Objectif
- Appréhender les techniques utilisées pour représenter numériquement les grandeurs qui nous entourent

16 CODAGE DES INFORMATIONS

On sait qu'aujourd'hui nos ordinateurs, téléphones et autres appareils savent manipuler aussi bien des nombres et du texte que des images, de la vidéo ou de la musique… Mais comment représenter, au sein d'un système numérique, cette diversité des objets du monde réel ou virtuel ?

1 Terminologie

a Unité de codage

Les composants constituant un système informatique réagissent, de manière interne, à des signaux « tout ou rien ». On représente les deux états stables ainsi définis par les symboles « 0 » et « 1 » ou encore par « L » (*low*) et « H » (*high*). Le système de numération adapté à la représentation de tels signaux est donc la **base 2**, on parle aussi de **codage binaire**.

L'**unité de codage** de l'information est donc un élément ne pouvant prendre que les valeurs 0 ou 1 : le **bit** (contraction de *Binary Digit*, c'est-à-dire *Binary Digital Unit*).

b Unité de transfert

Pour les échanges de données, les informations élémentaires (bits) sont manipulées par groupes qui forment ainsi des mots binaires. La taille de ces mots est le plus souvent un multiple de 8 = 2^3.

L'**unité de transfert** utilisée pour les échanges de données est le mot de 8 bits appelé **octet** (*byte*).

Dans un mot binaire, le bit situé à gauche est le bit le plus significatif, **MSB** (*Most Significant Bit*), celui de droite est le bit le moins significatif, **LSB** (*Less Significant Bit*).

Pour faciliter les manipulations, un octet peut aussi être divisé en deux mots de 4 bits que l'on appelle des quartets : celui situé à gauche est le quartet de poids fort, **MSQ** (*Most Significant Quartet*), et celui situé à droite, le quartet de poids faible, **LSQ** (*Less Significant Quartet*).

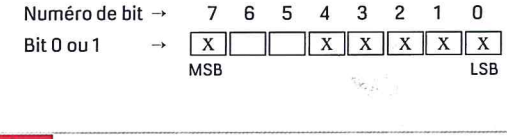
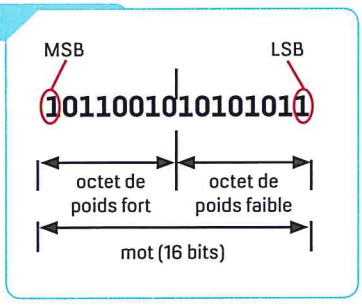


Fig. 1 Constituants d'un octet

c Mots binaires

En dehors de l'unité de transfert (*octet*), des regroupements plus importants de bits sont couramment utilisés : le **mot de 16 bits** = 2 octets (*word*), le **mot de 32 bits** = 4 octets (*double word*), et le **mot de 64 bits** = 8 octets (*quad word*)…

EXEMPLE



Traditionnellement, dans les notations de quantité binaires, les préfixes « kilo », « Méga »… sont utilisés pour exprimer des multiples en puissances de 2 ; mais cet usage est contraire aux normes SI en vigueur !

→ ko (kB)	=	kilo-octet (kiloByte)	=	10^3 octets	= 1000 octets
→ Mo (MB)	=	Méga-octet (MegaByte)	=	10^6 octets	= 1000 ko
→ Go (GB)	=	Giga-octet (GigaByte)	=	10^9 octets	= 1000 Mo
→ To (TB)	=	Téra-octet (TeraByte)	=	10^{12} octets	= 1000 Go
→ kio (kiB)	=	kibi-octet (kibiByte)	=	2^{10} octets	= 1024 octets
→ Mio (MiB)	=	Mébi-octet (MebiByte)	=	2^{20} octets	= 1024 ko
→ Gio (GiB)	=	Gibi-octet (GibiByte)	=	2^{30} octets	= 1024 Mo
→ Tio (TiB)	=	Tébi-octet (TebiByte)	=	2^{40} octets	= 1024 Go

norme CEI 60027-2 – International Electrotechnical Commission – déc. 1998, préfixes adoptés par IEEE 1541 / EN 60027-2:2007
(k , M , G , T , … = multiples du Système International – b = *bit* – B = *byte* – bi = *binary*)

Fig. 2 Unités multiples de l'octet

La capacité en octets des différents constituants tels que circuits mémoire, disques durs… est souvent très importante : il devient alors indispensable d'utiliser des **unités multiples de l'octet**.

ACTIVITÉ 1 **La fiche technique d'un disque dur externe indique** une capacité de 320 GB, exprimez cette capacité en Mio.

ACTIVITÉ 2 **Votre fournisseur ADSL vous annonce un débit descendant de 8 192 kibits/s.**
Vous faites une mesure de débit réel et vous trouvez une moyenne de 3 280 kibits/s.
Quel sera le temps théorique minimal de téléchargement d'une application de taille égale à 25 Mo ?

2 Codage binaire

Comme toutes les autres bases de calcul, la base 2 utilisée pour le codage binaire répond à la règle suivante :

Pour un nombre codé en base B, chaque symbole de ce nombre, de la gauche vers la droite, est un multiplicateur d'une des puissances successives de B.

EXEMPLE

365_{10} (base 10, donc 10 symboles de représentation : 0,1,2,3,4,5,6,7,8,9)

$3 . 10^2 + 6 . 10^1 + 5 . 10^0$

$300 + 60 + 5$

$= 365$

101101101_2 (base 2, donc 2 symboles de représentation : 0,1)

$1 . 2^8 + 0 . 2^7 + 1 . 2^6 + 1 . 2^5 + 0 . 2^4 + 1 . 2^3 + 1 . 2^2 + 0 . 2^1 + 1 . 2^0$

$256 + 64 + 32 + 8 + 4 + 1$

$= 365$

puissances successives de 2 très faciles trouver : , 4, 8, 16, 32, 64, , 256, 512, 1024, 48, ...

Comme pour toutes les autres bases de calcul, la plage de valeurs d'un nombre dépend de la quantité de symboles le constituant :

Un nombre codé avec N symboles en base B peut prendre B^N valeurs entières différentes (soit l'intervalle 0.. B^N – 1).

EXEMPLE

Un mot de **3 symboles en base 10** permet de coder les nombres de 0 à 10^3-1 (1 000 valeurs : 0...999)

Un mot de **8 symboles en base 2** peut prendre 2^8 = 256 valeurs distinctes, de 0 à 2^8-1 (0...255) :

%00000000	(0)
%00000001	(1)
%00000010	(2)
... ...	
%11111101	(253)
%11111110	(254)
%11111111	(255)

} 256 valeurs

préfixe % peut être isé pour spécifier valeur en base 2...

3 Algèbre de Boole

En plus des opérations arithmétiques de base (addition, soustraction, multiplication, et division), tous les systèmes combinatoires sont soumis aux règles de l'algèbre de l'anglais Boole (1815 – 1864) qui traduisent les raisonnements logiques à partir de propositions vraies ou fausses.

L'algèbre de Boole est commutative, associative et distributive. Un **booléen** (*boolean*) est donc un élément binaire, vrai ou faux, « 1 » ou « 0 ».

Tout processeur a la capacité d'effectuer des opérations arithmétiques et booléennes, qui sont fondamentales pour le développement des applications logicielles...

Variables		NON (*NOT*)		ET (*AND*)	OU (*OR*)	OU Exclusif (*XOR*)	Lois de Morgan	
a	b	$\overline{a}$	$\overline{b}$	a *AND* b	a *OR* b	a *XOR* b	$\overline{a \text{ OR } b} = \overline{a} \text{ AND } \overline{b}$	$\overline{a \text{ AND } b} = \overline{a} \text{ OR } \overline{b}$
0	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	0	1
1	0	0	1	0	1	1	0	1
1	1	0	0	1	1	0	0	0

Fig. 3 Table de vérité de l'algèbre de Boole

Note : a *XOR* b est la somme disjonctive égale à (a *AND* $\overline{b}$) OR (b *AND* $\overline{a}$)

ACTIVITÉ 3 Convertir manuellement les valeurs suivantes (base 2 ↔ base 10) :

25_{10}, 1000_{10}, 1000_{2}, 11010000000_{2}

ACTIVITÉ 4 Calculez manuellement les 6 expressions ci-dessous sans effectuer de conversion.

%1101 + %100 %1100 – %101 %10101 x %11

%101011 *XOR* %110 %11000001 *OR* %1011 %01010101 *AND* %11111011

4 Représentation hexadécimale

Le binaire, s'il est très représentatif du codage interne des machines, reste très délicat et fastidieux à manipuler... Les programmeurs ont très vite ressenti la nécessité d'utiliser une **représentation** plus rapide des nombres binaires.

Imaginons de représenter chaque quartet binaire par un unique symbole... Un quartet permettant de coder 2^4 valeurs (soit de 0 à 15), il faut donc trouver une base de représentation disposant de **16 symboles** : il s'agit de la base 16 (appelée hexadécimale).

Le préfixe $ peut être utilisé pour spécifier une valeur en base 16...

Cette base de calcul répond aux mêmes règles que les autres (décimale, binaire...) en ce qui concerne les plages de valeurs et les techniques de conversion. Elle est représentée par les 10 chiffres et les 6 premières lettres de l'alphabet.

Base 16	Base 2	Base 10
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Fig. 4 Table de correspondance

EXEMPLE

Conversion d'un mot de 16 bits

%	1	0	1	0	1	0	0	1	1	1	0	1	0	1	0	1
$	A				9				D				5			

$A9D5 se décompose en :

$A.16^3 + 9.16^2 + D.16^1 + 5.16^0$

$10.4096 + 9.256 + 13.16 + 5.1$

$40960 + 2304 + 208 + 5 = 43477_{10}$

ACTIVITÉ 5 Convertir les valeurs suivantes (base 16 ↔ base 10 ↔ base 2) : 1492_{10} 1515_{16} $BAFFE_{16}$

5 Interprétation signée

Un mot binaire de longueur donnée permet naturellement de coder la valeur nulle et un nombre plus ou moins élevé de valeurs entières strictement positives... Mais comment les **valeurs entières négatives** sont-elles représentées ?

Par convention, le MSB est interprété comme **bit de signe** (le nombre est considéré négatif si le MSB est égal à 1), tout en assurant l'homogénéité des calculs :
– 4 + 6 doit donner +2 !

La figure ci-contre traite du cas des mots de 8 bits. La plage des valeurs signées est ainsi limitée à l'intervalle –128..+127.

Le passage d'une valeur positive à la valeur négative de même valeur absolue peut être obtenu par le complément à 2 (inversion bit à bit puis ajout de 1) : $-N = \overline{N} + 1$

Codage réel	Hexa.	Interprétation décimale signée	
% 1000.0000	($80)	- 128	(plus grand nombre négatif)
% 1000.0001	($81)	- 127	
% 1000.0010	($82)	- 126	
... ...		...	...
% 1111.1110	($FE)	- 2	
% 1111.1111	($FF)	- 1	
% 0000.0000	($00)	0	(valeur centrale)
% 0000.0001	($01)	1	
% 0000.0010	($02)	2	
... ...		...	...
% 0111.1110	($7E)	126	
% 0111.1111	($7F)	127 (plus grand nombre positif)	

Fig. 5 Interprétation signée sur 8 bits

EXEMPLE

Comment écrire -123_{10} en binaire ?

$N = +123_{10} \rightarrow N = \%\ 0111.1011 \rightarrow N = \%\ 1000.0100 \rightarrow -N = \%\ 1000.0101$

ACTIVITÉ 6 Exprimez, en décimal, les valeurs 8/16 bits signées % 0001.1111, % 1010.1100, % 1000.1010.1101.0000.

6 Codage des entiers relatifs : synthèse

Taille	Interprétation	Intervalle		
8 bits	signée	–128 ... +127	$80 ... $7F	$-2^7 ... +(2^7-1)$
	non signée	0 ... 255	$00 ... $FF	$0 ... (2^8-1)$
16 bits	signée	–32 768 ...+32 767	$8000 ... $7FFF	$-2^{15} ... +(2^{15}-1)$
	non signée	0 ... 65 535	$0000 ... $FFFF	$0 ... (2^{16}-1)$
32 bits	signée	–2 147 483 648 ... +2 147 483 647	$80000000 ... $7FFFFFFF	$-2^{31} ... +(2^{31}-1)$
	non signée	0 ... 4 294 967 295	$00000000 ... $FFFFFFFF	$0 ... (2^{32}-1)$

Fig. 6 Codage des entiers relatifs sur 8, 16 et 32 bits

Il est ainsi possible de coder, à partir d'assemblages de bits, un sous-ensemble plus ou moins important de l'ensemble mathématique des entiers relatifs... Mais le monde n'est pas constitué exclusivement de grandeurs entières. Une nouvelle question se pose alors : comment représenter les valeurs à virgule ?

7 Codage des grandeurs réelles

Toute valeur réelle peut être représentée sous forme d'une **notation scientifique normalisée** composée d'un signe, d'un nombre à virgule de partie entière nulle et d'un multiplicateur exprimé en puissances de 10.

L'ensemble des chiffres situés à droite de la virgule s'appelle la **mantisse** M, la puissance signée du multiplicateur est nommée **exposant** E.

EXEMPLE

123,45 → normalisation → $+0{,}12345 . 10^{+3}$

→ signe positif, mantisse M = 12345, exposant E = +3

Cette normalisation n'est pas suffisante ! Il faut en plus respecter un standard de codage qui défini comment coder en binaire chacun des éléments sur une taille de mot préfixée. Le standard en vigueur est la norme P754 de l'IEEE (*Institute of Electrical and Electronics Engineers*), elle précise notamment l'encodage des nombres réels en simple précision (sur 32 bits), en double précision (sur 64 bits) et en précision étendue (sur 80 bits).

La figure ci-contre détaille la norme P754 simple précision (32 bits), elle permet :

- le codage de ±0, de ±∞ ;
- des intervalles ± ($1.4E^{-45}$ ... $3.4E^{+38}$) ;
- avec une précision de 2^{-23}.

Sur 64 bits, la norme autorise des intervalles ± (≈0 ... 1.7E+308) avec une précision de 2^{-52} !

bit	31	30	29	28	27	26	25	24	23	22	21	...	1	0
	S	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	...	2^{-22}	2^{-23}
	signe	exposant E (8 bits)								mantisse M (23 bits)				

contenu des 32 bits		valeur réelle
E = 0	M = 0	$(-1)^s * 0$
	M ≠ 0	$(-1)^s * 2^{-126} * (0.M)$
0 < E < 255		$(-1)^s * 2^{E-127} * (1.M)$
E = 255	M = 0	$(-1)^s *$ infini
	M ≠ 0	Nan *(Not a number)*

Fig. 7 Codage P754 simple précision

EXEMPLE

Comment exprimer en base 10 le nombre réel de codage IEEE 754 simple précision égal à $C2D5A000 ?

$C2D5A000 = % 11000010.11010101.10100000.00000000
Signe = 1, Exposant = 10000101, Mantisse = 10101011010000000000000

→ nombre négatif

→ $E = 2^7 + 2^2 + 2^0 = 133$

→ forme $(-1)^S . 2^{E-127} . (1,M)$ → $-1,1010101101 . 2^6$ → $-1101010,1101$

→ $-(2^6 + 2^5 + 2^3 + 2^1 + 2^{-1} + 2^{-2} + 2^{-4})$

→ $-(64 + 32 + 8 + 2 + 0,5 + 0,25 + 0,0625)$

→ $-106,8125$

ACTIVITÉ 7 Exprimez en hexadécimal le codage IEEE 754 simple précision du nombre 521,625.

8 Codage des caractères

Savoir encoder en binaire les nombres, qu'ils soient entiers ou réels, est une bonne chose… Mais une grosse part des informations manipulées par les systèmes numériques concerne le langage parlé ou écrit matérialisé sous forme de textes, eux-mêmes constitués de caractères typographiques. Comment coder universellement ces caractères et permettre ainsi l'échange d'informations entre machines et/ou utilisateurs, quelle que soit la langue utilisée (français, anglais, italien, chinois, russe…) ?

a Le code ASCII (ANSI X3.4:1986)

e morse (inventé par américain Samuel .B.Morse en 1844) a té le premier codage permettre une com- unication orientée aractères à longue istance. Ce code est omposé de points t de tirets (une sorte e codage binaire...).
OS ••• — — •••

Le jeu de caractères codés ASCII (*American Standard Code for Information Interchange* « Code américain normalisé pour l'échange d'informations »), inventé par Bob Bemer en 1961, est la norme de codage de caractères en informatique la plus connue, la plus ancienne et la plus largement compatible.

Le code ASCII est un code sur 7 bits (valeurs 0 à 127), il permet de définir :

- des caractères imprimables universels : lettres majuscules et minuscules, chiffres, symboles de ponctuation…
- des codes de contrôle non imprimables : indicateur de saut de ligne, de fin de texte, codes de gestion des échanges, codes de contrôle de périphériques…

	LSQ	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
MSQ		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
0001	1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EN	SUB	ESC	FS	GS	RS	US
0010	2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Fig. 8 Codage ASCII

Le tableau ci-dessus regroupe les 128 éléments du code ASCII. Le 8e bit étant toujours nul, le codage n'utilise bien que les 7 bits de droite d'un octet. Le caractère de code %00100000 ($20) est le caractère « espace » utilisé comme séparateur de mots dans un texte.

Les caractères de contrôle (codes de $00 à $1F et code $7F) ne sont pas des caractères imprimables. Ils ont été choisis initialement pour provoquer une action sur le terminal texte récepteur. Par exemple, le caractère CR signifie *Carriage Return* (retour chariot) par analogie avec les anciennes machines à écrire, il provoque un retour en colonne gauche du périphérique de sortie ; de même, le caractère LF (*Line Feed*) sert à générer un saut de ligne, c'est-à-dire un passage à la ligne suivante… La combinaison CRLF est donc utilisée pour faire un passage en début de ligne suivante lors de l'affichage de texte !

ACTIVITÉ 8 Décrypter la chaîne ASCII suivante représentée sous forme d'une suite d'octets

→ 0101 0011 0101 0100 0100 1001 0010 0000 0011 0010 0100 0100 0000 1101 0000 1010

Chaque langue pourrait utiliser les 128 codes libres (ceux avec le MSB à 1) pour ses propres spécificités… mais le codage ne serait plus universel. Sans compter que le répertoire universel des langues écrites contient de l'ordre de 100 000 caractères, dont plus de la moitié pour les seuls sinogrammes…

ASCII contient les caractères nécessaires pour écrire… en anglais ! Si cela est suffisant pour écrire des programmes, il n'en va pas de même lorsqu'il s'agit de transcrire un texte en français. Comment coder nos lettres accentuées et autres cédilles… ?

b Universalité

Pour coder de manière universelle l'ensemble des symboles utilisés quelle que soit la langue (anglais, français, grec, chinois…), il faut attribuer à tout caractère ou symbole de n'importe quel système d'écriture de langue **un nom et un identifiant numérique**, et ce de manière unifiée, quelle que soit la plate-forme informatique ou le logiciel !

C'est ce que propose la norme **Unicode** (développée par le Consortium du même nom, www.unicode.org).

Chaque symbole d'écriture est représenté par un nom et une valeur hexadécimale préfixée par « U+ ».

EXEMPLE

A	« lettre majuscule latine A »	**U+0041**
é	« lettre minuscule latine E accent aigu »	**U+00E9**
€	« symbole euro »	**U+20AC**

Avec Unicode, le **glyphe** devant représenter chaque caractère n'est pas défini, ceci est du ressort des fontes de caractères utilisées…

Pour stocker sur un support informatique un texte constitué de caractères Unicode, il faut encore choisir un procédé transformant chaque définition Unicode en une suite d'octets et réciproquement… C'est le **processus d'encodage**.

Actuellement, un des systèmes d'encodage couramment utilisés (Unix, Internet…) est **UTF-8** (*Unicode Transformation Format*), spécifié par la RFC 3629. Ce système remplit les conditions suivantes :

- Les caractères faisant partie de l'ASCII sur 7 bits doivent rester inchangés. L'objectif est de permettre un maximum de compatibilité entre les documents UTF-8 et les applications qui ne connaissent que l'ASCII et supposent qu'un caractère correspond à un octet ;
- Les caractères encodés peuvent occuper de 1 à 4 octets. Comme la majorité des caractères font plusieurs octets, il doit être possible d'en retrouver le début ; ceci afin de pouvoir l'utiliser dans un flux de données.

UTF-8 propose pour cela un codage sous forme d'octets dont les bits de poids fort sont discriminants.

Représentation binaire UTF-8				Signification
0xxxxxxx				1 octet codant 1 à 7 bits
110xxxxx	10xxxxxx			2 octets codant 8 à 11 bits
1110xxxx	10xxxxxx	10xxxxxx		3 octets codant 12 à 16 bits
11110xxx	10xxxxxx	10xxxxxx	10xxxxxx	4 octets codant 17 à 21 bits

Fig. 9 Encodage UTF-8

EXEMPLE

On trouve dans la page www.unicode.org/charts/PDF/U0370.pdf l'information suivante :

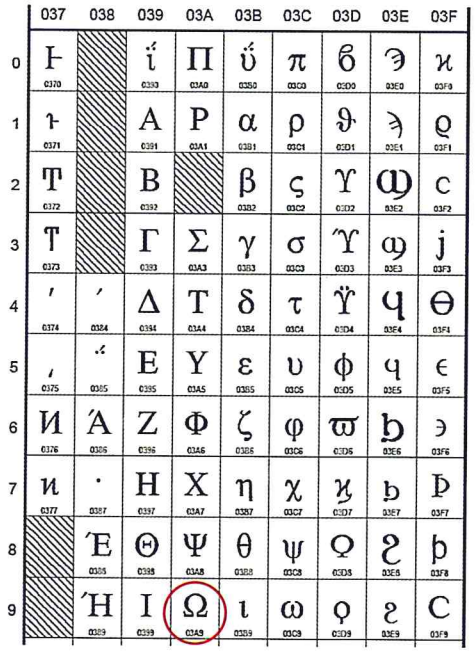
« U+03A9 Ω GREEK CAPITAL LETTER OMEGA ».

Quel est le codage UTF-8 de ce caractère ?

– $03A9 = % 0000.0011.1010.1001, soit 10 bits significatifs

– Dans l'intervalle 8 à 11 bits, encodage sur 2 octets : % **110**01110 **10**101001.

– Soit en hexadécimal : $CEA9.



U0370 « Greek and Coptic » (extrait)

ACTIVITÉ 9 Quel est, en hexadécimal, le résultat de l'encodage UTF-8 des symboles « A », « é » et « € » (le codage Unicode de ces symboles est donné plus haut) ?

ACTIVITÉ 10 En vous aidant des informations disponibles sur le site Web d'Unicode, saurez-vous déchiffrer le message encodé UTF-8 suivant ?

E3 83 AA E3 83 BC E3 83 8C E3 82 AF E3 82 B9

9 Codage des images

Après le texte, l'image est sans nul doute le support le plus utilisé pour communiquer.

Dans le domaine du numérique, les images sont constituées d'une **matrice L x H** (Largeur x Hauteur) de points élémentaires que l'on nomme généralement des pixels (abréviation de *Picture Element*).

Chaque pixel a une **couleur** codée sur un nombre plus ou moins grand de bits. L x H (en pixels) correspond à la **définition** de l'image.

Le périphérique de sortie (écran, imprimante...) se doit de restituer les pixels de manière ordonnée en fonction de leur position respective (x,y) et de leur couleur...

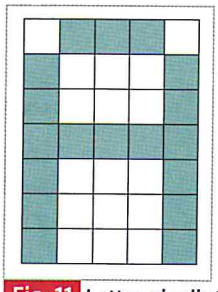


Fig. 11 Lettre pixelisée

EXEMPLE

Le caractère ASCII $41 peut être simplement représenté sur une matrice 5×7 en « allumant » les pixels adéquats. La « couleur » peut être ici matérialisée par un unique bit (pixel allumé ou éteint)...

Le codage d'une image peut donc dans un premier temps se résumer en la succession des codages des pixels suivant un ordre bien défini (balayage lignes-colonnes en partant du coin supérieur gauche).

Les imprimantes (fond blanc) utilisent le système de composition complémentaire **CMJN** (Cyan-Magenta-Jaune-Noir).

a Codage de la couleur

Le mode de représentation RVB (**Rouge, Vert et Bleu**, ou en anglais RGB pour *Red Green Blue*) correspond à celui fourni par la plupart des caméras couleurs, il est naturellement utilisé pour la reproduction de couleurs sur écran (base noire). C'est le mode de composition des couleurs basé sur le principe des **couleurs additives** : le rouge, le vert et le bleu sont les trois primaires utilisés dans la constitution de couleurs à partir de sources lumineuses.

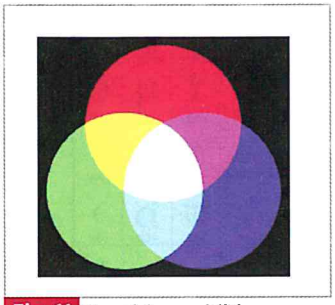


Fig. 11 Synthèse additive

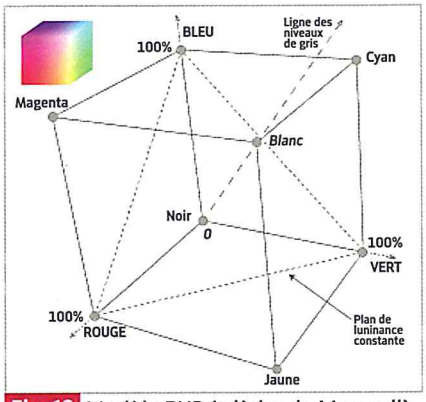


Fig. 12 Modèle RVB (trièdre de Maxwell)

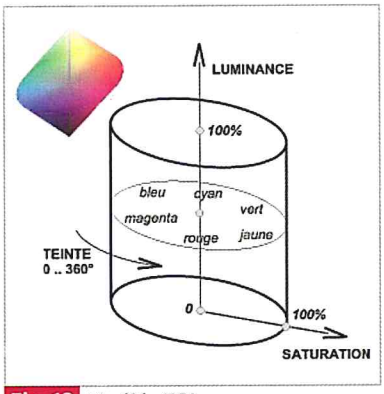


Fig. 13 Modèle TSL

Le **modèle TSL** pour Teinte-Saturation-Luminance (HSI : Hue-Saturation-Intensity), est un autre modèle plus proche de la perception humaine des couleurs. Ses coordonnées se calculent à partir des proportions RVB. La composition de la teinte et de la saturation est appelée **chrominance**.

EXEMPLE

pixel blanc	→	codage RVB = $FFFFFF
pixel noir	→	codage RVB = $000000
pixel rouge	→	codage RVB = $FF0000
pixel vert	→	codage RVB = $00FF00
pixel cyan	→	codage RVB = $00FFFF
pixel jaune	→	codage RVB = $FFFF00
pixel rose	→	codage RVB = $FFB5C5
pixel gris foncé	→	codage RVB = $A9A9A9

ACTIVITÉ 11 Exprimez en hexadécimal les couleurs RVB suivantes (proportions en %), en déduire un nom possible de la couleur : %RVB (100, 65, 0) %RVB (93, 51, 93) %RVB (60, 80, 20).

b Profondeur d'une image

Les images codées ainsi en 24 bits sont dites en vraies couleurs (*true color*) ; une composante Alpha permettant d'inclure une information de transparence peut être ajoutée à ce type de codage, chaque pixel est alors codé sur 32 bits.

Le terme de **profondeur d'image** est utilisé par spécifier le nombre de bits alloué au codage de chaque pixel ; les valeurs courantes de profondeur sont 1 (image binaire), 8 (256 couleurs ou niveaux de gris)... et 32 (vraies couleurs avec canal alpha).

ACTIVITÉ 12 Quelle est la taille (en octets) d'une image non compressée, de définition 640×480 et de profondeur 24 bits ?

L'image est incorporée à un document destiné à être distribué sous forme de photocopies N&B. Quelle économie de taille réalisez-vous en convertissant l'image en 256 niveaux de gris ?

ACTIVITÉ 13 Votre ordinateur affiche sans problème des images de définition 1024×768 et de profondeur 32 bits (RVBA).

Que pouvez-vous en déduire quant à la taille mémoire de votre carte vidéo ?

SYNTHÈSE

Le **codage binaire** (base 2) est la représentation naturelle des grandeurs du monde réel ou virtuel au sein des systèmes numériques.

L'unité de codage est le **bit** (*Binary Digital Unit*) qui ne peut prendre que les 2 valeurs 1 et 0 (ou vrai et faux).

L'unité de transfert est **l'octet** (*byte*) qui est un mot binaire de 8 bits.

En pratique, on utilise des mots binaires de **16 bits** (*word*), **32 bits** (*double word*), **64 bits** (*quad word*). Dans un mot binaire, le bit situé le plus à gauche est le **bit de poids fort** (MSB : *Most Significant Bit*), et le bit le plus à droite est le **bit de poids faible** (LSB : *Less Significant Bit*).

Dans le SI (Système international), les multiples de l'octet sont le kilo-octet (ko), le Méga-octet (Mo), le Giga-octet (Go) et le Téra-octet (To).

Les multiples en base 2 de l'octet sont le kibi-octet (kio), le Mébi-octet (Mio), le Gibi-octet (Gio) et le Tébi-octet (Tio).

Pour un nombre codé en base 2, chaque symbole 0 ou 1 de ce nombre est un multiplicateur d'une puissance de 2. Un nombre codé avec N symboles 0 ou 1 en base 2 peut prendre 2^N valeurs entières différentes.

La **représentation hexadécimale** des nombres binaires en facilite la manipulation. Chaque symbole hexadécimal 0..1,A..F représente un quartet binaire (mot de 4 bits).

Un mot binaire peut être interprété comme une **valeur signée**. Dans ce cas, c'est le bit de poids fort qui joue le rôle de signe.

Un octet permet de coder les valeurs de 0 à 255_{10} en interprétation non signée. L'intervalle de codage en interprétation signée va de -128_{10} à $+127_{10}$. Dans les deux cas, il y a bien 256 valeurs distinctes (2^8).

Le codage binaire des **nombres à virgule flottante** est régi par la norme P754 de l'IEEE ; les encodages courants occupent 32 bits (simple précision) ou 64 bits (double précision).

La table **ASCII** associe un code à 7 bits à un jeu de caractères d'imprimerie : chiffres, lettres majuscules et minuscules, symboles de ponctuation… Ce standard contient aussi le codage de certains caractères non affichables destinés au contrôle des périphériques de sortie.

Le système **Unicode** permet de représenter de manière universelle l'ensemble des symboles utilisés dans la totalité des langages de la planète. **UTF-8** est un système d'encodage spécifiant une valeur binaire discriminante pour chacune des définitions de l'Unicode.

Une image est une matrice de **pixels**. Chaque pixel peut être codé en binaire sous la forme de 3 valeurs spécifiant les proportions des 3 couleurs de base de la synthèse additive : RVB (Rouge-Vert-Bleu).

Ce chapitre a démontré qu'avec un peu d'imagination, on peut coder en binaire n'importe quel objet du monde réel ou virtuel…

Pour être sauvegardées, échangées puis exploitées, les données obtenues sont en général enregistrées sur un support type mémoire de masse (disque dur, clé USB, CD-Rom, …) sous forme de fichiers qui ne sont que des suites linéaires d'octets…

Les applications manipulant les données doivent encore se mettre d'accord quant à leur sémantique (une suite de caractères n'a de sens que si l'on comprend les mots que ces caractères forment…) ; c'est là qu'interviennent notamment les langages informatiques…