Chapitre 7 - Codage nombres, caractères, images, sons, vidéos

On voit dans ce chapitre comment sont numérisés :

- Les nombres entiers relatifs de l'ensemble $\mathbb{Z}$,
- Les nombres réels de l'ensemble $\mathbb{R}$,
- Les caractères : lettres des différentes langues, symboles, …
- Les images, sons et vidéos.

1

2,58

3,14159

2023

Aa Bb Cc Dd Ee Ff
Gg Hh Ii Ji Kk Ll
Mm Nn Oo Pp Qq
Rr Ss Tt Uu Vv Ww
Xx Yy Zz

Аа Бб Вв Гг Дд Ее
Жж Зз Ии Йй Кк
Лл Мм Нн Оо Пп
Рр Сс Тт Уу Фф Хх
Цц Чч Шш Щщ Ъъ
Ьь Юю Яя

1- Comment on numerise des nombres entiers relatifs ?

a. Methode du bit signe :

La première idée qui vient à l'esprit est de réserver le bit le plus à gauche pour identifier le signe du nombre (bit à 0 pour $+$ et à 1 pour $-$) . Par exemple, pour un codage des nombres 57 et -57 , sur 1 octet, on aurait :

Codage de $(+57)_{10}$:	Codage de $(-57)_{10}$:

Si le codage est fait sur 1 octet, comme 1 bit sur les 8 est réservé au signe, seul 7 bits sont exploitables pour numériser la valeur absolue du nombre. Ainsi il ne sera que possible de coder des nombres entre -127 et $+127$.

L'avantage de cette méthode est qu'il est facile de faire la correspondance avec la base 10. Par contre, l'inconvénient apparait lorsque l'on additionne un nombre avec son opposé. Par exemple, l'opération $(+57) + (-57)$ réalisée en binaire donne :

Le résultat de cette opération ne donne donc pas le nombre 0. Comme les nombres sont souvent utilisés pour réaliser des opérations mathématiques, cet inconvénient devient majeur. Cette méthode, dite du « *bit signé* », n'est ainsi pas utilisée.

b. METHODE DU COMPLEMENT A 2 :

On présente cette technique dite du « *complément à 2* », pour un codage sur 1 octet, des nombres *57* et *-57*. Comme pour la méthode précédente, sur 1 octet, il ne sera possible de coder uniquement des nombres compris entre -127 et $+127$.

Codage de $N = (+57)_{10}$:	Codage de $(-57)_{10}$: - Complément à 1 de $(+57)_{10}$ noté $\overline{\overline{N}}$: - Complément à 1 de $(+57)_{10}$ noté $\overline{\overline{N}} + 1$: - Le codage de $(-57)_{10}$ est :

Avec cette méthode, l'opération $(+57) + (-57)$ réalisée en binaire donne :

Si on veut faire la correspondance entre la notation binaire d'un nombre entier relatif et sa notation en base 10, on procède de la manière suivante :

Codage du nombre $(0011\ 1001)_2$ sur 1 octet :	Codage du nombre $(1100\ 0111)_2$ sur 1 octet :

2- COMMENT ON NUMERISE DES NOMBRES REELS ?

a. CAS DES NOMBRES DECIMAUX :

En notation décimal, les chiffres à gauche de la virgule représentent les unités, les dizaines, centaines etc … et ceux à droite de la virgule, les dixièmes, les centièmes etc.. Par exemple, le nombre décimal $526{,}73$ peut se décomposer ainsi :

De la même manière en binaire, les chiffres à droite de la virgule représentent les demis, les quarts, les huitièmes, les seizièmes etc … Par exemple, le nombre décimal $11{,}375$ peut se décomposer ainsi :

On convertit facilement un nombre décimal de la base 2 vers la base 10. Par exemple, la conversion en base 10 du nombre $(111{,}101)_2$ donne :

Par contre, la conversion de la base 10 à la base 2, ne peut pas toujours être réalisée de manière exacte. Par exemple, pour la conversion sur 1 octet du nombre décimal $(0{,}3)_{10}$, on aurait :

2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
1	0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125

Les nombres tels que $0{,}1$ et $0{,}3$ ne sont pas représentables exactement en binaire. Ils ont une représentation tronquée en machine, ce qui introduit des erreurs dans les opérations :

```
>>> 0.2 +0.1
0.30000000000000004
```

b. NOTATION SCIENTIFIQUE

Pour représenter de manière générale les nombres réels on utilise donc une valeur approchée. Cette valeur approchée s'inspire de la notation scientifique. Par exemple la notation scientifique décimale des nombres $53{,}47$ et $-0{,}0382$ est :

En notation scientifique décimale, un nombre est représenté sous la forme : s ***m*** x 10^n où :

- s : est le signe du nombre + ou -
- m est un nombre réel tel que $1 \leq m < 10$ il s'appelle la **mantisse**
- n est un entier relatif il s'appelle **l'exposant**

En notation scientifique binaire un nombre est représenté également sous la forme : s ***m*** x 2^n où :

- s : est le signe du nombre + ou -
- m est un nombre réel tel que $1 \leq m < 2$ il s'appelle la **mantisse**
- n est un entier relatif il s'appelle **l'exposant**

Par exemple, la notation scientifique binaire des nombres suivants est :

- $(+57)_{10} = (0011\ \ 1001\ )_2$
- $(+0{,}3)_{10} = (0{,}010011001\ )_2$

Cette écriture est appelée virgule flottante car la virgule peut flotter de gauche à droite. Un réel est donc représenté en machine par un nombre décimal d'un type particulier : les nombres flottants (floats).

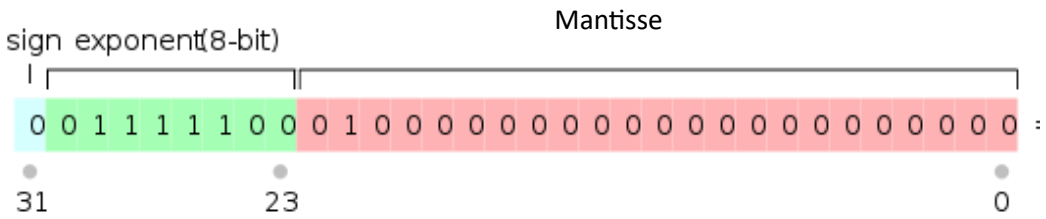
c. REPRESENTATION EN MACHINE :

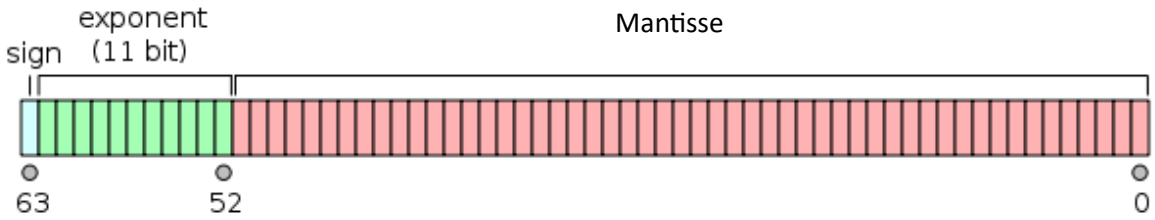
A la fin des années 70, chaque ordinateur possédait sa propre représentation pour les nombres à virgule flottante. En 1985 la norme IEEE 754 est apparue pour normaliser le codage des nombres.

Il existe deux formats associés à cette norme : le format dit "simple précision" et le format dit "double précision". Le format "simple précision" utilise 32 bits pour écrire un nombre flottant alors que le format "double précision" utilise 64 bits. Dans la suite nous travaillerons principalement sur le format 32 bits.

Que cela soit en simple précision ou en double précision, la norme IEEE754 utilise :

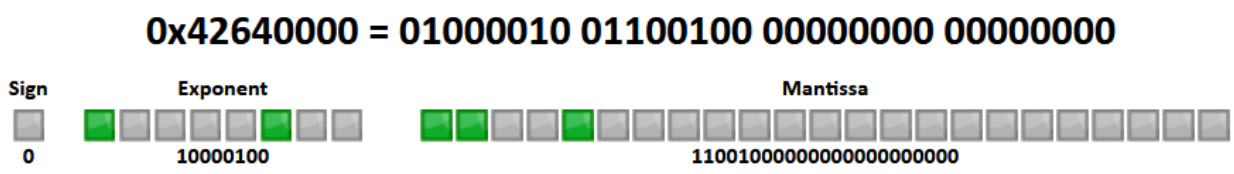
- 1 bit de signe (1 si le nombre est négatif et 0 si le nombre est positif)
- des bits consacrés à l'exposant (8 bits pour la simple précision et 11 bits pour la double précision)
- des bits consacrés à la mantisse (23 bits pour la simple précision et 52 bits pour la double précision)



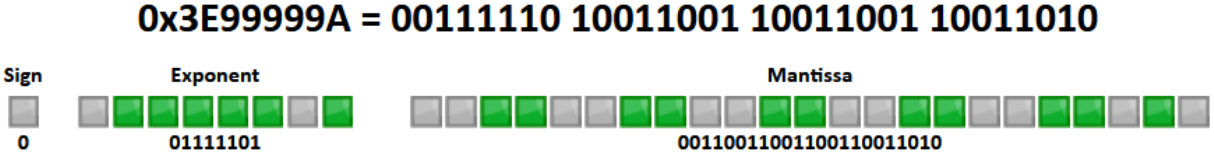


Nous pouvons vérifier que l'on a bien 1 + 8 + 23 = 32 bits pour la simple précision et 1 + 11 + 52 = 64 bits pour la double précision.

⇨ Pour comprendre comment cela fonctionne, on commente la numérisation du nombre $(+57)_{10} = (0011\ 1001\ )_2$:



⇨ On commente de même la conversion du nombre $(+0{,}3)_{10} = (0{,}010011001\ )_2$:



d. Exercice :

Déterminer la représentation au format simple précision de $(32{,}75)_{10}$ en binaire et en hexadécimal :

3- Comment on numerise des caracteres ?

a. Code ASCII :

L'***American Standard Code for Information Interchange*** (Code américain normalisé pour l'échange d'information), plus connu sous l'acronyme **ASCII** ([aski:]), est une norme informatique de codage de caractères apparue dans les années 1960. C'est la norme de codage de caractères la plus influente à ce jour. Le tableau ci-dessous permet de retrouver le codage en hexadécimal de caractères sur 1 octet. L'unité du nombre hexa est donné par la colonne, la ligne donne le second chiffre :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STH	ETH	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	CD2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8	€	□	‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ	□	Ž	□
9	□	'	'	"	"	•	–	—	˜	™	š	›	œ	□	ž	Ÿ
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Par exemple :

- Le caractère 'a' est codé de la manière suivante :
- Le caractère 'A' est codé de la manière suivante :

b. L'UNICODE :

Unicode est un standard informatique qui permet des échanges de textes dans différentes langues, à un niveau mondial. Unicode se contente de recenser, nommer les caractères et leur attribuer un numéro. Mais il ne dit pas comment ils doivent être codés en informatique.
Plusieurs codages des caractères Unicode existent :

- UTF-32 qui code chaque caractère sur 32 bits (soit quatre octets)
- UTF-16 qui code chaque caractère sur 16 ou 32 bits (soit deux ou quatre octets)
- UTF-8 qui code chaque caractère sur 8, 16, 24 ou 32 bits (soit un, deux, trois ou quatre octets).

L' UTF-8 est le plus couramment utilisé, notamment pour les pages Web.

La principale caractéristique d'UTF-8 est qu'elle est rétro-compatible avec le standard ASCII, c'est-à-dire que **tout caractère ASCII se code en UTF-8 sous forme d'un unique octet, identique au code ASCII.** Par exemple « A » (A majuscule) a pour code ASCII $(65)_{10}$. Chaque caractère dont le point de code est supérieur à 127 en base 10 (caractère non ASCII) se code sur 2 à 4 octets. Le caractère « € » (euro) se code par exemple sur 3 octets dont la conversion en base 10 est respectivement $(226)_{10}$, $(130)_{10}$, $(172)_{10}$.

Tableau récapitulatif de la concordance entre les codes UNICODE et UTF-8

Caractères codés	Représentation binaire UTF-8	Premier octet valide (hexadécimal)	Signification
U+0000 à U+007F	0xxxxxxx	00 à 7F	1 octet, codant 7 bits
U+0080 à U+07FF	110xxxxx 10xxxxxx	C2 à DF	2 octets, codant 11 bits
U+0800 à U+0FFF	1110**0000** 10**1**xxxxx 10xxxxxx	E0 (le 2e octet est restreint de A0 à BF)	3 octets, codant 16 bits
U+1000 à U+1FFF	1110**0001** 10xxxxxx 10xxxxxx	E1	
U+2000 à U+3FFF	1110**001x** 10xxxxxx 10xxxxxx	E2 à E3	
U+4000 à U+7FFF	1110**01xx** 10xxxxxx 10xxxxxx	E4 à E7	
U+8000 à U+BFFF	1110**10xx** 10xxxxxx 10xxxxxx	E8 à EB	
U+C000 à U+CFFF	1110**1100** 10xxxxxx 10xxxxxx	EC	
U+D000 à U+D7FF	1110**1101** 10**0**xxxxx 10xxxxxx	ED (le 2e octet est restreint de 80 à 9F)	
U+E000 à U+FFFF	1110**111x** 10xxxxxx 10xxxxxx	EE à EF	
U+10000 à U+1FFFF	11110**000** 10**01**xxxx 10xxxxxx 10xxxxxx	F0 (le 2e octet est restreint de 90 à BF)	4 octets, codant 21 bits
U+20000 à U+3FFFF	11110**000** 10**1**xxxxx 10xxxxxx 10xxxxxx		
U+40000 à U+7FFFF	11110**001** 10xxxxxx 10xxxxxx 10xxxxxx	F1	
U+80000 à U+FFFFF	11110**01x** 10xxxxxx 10xxxxxx 10xxxxxx	F2 à F3	
U+100000 à U+10FFFF	11110**100** 10**00**xxxx 10xxxxxx 10xxxxxx	F4 (le 2e octet est restreint de 80 à 8F)	

Pour avoir une idée du travail accompli pour coder symboles et caractères, on peut aller visiter le site suivant : https://symbl.cc/fr/

Par exemple, pour le caractère arabe ci-contre :

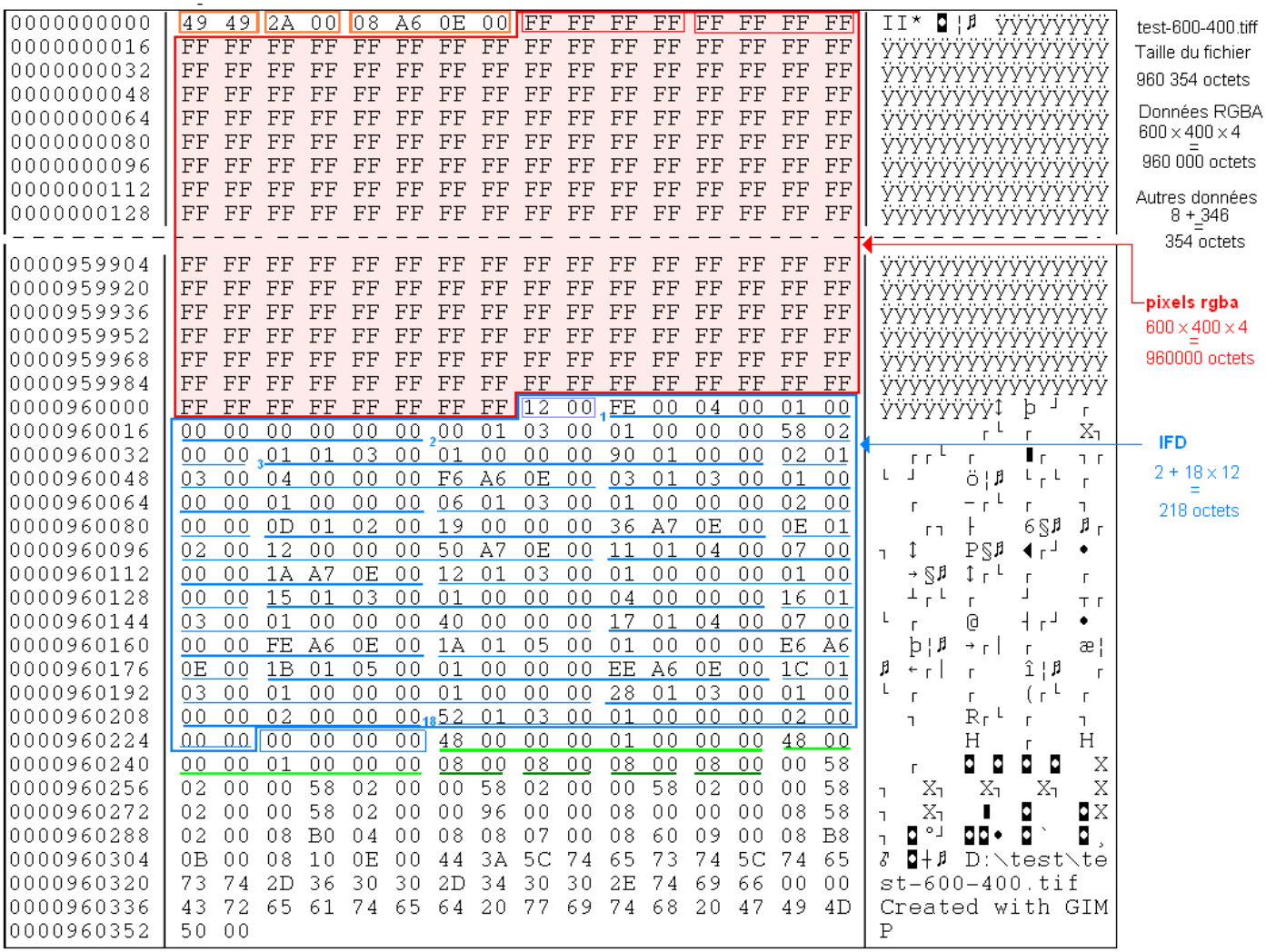
Codage	hex	dec (bytes)	dec	binary
UTF-8	DA 81	218 129	55937	11011010 10000001
UTF-16BE	06 81	6 129	1665	00000110 10000001
UTF-16LE	81 06	129 6	33030	10000001 00000110
UTF-32BE	00 00 06 81	0 0 6 129	1665	00000000 00000000 00000110 10000001
UTF-32LE	81 06 00 00	129 6 0 0	2164654080	10000001 00000110 00000000 00000000

4- COMMENT ON NUMERISE DES IMAGES ?

a. IMAGE AU FORMAT TIFF :

Le format TIFF non compressé est pratique pour manipuler aisément et efficacement les images à l'aide d'un langage de programmation. Ce format permet de gérer la transparence des images tout en assurant une qualité d'image maximale.

Pour un fichier au format TIFF, contenant une image blanche de 600 px de large et 400 px de haut, à laquelle on a ajouté un canal α, on retrouve le contenu binaire suivant, converti en hexadécimal :



Le fichier TIFF de cette image a une taille totale de 980 354 octets, soit 0,98 Mo. Parmi ces octets, on retrouve ceux qui permettent de numériser les différents pixels :

- Nombre de pixels :
- Nombre d'octets pour définir la couleur et le degré de transparence de chaque pixel (*pixels définis en rgba()*) :
- Nombre d'octets pour définir tous les pixels :
- Proportion en % par rapport à la taille totale :

b. IMAGE AU FORMAT JPEG :

Le format JPEG est un des formats les plus utilisés, notamment dans le web, car il présente le gros avantage de pouvoir réduire significativement la taille des fichiers. Les algorithmes de compression utilisés entrainent par contre une perte de qualité. D'autre part, la transparence des images n'est pas prise en compte.

c. IMAGE AU FORMAT PNG :

Le PNG est un format qui permet également une compression, mais ici sans perte. Les images conservent ainsi leur qualité d'origine. Par contre, la taille des fichiers est plus importante qu'avec une compression au format jpeg.
Ce format est surtout utilisé dans le web, car il gère la transparence des pixels à travers un canal alpha.

d. IMAGE AU FORMAT SVG :

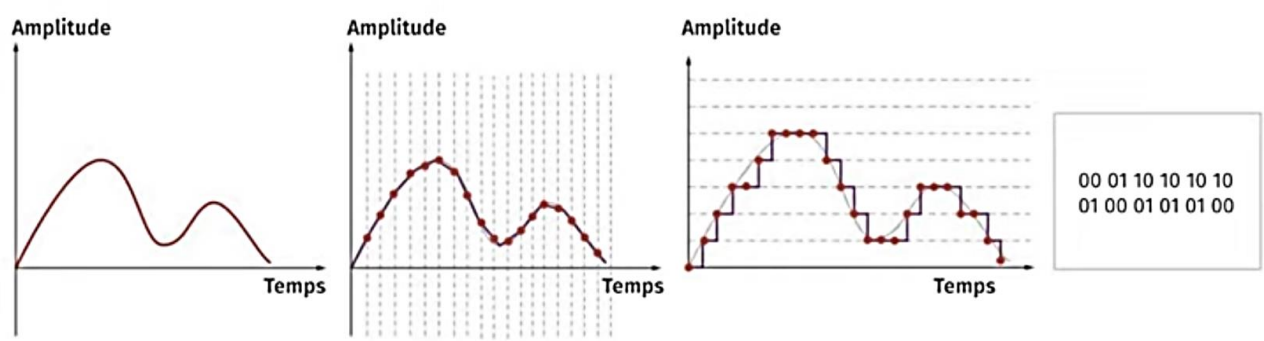
Le SVG permet de sauvegarder et d'afficher des images vectorielles.
Une image vectorielle est une image numérique composée de plusieurs objets géométriques individuels (droites, polygones, arcs de cercle). L'image vectorielle est créée à partir d'équations mathématiques. Chaque forme dépend de plusieurs paramètres (hauteur, largeur, rayon) donnés à des vecteurs.

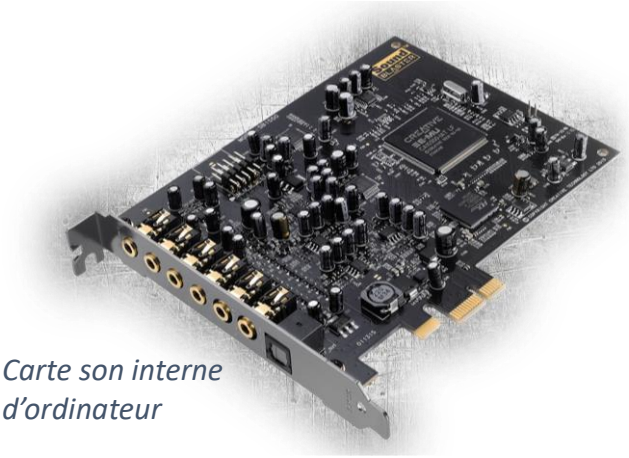
Ce format est réservé à des formes géométriques simples. La taille du fichier est très petite. Le gros avantage est de pouvoir réaliser des zooms de l'image sans aucune perte de qualité. Contrairement aux autres formats, on peut ici zoomer autant que l'on veut, sans voir le rendu se dégrader.

5- COMMENT ON NUMERISE DES SONS ?

a. SON AU FORMAT WAV :

Une onde acoustique correspond à une variation de pression qui se propage dans l'air à une vitesse de 340 m/s. Lorsqu'elle arrive dans l'oreille, elle fait vibrer les tympans, les osselets et les terminaisons nerveuses qui y sont accolées. Un microphone permet transformer le profil de l'onde acoustique en un signal électrique compris entre -5 V et +5 V. La carte son des ordinateurs permet d'échantillonner ce signal avec une fréquence d'échantillonnage de 44 100 Hz. C'est-à-dire que pour numériser 1 seconde d'un son, on aura 44 100 valeurs de la tension du signal.





Carte son interne d'ordinateur

Dans un fichier au format Wav de qualité CD, le son est échantillonné avec une fréquence de 44 100 Hz et le niveau du signal est numérisé sur 2 octets, en stéréo, c'est à dire pour le canal droit et pareil pour le canal gauche.
Ainsi, pour numériser 1 seconde, le nombre d'octets nécessaire est de :

Dans un fichier au format Wav, le flux de données est enregistré sans compression. Un entête de 44 octets est écrit en début de fichier, afin de préciser à l'appareil qui lira ce fichier, les caractéristiques de ce flux. On présente ci-dessous l'organisation de cet entête :

```
[Bloc de déclaration d'un fichier au format WAVE]
   FileTypeBlocID  (4 octets) : Constante « RIFF »  (0x52,0x49,0x46,0x46)
   FileSize        (4 octets) : Taille du fichier moins 8 octets
   FileFormatID    (4 octets) : Format = « WAVE »  (0x57,0x41,0x56,0x45)

[Bloc décrivant le format audio]
   FormatBlocID    (4 octets) : Identifiant « fmt␣ »  (0x66,0x6D, 0x74,0x20)
   BlocSize        (4 octets) : Nombre d'octets du bloc - 16  (0x10)

   AudioFormat     (2 octets) : Format du stockage dans le fichier (1: PCM entier, 3: PCM flottant, 65534:
WAVE_FORMAT_EXTENSIBLE)
   NbrCanaux       (2 octets) : Nombre de canaux (de 1 à 6, cf. ci-dessous)
   Frequence       (4 octets) : Fréquence d'échantillonnage (en hertz) [Valeurs standardisées : 11 025, 22
050, 44 100 et éventuellement 48 000 et 96 000]
   BytePerSec      (4 octets) : Nombre d'octets à lire par seconde (c.-à-d., Frequence * BytePerBloc).
   BytePerBloc     (2 octets) : Nombre d'octets par bloc d'échantillonnage (c.-à-d., tous canaux confondus :
NbrCanaux * BitsPerSample/8).
   BitsPerSample   (2 octets) : Nombre de bits utilisés pour le codage de chaque échantillon (8, 16, 24)

[Bloc des données]
   DataBlocID      (4 octets) : Constante « data »  (0x64,0x61,0x74,0x61)
   DataSize        (4 octets) : Nombre d'octets des données (c.-à-d. "Data[]", c.-à-d. taille_du_fichier -
taille_de_l'entête  (qui fait 44 octets normalement).
   DATAS[] : [Octets du  Sample 1 du Canal 1] [Octets du Sample 1 du Canal 2] [Octets du Sample 2 du Canal 1]
[Octets du Sample 2 du Canal 2]
```

b. SON AU FORMAT MP3 :

Il s'agit d'un format de compression audio avec perte. Il permet une réduction importante de la taille du flux de données audio, tout en conservant une qualité de restitution acceptable.
C'est aussi l'un des formats de musique numérique les plus répandus.

6- COMMENT ON NUMERISE DES VIDEOS ?

a. CONTENEUR ET CODEC :

Dans le domaine du multimédia, il est important de bien séparer le rôle des *conteneurs* de celui des *codecs* :

- Un **conteneur** permet de stocker des flux vidéo et audio liés selon une séquence précise.
- Un **codec** permet d'encoder (COmpression) et de décoder (DECompression) ces flux.

Un logiciel capable de reconnaître et d'ouvrir un conteneur pourra accéder aux flux, mais ne pourra les décoder que s'il dispose également des codecs appropriés à chacun.

Le conteneur peut donc être vu comme une boîte (le *contenant*), et les flux comme ce que l'on met à l'intérieur de la boîte (le *contenu*).

Un conteneur vidéo permet de rassembler en un seul fichier :

- un ou plusieurs flux vidéo (ce qui permet par exemple de regarder une scène filmée sous plusieurs angles différents) qui correspond à une succession d'images : 24 images par secondes au minimum (24 Frame Per Second , fps), voire 30 fps ou 120 fps. Avec une valeur en fps élevée, il sera possible de réaliser des ralentis importants.
- un ou plusieurs flux audio (ce qui permet d'obtenir une version multilingue du média) ;
- des sous-titres (ce qui permet également le multilingue) ;
- des éléments de chapitrage (de la même manière que sur les DVD) ;
- des métadonnées (par exemple le titre du média, le nom du réalisateur, la date, etc.) ;
- une description des flux que contient le conteneur ;
- éventuellement d'autres données.

b. Fichier au format MP4 :

Le format de fichier MP4 fait depuis longtemps partie des formats vidéo les plus prisés sur le Net. On peut obtenir d'excellents compromis si on combine une bonne compression avec une bonne qualité du fichier vidéo.

Les avantages
Ce format propose un bon compromis entre qualité de l'image et compression du fichier. Il présente avant tout l'avantage de pouvoir **réunir dans un seul format plusieurs pistes audio et d'autres éléments**.

Les inconvénients
Avec les codecs par défaut, les fichiers MP4 sont généralement **fortement comprimés**, ce qui rend leur traitement et leur création parfois difficiles.

c. Fichier au format AVI :

L'abréviation AVI signifie Audio Video Interleave. Ces mots indiquent que les pistes vidéo et audio sont imbriquées les unes dans les autres. Ce format est très répandu, bien qu'un peu dépassé sur le plan technique.

Les avantages
Ce format présente surtout l'avantage d'être **accepté par beaucoup de systèmes**. On ne trouve quasiment aucun lecteur ou navigateur qui ne sache lire des fichiers au format AVI.

Les inconvénients
Les limites techniques de ce format de fichier, et en particulier son **incapacité à prendre en compte plusieurs pistes audio**, ont contribué à faire baisser le nombre d'utilisateurs du format AVI.

On n'a pas la place ici de présenter tous les formats existants (.mkv , .mov , .ogg , .wmv).

7- Exercices :

a. Exercice 1 : Complement a 2

1- Utiliser la méthode du complément à 2 pour convertir en binaire les nombres relatifs suivants : $a = (-33)_{10}$; $b = (+100)_{10}$
2- En effectuant la somme en base 2, donner l'écriture binaire du nombre $c = a + b$
3- Convertir en base 10, le nombre c . Retrouve-t-on le bon résultat ?

b. Exercice 2 : Conversion nombres decimaux

1- Convertir les nombres suivants en base 10 : $a = (111{,}10101)_2$; $b = (10011{,}001)_2$
2- Convertir les nombres suivants en base 2 : $c = (25{,}375)_{10}$; $d = (0{,}125)_2$

c. Exercice 3 : Codage des nombres decimaux en machine

1- Déterminer la représentation au format simple précision de $c = (-25{,}375)_{10}$
2- Déterminer la représentation au format simple précision de $d = (-0{,}125)_{10}$

d. Exercice 4 : Codage ASCII

1- Donner en hexa et en binaire le codage des lettres 'd' et 'D'.
2- Donner en hexa et en binaire le codage des symboles '#' et '%'.

e. Exercice 5 : Codage d'une image

1- Une photo a une taille de 1200 px de large et 900 px de haut. Elle est enregistrée au format .tiff sans compression, avec un canal alpha. Quelle est la taille minimale de ce fichier.

2- Même question pour la même photo, mais dont la largeur et la hauteur ont été réduites de moitié : 600 px de large et 450 px de haut.

f. Exercice 6 : Codage d'un son

Une musique est échantillonnée avec une fréquence de 44 100 Hz et le niveau du signal est numérisé sur 4 octets, en stéréo, c'est à dire pour le canal droit et pareil pour le canal gauche. Cette musique a une durée de 3 mn et 30 s. Elle est enregistré au format Wav sans compression. Quelle est la taille minimale de ce fichier ?