vectorielles et bitmap ou True Type, autant pour Mac OS et Windows, dans leurs différentes déclinaisons, ainsi que leur définitions métriques. Contrairement aux autres formats cités plus avant, chaque caractère est codé sur deux octets. Cela autorise un jeu de près de 65 000 caractères contre 256. Les polices Open Type (pour les plus sophistiquées) peuvent posséder toutes les variantes de caractères, comme les ligatures (groupe de lettres reliées entre elles, fig 29), lettres ornées, les vraies petites capitales, les fractions, etc., ainsi que tous

ffi

Fig. 29 : ligature des 3 lettres f, f et i

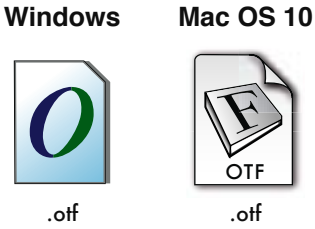


Fig. 30 : icônes des polices Open Type

les signes et caractères des langues étrangères, même non latines (arabe, cyrillique, grec…). Elles sont bien sûrs utilisables sans conversion sur tous les systèmes d'exploitation et évitent les problèmes d'incompatibilité (un document sur Macintosh sera restitué sans problème sur un PC et inversement).

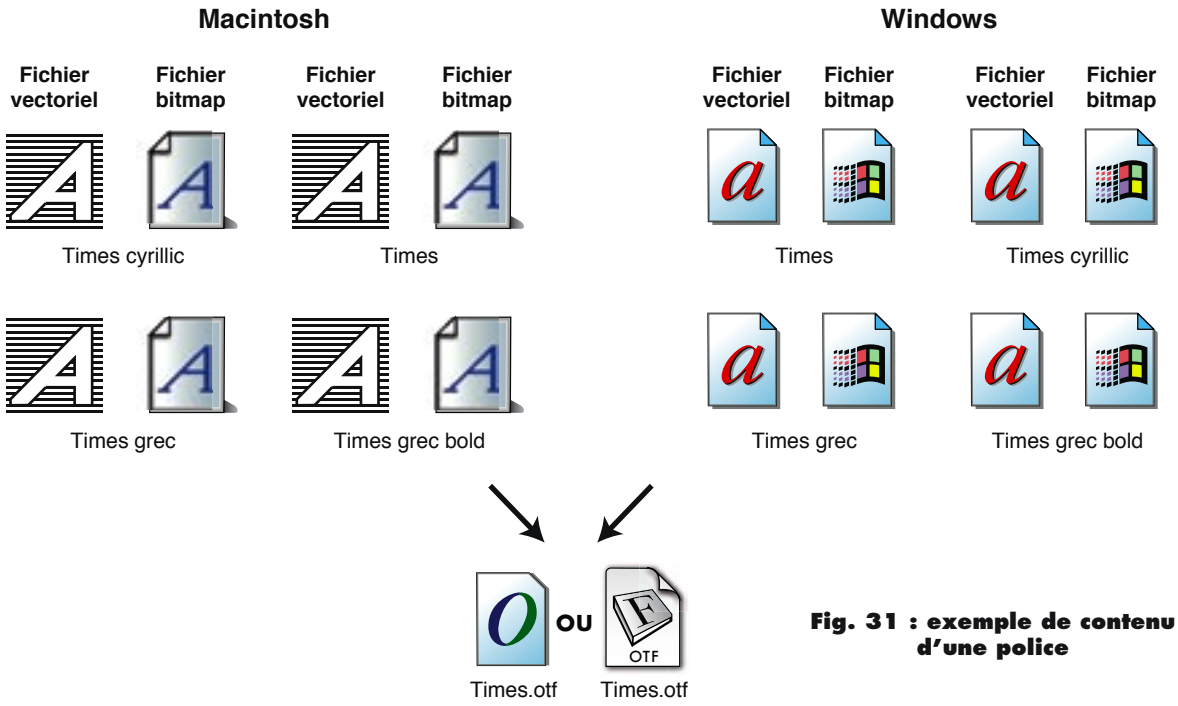


Fig. 31 : exemple de contenu d'une police

Gestion des polices

Il est nécessaire de faire un tri très sélectif des polices que l'on possède sur son système informatique et une typothèque (l'ensemble des polices dont dispose un utilisateur) doit être parfaitement gérée tant en terme de qualité que de quantité. Il est clair qu'il existe des polices de caractères de meilleure qualité que d'autres. Les polices téléchargées gratuitement sur Internet doivent être bannies d'une typothèque professionnelle. En effet, ces polices ne contiennent pas forcément de hints et donc ne sont pas optimisées (c'est long et coûteux à développer et seuls les grands fondeurs en ont les moyens), elles ne possèdent pas forcément tous les caractères, notamment ceux accentués. Elles peuvent également poser de gros problèmes à l'impression.

Un nombre important de polices nuit à la créativité. En effet, quelle police choisir pour illustrer le message à faire passer quand le choix est trop important ? Les chances de se tromper sont grandes et l'on risque de faire passer ses propres

goûts artistiques au détriment de l'efficacité. Il vaut mieux se contenter de quelques polices que l'on maîtrise bien (au sens artistique).

La multiplicité des polices est source de ralentissement de la machine. Il faut savoir que les polices sont chargées en mémoire vive lors du démarrage de l'ordinateur. Donc, plus il y en a et plus ça « rame ». Les logiciels dédiés (comme Suitcase, livre des polices, etc.) permettent d'activer seulement les polices utilisées pour un travail bien précis. De plus, ils permettent une gestion plus efficace : suppression des doublons, suppression sans risque des polices inutiles, installation automatique dans les bons dossiers du système d'exploitation, visualisation des caractères, classement par thèmes, production d'un catalogue des polices disponibles, etc.

Attention cependant de ne pas supprimer « les polices système » de votre ordinateur. Ces polices sont absolument nécessaires au bon fonctionnement de votre machine car utilisées par le système d'exploitation (affichage du nom des fenêtres par exemple).

Les polices numériques sont des fichiers informatiques et sont soumis aux droits d'auteurs. Il est formellement interdit de les dupliquer. Il est juste toléré de joindre les polices d'un document que l'on a créé à un partenaire tiers (un flasheur par exemple), pour qu'il puisse traiter le document dans de bonnes conditions. Ces polices de même nom et de même dessin peuvent ne pas avoir les mêmes caractéristiques informatiques (qualité d'affichage, chasse, format, etc.). Après traitement du document, le partenaire tiers doit enlever les polices prêtées de son système informatique.

Glyphes ou caractères ?

Ces deux notions sont souvent confondues.

Un glyphe, c'est la forme donnée à un caractère. Par exemple, le « e » bas de casse et le « e » petite capitale forment un même caractère : le « e » mais deux glyphes distincts.

Un glyphe peut aussi représenter plusieurs caractères comme les ligatures (ffi, œ, par exemple).