2.2 Représentation des nombres entiers avec signe (signés, « signed »)

Représentation usuelles des entiers signés

- **Avec bit de signe**
- **Avec décalage**
- Décimal codé binaire (BCD = Binary Coded Decimal)
- Complément à 1
- **Complément à 2**

Représentation usuelle des entiers signés

- **Avec bit de signe**

  - **Le signe = information binaire (+ ou -)**
    → possibilité de représentation par 1 bit : 0 ou 1

  ↓

  - **Le nombre de bits réservés au nombre à représenter est raccourci de 1 bit**

  ↓

  - **Avec n bits, seuls (n-1) bits peuvent être utilisés pour représenter la valeur absolue du nombre**

  ↓

  - **Sur n bits, on peut représenter de $(-2^{n-1}-1)$ à $(2^{n-1}-1)$**

Représentation usuelle des entiers signés

- Avec bit de signe

- **Problème : on peut <u>représenter le zéro</u> de deux manières : + 0 et - 0 !**
- **Exemple sur un octet (n = 8) : deux représentation possibles**

0000 0000
1000 0000

↓

X +(-X) ne donne pas la représentation du zéro !

Représentation usuelle des entiers signés

- **Avec décalage**

  - **Sur n bits, on représente le 0 au milieu de la pleine échelle**
  - **Exemple sur 1 octet (8 bits) :**

Valeur	Représentation
-128	0000 0000
-127	0000 0001
….	
0	1000 0000
1	1000 0001
…	
127	1111 1111

Représentation usuelle des entiers signés

Valeur	Représentation
-128	0000 0000
-127	0000 0001
….	
0	1000 0000
1	1000 0001
…	
127	1111 1111

- Avec décalage

- **Sur n bits, on peut représenter de (-2^{n-1}) à $(2^{n-1}-1)$:**
- **Si le bit de poids fort est égal à 0, le nombre est négatif**
- **Si le bit de poids fort est égal à 1, le nombre est positif**

- **Pour représenter (« coder ») le nombre X sur n bits:**
  il suffit de représenter $X+ 2^{n-1}$ comme un nombre positif

- **Pour «décoder » un nombre codé sur n bits :**
  on décode le nombre comme un nombre positif,
  puis on lui soustrait 2^{n-1}

Représentation usuelle des entiers signés

- **Complément à 2**
  - En fait, il s'agit du complément à 2^n de X : $X_{cpl} = 2^n - X$
  - Pour représenter – X sur n bits :
    - On calcule le complément à 1 de X
    - On ajoute 1
  - Exemple sur 8 bits :
    - Représentation de -11

11 :	0 0 0 0 1 0 1 1
Complément à 1 :	1 1 1 1 0 1 0 0
+ 1 :	+ 1
	1 1 1 1 0 1 0 1

Représentation usuelle des entiers signés

- **Intérêts du complément à 2**
  - **Les opérations arithmétiques fonctionnent normalement**
  - **Représentation « normale » du zéro : 0 0 0 0 0 0 0 0**

- **Il s'agit de la représentation des entiers signés la plus répandue**

- **Sur n bits, on peut représenter l'intervalle $[-2^{n-1}; 2^{n-1}-1]$**

Représentation usuelle des entiers signés

- **Décodage du complément à 2**
  - Le complément à 2 du complément à 2 donne le nombre initial : $X = 2^n - (2^n - X)$ → on fait le complément à 1 et on rajoute 1

  - **Exemple :**

    Que représente le nombre $1\,0\,1\,1\,0\,1\,1\,0_2$?
  - C'est un nombre négatif car le bit le + significatif = 1
  - Complément à 1 de $1\,0\,1\,1\,0\,1\,1\,0$

    $0\,1\,0\,0\,1\,0\,0\,1$
  - On ajoute 1 $+ \quad 1$

    $0\,1\,0\,0\,1\,0\,1\,0 = 74_{10}$
  - Donc, $1\,0\,1\,1\,0\,1\,1\,0_2 = -74_{10}$

Représentation usuelle des entiers signés

- Complément à 2

Changement de taille :

- **Dans le sens entier taille n → entier taille m > n**
  - **Si nombre positif : on rajoute des 0**
  - **Si nombre négatif : on rajoute des 1**

- **Dans le sens entier taille n → entier taille m < n :**
  - **Cela ne fonctionne que si le nombre est suffisamment petit…**

Représentation usuelle des entiers signés

- **Décimal Codé Binaire (Binary Coded Decimal)**
  - Chaque chiffre est codé indépendamment
  - Les chiffres allant de 0 à 9, il faut 4 bits
  - Le signe est également codé sur 4 bits (à droite) :
    - 1100 pour le +
    - 1101 pour le -
  - Utilisation : afficheurs 7 segments

2.3 Représentation des nombres non entiers

Nombres décimaux

- **Nombre décimal = nombre qui s'écrit avec une quantité quelconque, mais finie, de chiffres derrière la virgule en base 10**
- **Exemples : 10 ; -15,3 ; 1,666**
- **Remarque :** 1,6666… n'est pas un nombre décimal car il n'a pas un nombre fini de chiffres derrière la virgule ! C'est un nombre réel non entier.

- Nombres décimaux entiers : représentation selon les manières vues précédemment (base 2/16 avec codage du signe ou pas…)
- **Nombres décimaux ou réels non entiers : 2 types de représentation :**
  - **Représentation dite « en virgule fixe » (« Fixed Point »)**
  - **Représentation dite « en virgule flottante » (« Floating Point »)**

Nombres décimaux non entiers

- **Représentation en virgule fixe (« Fixed Point »)**

- **On représente séparément la partie entière et la partie décimale**
- **Exemple :** Représentation d'un nombre **sur n bits** :
  - **Partie entière sur m bits (puissances positives de 2)**
  - **Partie décimale sur k bits (puissance négatives de 2)**
  - **n = m + k**

$$a_{m-1}a_{m-2}\ldots a_1a_0,a_{-1}a_{-2}\ldots a_{-k} = a_{m-1}\times 2^{m-1} + a_{m-2}\times 2^{m-2} + \ldots + a_1\times 2 + a_0 + a_{-1}\times 2^{-1} + \ldots + a_{-k}\times 2^{-k}$$

Nombres décimaux non entiers

- **Représentation en virgule fixe (« Fixed Point »)**

- **Pratiquement : comment faire pour convertir ?**

- **Méthode 1 :**

  En utilisant la table des multiples de 2 négatifs

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125	0,00390625

Nombres décimaux non entiers

- **Représentation en virgule fixe (« Fixed Point »)**
- **Méthode 1 :** En utilisant la table des multiples de 2 négatifs

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125	0,00390625

- **Exemple : représenter 14,42 sur 2 octets (m = 8, k = 8)**
- **14,42 = 14 + 0,42**
- **$14_{10} = (0\ 0\ 0\ 0\ 1\ 1\ 1\ 0)_2$**
- **$0,42_{10}$ = 0,25 + 0,125 + 0,03125 + 0,007825 + 0,00390625**
  $= (0,\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1)_2$
- **Donc $14,42_{10} = (0\ 0\ 0\ 0\ 1\ 1\ 1\ 0,\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1)_2$**

Nombres décimaux non entiers

- Représentation en virgule fixe (« Fixed Point »)
- **Autre méthode pour représenter facilement la partie décimale: par multiplications successives de la partie décimale**
- **Exemple pour $0,42_{10}$ sur 8 bits :**

0,42 * 2 = **0**,84
0,84 * 2 = **1**,68
0,68 * 2 = **1**,36
0,36 * 2 = **0**,72
0,72 * 2 = **1**,44
0,44 * 2 = **0**,88
0,88 * 2 = **1**,76
0,76 * 2 = **1**,52

- **Donc, $0,42_{10} = (0, 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1)_2$**

Nombres décimaux non entiers

- Représentation en virgule fixe (« Fixed Point »)
- **Intervalle de représentation : $[-2^{(n-k)-1} ; 2^{(n-k)-1}-1]$**
- **Précision : $\pm 2^{-k}/2$**
- **Donc, pour n fixe :** n (nb total bits) = m (nb bits partie entière) + k (nb bits partie décimale)
  - Si l'on souhaite une grande précision
    → il faut k grand, donc m petit
    → on ne peut l'avoir que pour des nombres petits
  - Si l'on veut convertir un nombre grand
    → il faut m grand → k sera petit
    → précision faible
- Question : comment faire pour avoir une même précision relative pour tous les nb. ?

Nombres réels

- **Généralités sur la représentation des réels**

- **Il est impossible représenter tous les réels car :**
  - **Il existe une infinité de nombre réels**
  - **Un nombre réel non entier peut avoir une infinité de chiffres après la virgule**

- **Exemple :**

  1/3 = 0,33333….

  $= 0,1 + 3\times10^{-1} + 3\times10^{-2} + 3\times10^{-3} + \ldots$

Nombres réels

- Généralités sur la représentation des réels
- **On ne peut donc représenter un nombre réel qu'avec une erreur plus ou moins grande**
- **Rappels :**

$$Erreur\ absolue = Valeur\ exacte - Valeur\ representee$$

$$Erreur\ relative = \frac{Valeur\ exacte - Valeur\ representee}{Valeur\ exacte}$$

$$Erreur\ relative[\%] = \frac{Valeur\ exacte - Valeur\ representee}{Valeur\ exacte} \times 100$$

Nombres réels

- **Ecriture scientifique = représentation d'un nombre réel sous la forme $\pm a \times 10^n$**
  - **a = mantisse = nb. décimal $\in [1, 10)$**
  - **n = exposant = nb. entier**

- **Exemple : $12456{,}35 = +1{,}245635 \times 10^4$**

  Signe : +

  Mantisse : 1,2345635

  Exposant : 4

Nombres réels

- **Ecriture scientifique** $\pm a \times 10^n$
- Exemple : $12456{,}35 = +1{,}245635 \times 10^4$

- **Il n'y a donc qu'un seul chiffre non nul à gauche de la virgule, puis un nombre variable de décimales qui dépend de la précision**

- **Intérêt :**
  - **Connaître rapidement l'ordre de grandeur du nombre**
  - **Simplifier les multiplications (divisions) en procédant à des multiplications (divisions) des mantisses et à des additions (soustractions) des exposants**

Nombres réels

- **Représentation en virgule flottante (« Floating Point ») – Norme IEEE 754 = Représentation sur 32 ou 64 bits d'un nombre réel en écriture scientifique, dont :**
  - **1 bit pour le signe (0 = nb +, 1 = nb -)**
  - **8 ou 11 bits pour l'exposant**
  - **23 ou 52 bits pour la mantisse**
- **Intérêt : avoir la même précision relative pour tous les nombres, du plus petit au plus grand**

Nombres réels

- **Représentation en virgule flottante (« Floating Point »)**

- **Décalage de l'exposant :**

  - **L'exposant peut être + ou - !!!**
  - **Pour éviter sa représentation en complément à 2, l'exposant est décalé, ce qui lui permet d'être représenté sous la forme d'un nombre non signé**
  - **Le décalage est de $2^{e-1}-1$, avec e = nb. de bits de l'exposant**
  - **Exemple : exposant représenté sur 8 bits (e = 8) → décalage de $2^{8-1}-1 = 127$**

Nombres réels

- Représentation en virgule flottante (« Floating Point »)

- **Interprétation d'un nombre représenté en virgule flottante selon la norme IEEE 754 (décodage) :**

$$\text{Valeur} = \text{signe} \times 1,\text{mantisse} \times 2^{(\text{exposant} - \text{décalage})}$$

$$\text{Signe} = \pm 1$$

$$\text{Décalage} = 2^{(\text{nb bits exposant}-1)} - 1$$

Nombres réels

- Représentation en virgule flottante (« Floating Point »)
- Interprétation d'un nombre représenté en virgule flottante selon la norme IEEE 754 (décodage)
- **Exemple sur 32 bits** (1 bit signe + 8 bits exposant + 23 bits mantisse)

  signe exposant mantisse

  0 10000001 11010000000000000000000
- **Signe = 0 → nb positif**
- **Exposant : 10000001 = 128 + 1 = 129**
- **Remarque : exposant représenté sur 8 bits**
  → l'exposant est en fait décalé de $2^{8-1}-1 = 127$
- **Mantisse : 11010000000000000000000 = 1/2 + 1/4 + 1/16 = 0,8125**
- **Donc valeur : + 1,8125×$2^{129-127}$ = + 1,8125×2^2 = 7,25**

Nombres réels

- Représentation en virgule flottante (« Floating Point »)

- **Représentation (« codage ») d'un nombre en virgule flottante**

1) **On détermine le signe du nombre**
2) **On représente le nombre réel en représentation binaire, en virgule fixe**
3) **On compte le décalage nécessaire pour qu'il ne reste qu'un seul chiffre à gauche de l'exposant**
4) **On calcule le codage de l'exposant = décalage + 127 (si exposant sur 8 bits)**

   ou = décalage + 1023 (si exposant sur 11 bits)
5) **On recopie la mantisse, en enlevant le « 1 »**

Nombres réels

- Représentation en virgule flottante (« Floating Point »)
- Représentation (« codage ») d'un nombre en virgule flottante
- **Exemple 1 : représenter -12,5 en virgule flottante sur 32 bits**
  (1 bit signe + 8 bits exposant + 23 bits mantisse)
  - Bit de signe : 1
  - 12,5 = 8 + 4 + 0,5

    $= 1*2^3 + 1*2^2 + 0*2^1 + 0*2^0 + 1*2^{-1}$

    Soit 1 1 0 0 , 1 0 0 0 0 0 mantisse

    décalage de 3

    Il faut donc décaler la virgule de 3 positions vers la gauche => décalage = + 3
  - Exposant : 127 + 3 = 130 soit 10000010
  - Mantisse : 10010000000000000000000 le bourre le reste de 0

Nombres réels

- **Représentation (« codage ») d'un nombre en virgule flottante**
- **Exemple 2 : représenter 1/3 = 0,333333…**
  en virgule flottante sur 32 bits (1 bit signe + 8 bits exposant + 23 bits mantisse)
  - Bit de signe : 0
  - 0,333 = ¼ + 1/16 + 1/64 + 1/256 + …

    $= 1*2^{-2} + 1*2^{-4} + 1*2^{-6} + 1*2^{-8} + \ldots$

  Soit 0 , 0 1 0 1 0 1 0 1… mantisse

  décalage de la virgule après le 1er "1"

  Il faut donc décaler de 2 vers la droite => - 2
  - Exposant : 127 - 2 = 125 soit 01111101
  - Mantisse : 01010101010101010101010

Nombres réels

- Représentation en virgule flottante (« Floating Point »)

• Valeurs particulières

- **Les valeurs où tous les chiffres de l'exposant sont à 0 ou à 1 servent à représenter des nombres particuliers :**
  - **Exposant = 0 et mantisse = 0 → nombre = 0**
  - **Exposant = 11111111 et mantisse = 0 → nombre ∞**
  - **Exposant = 11111111 et mantisse ≠ 0 → Not a Number (NaN)**
  - **Exposant = 0 et mantisse ≠ 0 → nombre dénormalisé (valeur absolue très petite – on abandonne ici la notation scientifique :** $valeur = signe \times mantisse \times 2^{-décalage + 1}$ $= signe \times mantisse \times 2^{-126}$**)**

Nombres réels

- Représentation en virgule flottante (« Floating Point »)

- **Précision en représentation sur 32 bits (« simple precision ») :**
  23 bits de mantisse ←→ 7 chiffres décimaux
- **Exemple : si l'on souhaite représenter π**
  π = 3,14159265358979323846264338327 95.....
  - **On peut le représenter soit :**
    - **Par 3,141592 (en tronquant)**
    - **Par 3,141593 (en arrondissant)**

- **Pour une précision + grande → représentation sur 64 bits (dont 52 bits de mantisse) = « double precision »**

Nombres réels

- **IEEE 754 standard : simple & double precision floating point representations**

	Simple précision	Double précision
Bit de signe	1	1
Bit d'exposant	8	11
Bit de mantisse	23	52
Nombre total de bits	32	64
Codage de 1'exposant	Excédant l27	Excédant l023
Variation de 1'exposant	-126 à +127	-1022 à +1023
Plus petit nombre normalisé	2^{-126}	2^{-1022}
Plus grand nombre normalisé	Environ 2^{+128}	Environ 2^{+1024}
Echelle des nombre décimaux	Environ 10^{-38} à 10^{+38}	Environ 10^{-308} à 10^{+308}
Plus petit nombre dénormalisé	Environ 10^{-45}	Environ 10^{-324}

Nombres réels

- **IEEE 754 standard : simple precision (32 bits)**

- **Plus petit nombre :**
  1 00000001 00000000000000000000000
- **Plus grand nombre :**
  0 11111110 11111111111111111111111
- **Echelle** : de 2^{-126} à 2^{128}
- **Plus petit écart** : mantisse = 1 → 2^{-24}

2.4 Représentation des caractères

(valeurs dites « alphanumériques »)

Caractères

- Dans l'alphabet français, il y a 26 lettres
  → il faut 5 bits pour les « coder »
- Il faut également rajouter les majuscules, minuscules + ponctuation → une centaine de caractères
  → il faut au moins 7 bits

- On fait correspondre à chaque caractère un nombre : code **ASCII** pour **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
- Avec 7 bits, on peut représenter jusqu'à 128 caractères
  = table ASCII de base
- En pratique, on utilise un codage sur 1 octet (8 bits)
  = table ASCII étendue

Caractères

- **Table ASCII de base**

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	Y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Caractères

- **Table ASCII étendue**

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	ÿ	168	¿	184	╕	201	╔	217	┘	233	Θ	249	∙
137	ë	153	Ö	169	⌐	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	ⁿ
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	φ	253	²
141	ì	158	₧	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

- ***L'apparence exacte des caractères dépend du périphérique utilisé (type d'écran, d'imprimante etc)***

Caractères

- **Exemple :**

Données présentes sur un disque dur

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000000000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000001B0	00	00	00	00	00	00	00	00	85	98	9C	0B	00	00	00	00
00000001C0	02	00	EE	FF	FF	FF	01	00	00	00	FF	FF	FF	FF	00	00
00000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA
0000000200	45	46	49	20	50	41	52	54	00	00	01	00	5C	00	00	00
0000000210	FF	5F	FC	CD	00	00	00	00	01	00	00	00	00	00	00	00
0000000220	AF	12	9E	3B	00	00	00	00	22	00	00	00	00	00	00	00
0000000230	8E	12	9E	3B	00	00	00	00	7D	A7	B4	9D	8E	41	47	49
0000000240	91	47	BC	3C	14	EC	3B	32	02	00	00	00	00	00	00	00
0000000250	80	00	00	00	80	00	00	00	CB	78	D9	E3	00	00	00	00
0000000260	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000270	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000280	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000290	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000002A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Caractères

- **Si on veut représenter des caractères chinois, arabes, russes… on est vite à court de place…**

↓

- **Unicode : un code standard (sorte de ASCII sur 16 bits) pour couvrir toutes les langues du monde**

- **Lorsque ça ne suffit pas (ex. : symboles mathématiques)**
  → en mode graphique, on utilise des « polices » de caractères
  = tables faisant correspondre à une valeur du code, pour chaque police, une « image » du caractère
- **Le même code correspond donc à un caractère différent en fonction de la police utilisée → possibilité d'utiliser une infinité de caractères en définissant différentes polices**

2.5 Représentation du son

Son

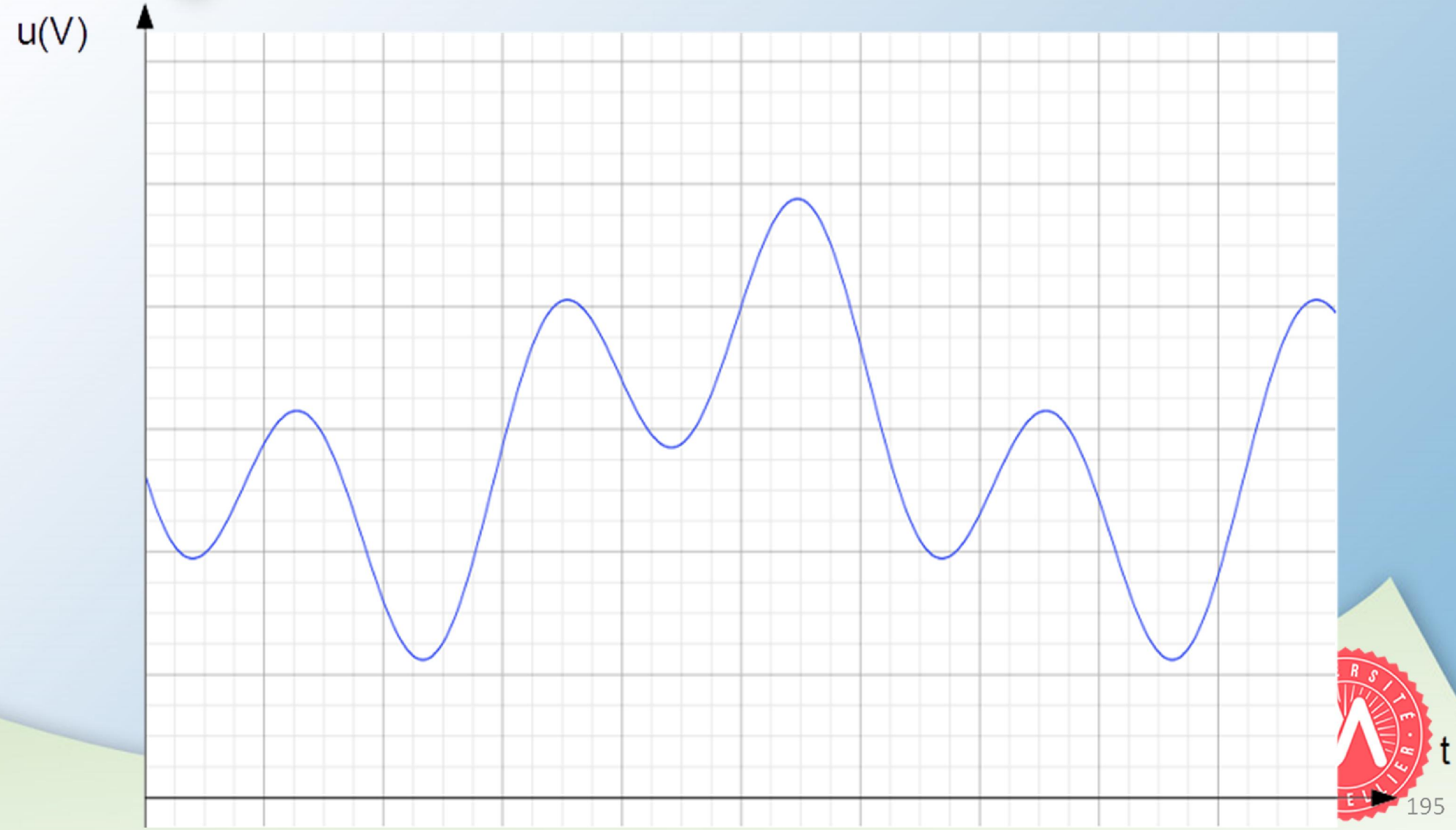
- Comme pour les nombres réels, on ne peut pas représenter tous les sons présents dans la nature car il y en a une infinité
- On ne peut pas non plus représenter tous les niveaux sonores… pour les mêmes raisons
- Le son est donc un signal analogique (continu)
- Enfin, les espaces de stockage sont limités

↓

- Avant d'enregistrer le son, on procède à sa conversion en un signal électrique analogique, qui subit ensuite un « échantillonnage » = conversion analogique/numérique (CA/N)
  - Dans le temps → période d'échantillonnage
  - Au niveau de la valeur (écart entre deux valeurs successives de l'intensité sonore → résolution

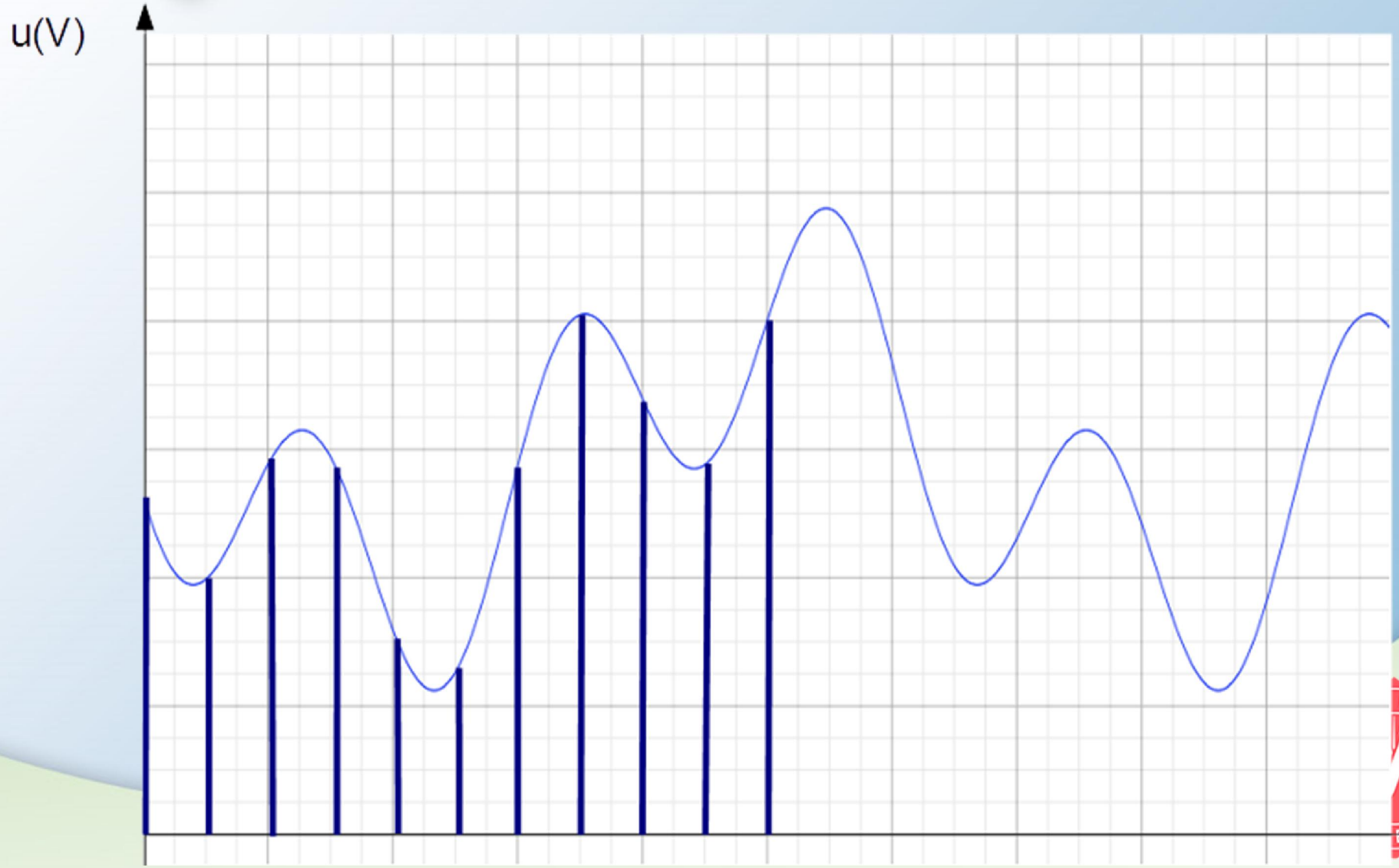
Son

- **Echantillonnage et période (fréquence) d'échantillonnage**



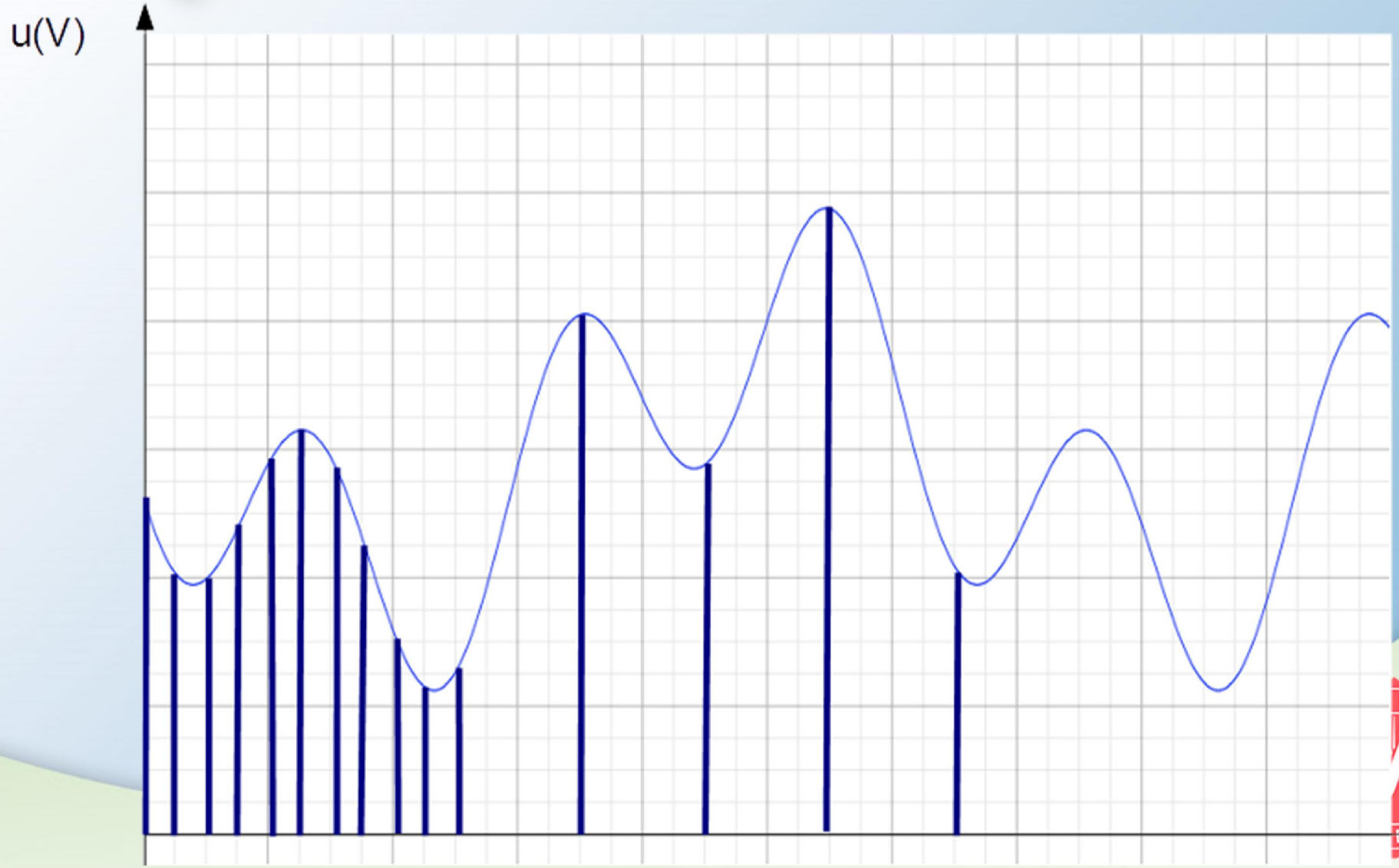
Son

- **Echantillonnage et période (fréquence) d'échantillonnage**



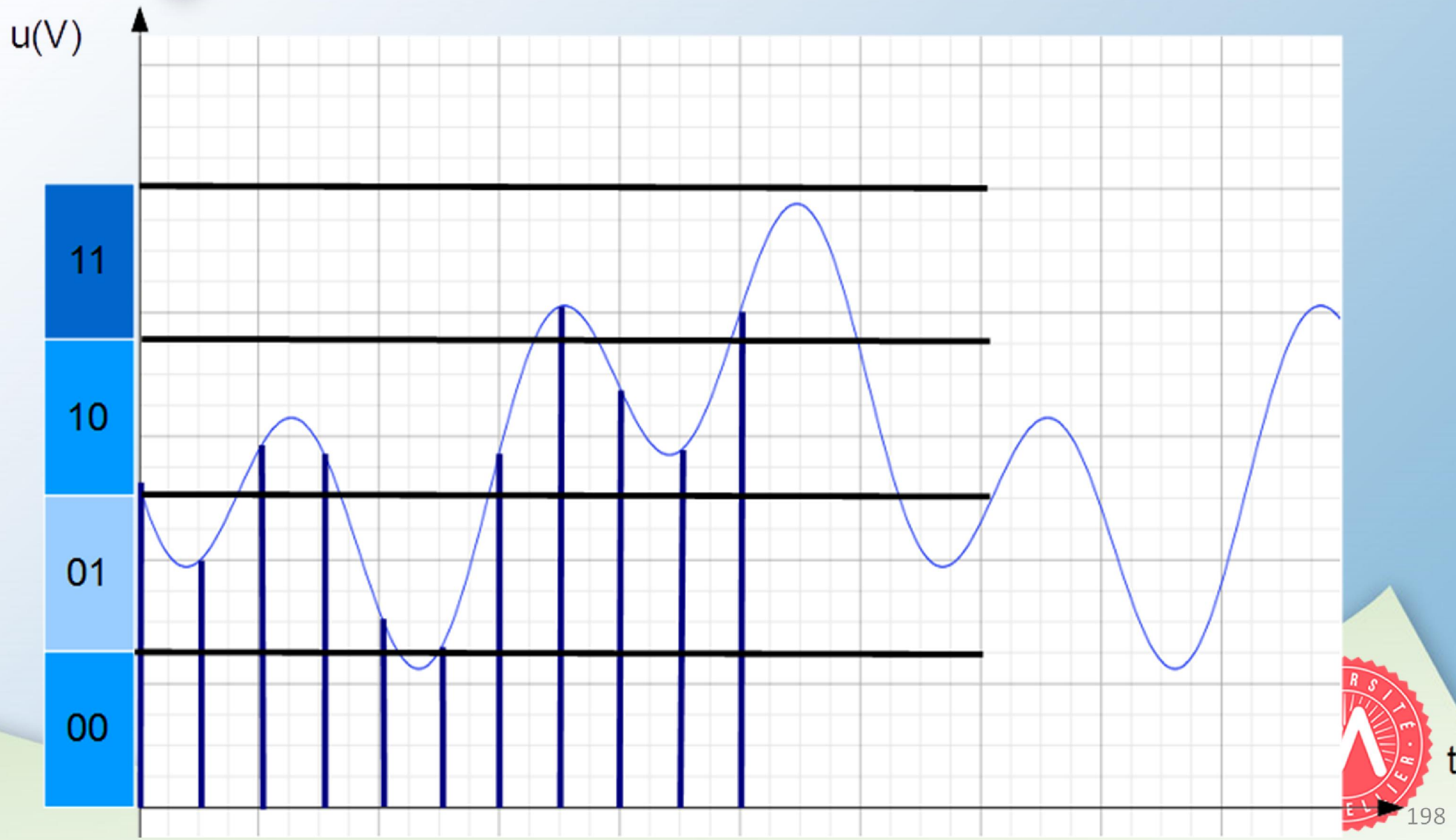
Son

- **Echantillonnage et période (fréquence) d'échantillonnage**



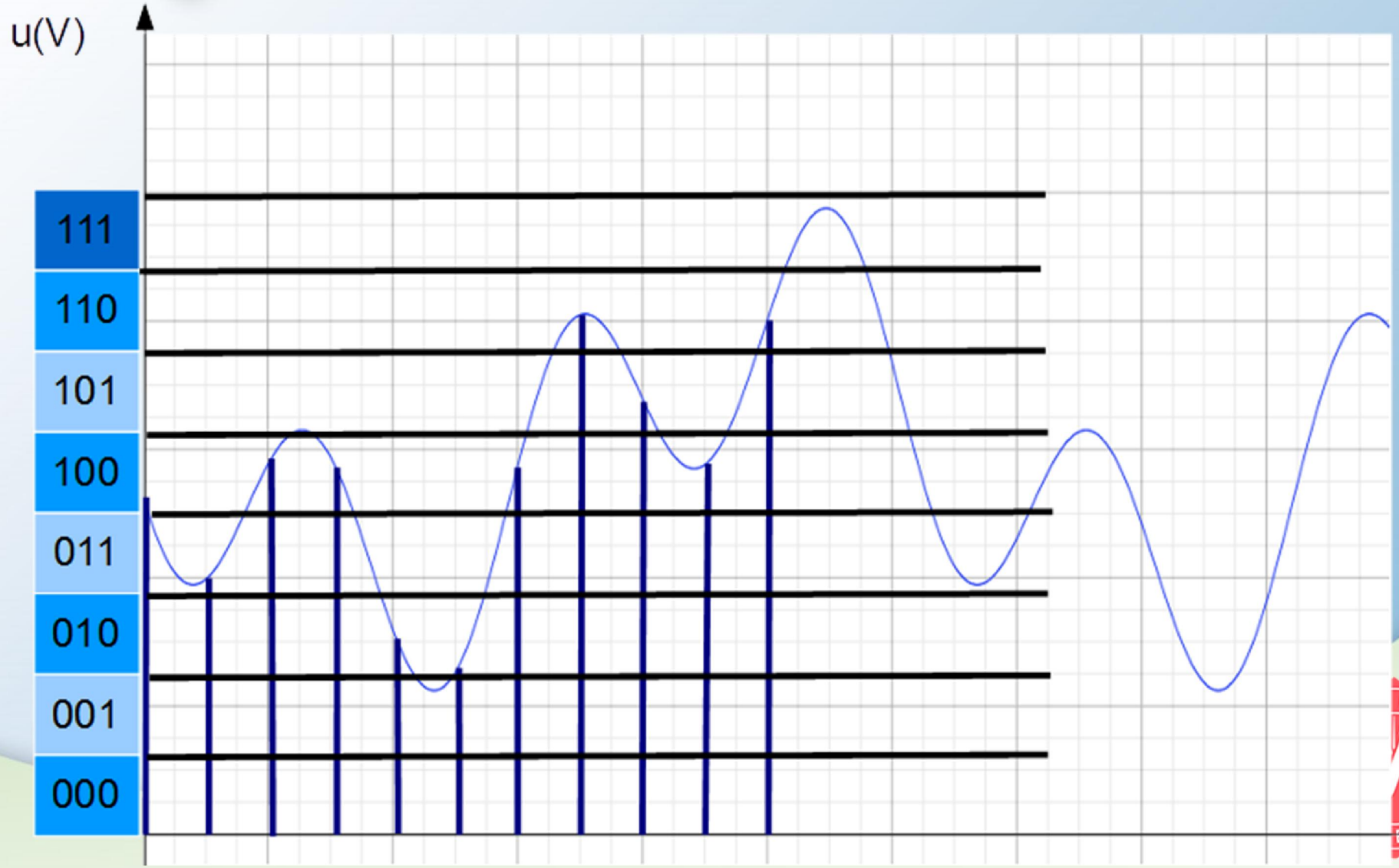
Son

- **Résolution**



Son

- **Résolution**



Son

- **Quelques valeurs numériques**
- **Résolutions :**
  - **Téléphone fixe : 8 bits**
  - **Téléphone mobile : 13 bits**
  - **CD : 16 bits/canal**
  - **MP3 : flux d'entrée 16 ou 24 bits/échantillon + compression → flux de sortie ~ 1,33 bits/échantillon**
- **Fréquences d'échantillonnage :**
  - **Téléphone fixe : 8 kHz**
  - **CD : 44,1 kHz**
  - **MP3 : fréquence d'entrée 32 kHz, 44,1 kHz, 48 kHz, 96 kHz + compression → débit de sortie 32 à 384 kbps**

- **Débit et vitesse de transmission : Exemple du CD**

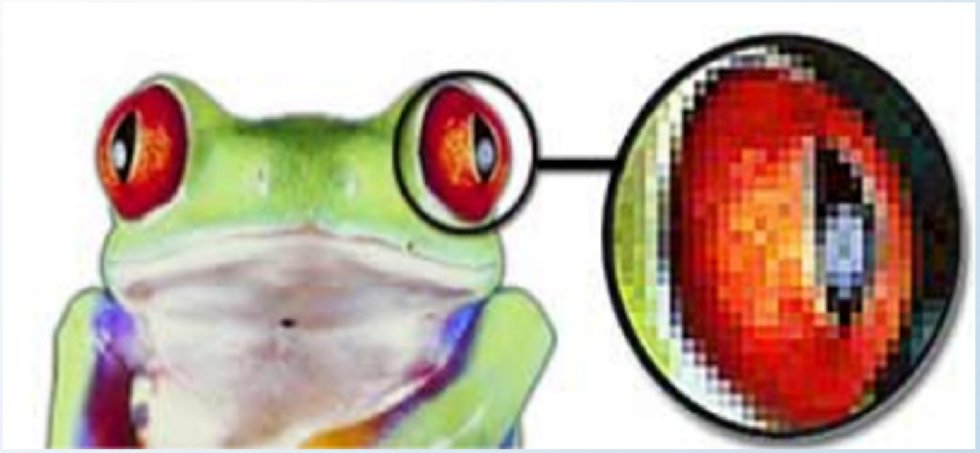
Une minute (60 s) de musique stéréo nécessite:

$$44100 * 60 * 16 * 2 = 84{,}672 * 10^{6}\ \text{bits} \sim 10\ \text{Mo}$$

2.6 Représentation des images fixes

Images

- **Deux types de représentations :**



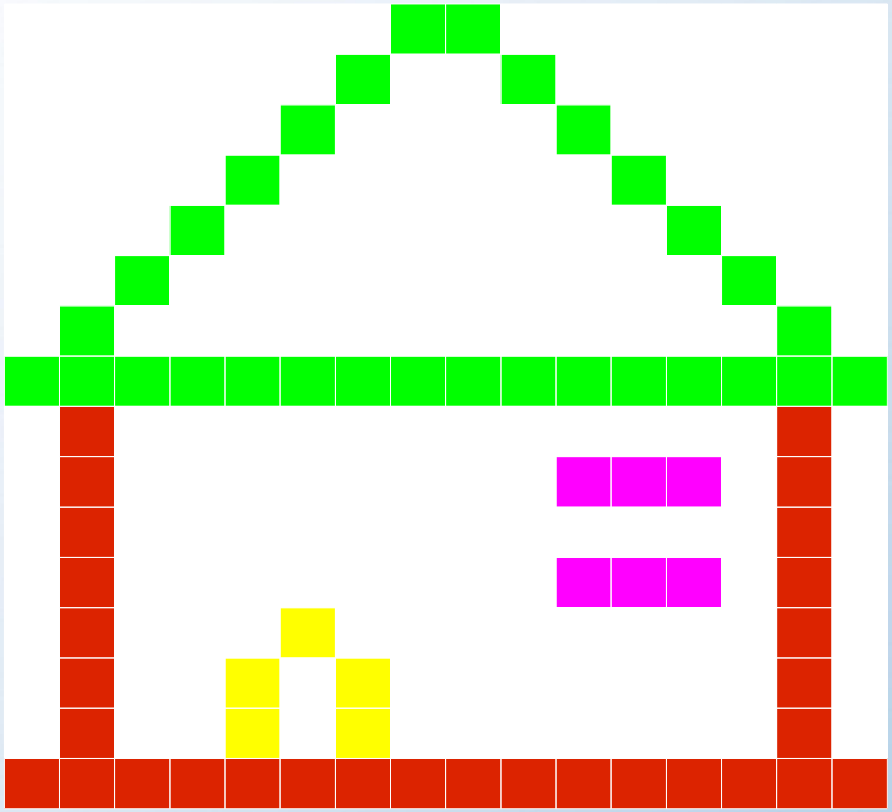
- **Représentation matricielle (« bitmaps »)**

  JPEG, GIF, TIFF, BMP, PNG

- **Représentation vectorielle (tracé de courbes définies par leurs expressions mathématiques)**
  PS/EPS, SVG, AI, FLA/SWF, PDF

Images

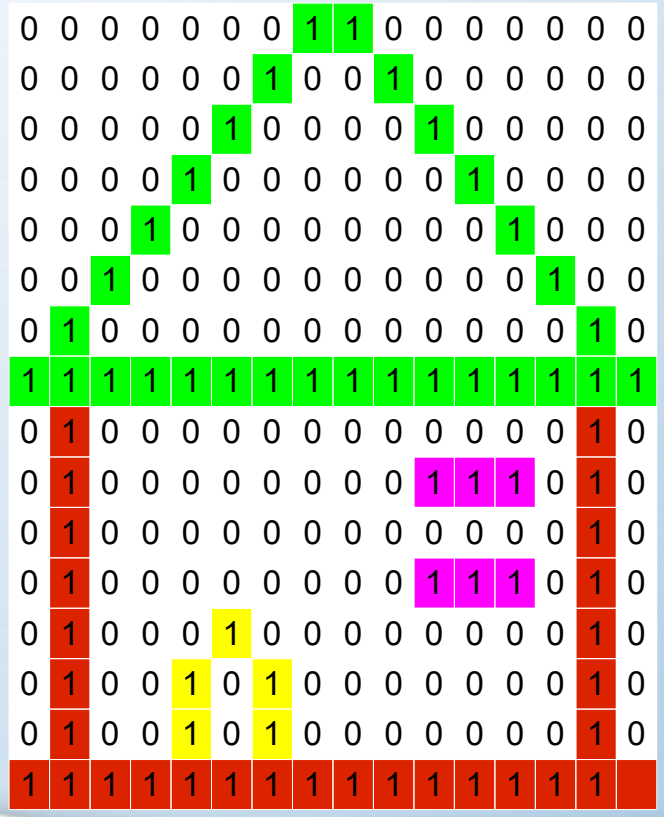
- **Images de type « bitmap »**

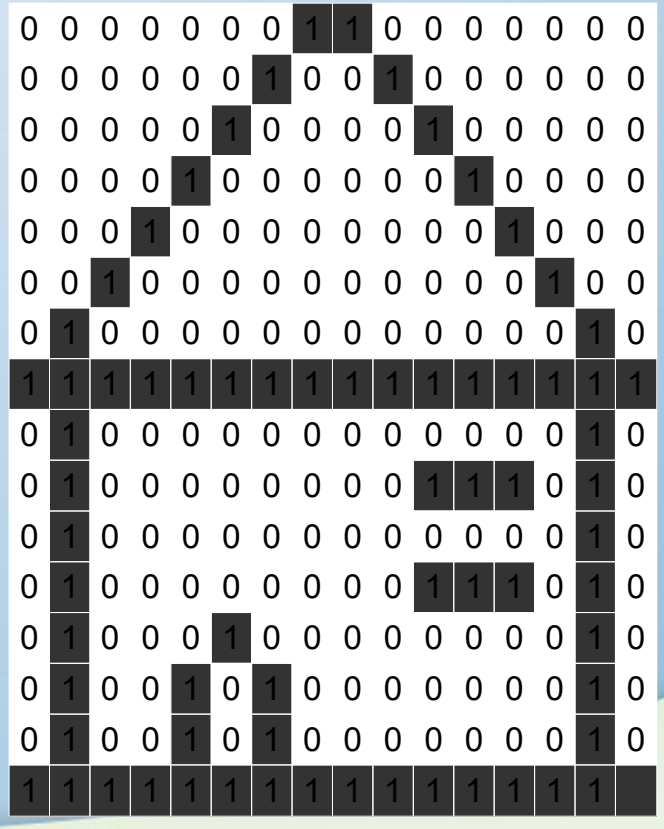


- **Paramètres nécessaires à la représentation :**
  - **Nb de colonnes**
  - **Nb de lignes**
  - **Nb de bits/pixel**

Images

- **Images de type « bitmap »**
  - **Représentation couleur/noir et blanc**





Images

- **Images de type « bitmap »**
  - **Représentation couleur/niveaux de gris/noir et blanc**
    - 2 bits => 4 niveaux de gris
      - 00 : noir 01 : gris foncé
      - 10 : gris clair 11 : blanc
    - 3 bits => 8 couleurs :
      - 000 : noir 001 : rouge
      - 010 : vert 011 : jaune
      - 100 : magenta 101 : blanc

Images

- Images de type « bitmap »
- Représentations graphiques les plus courantes :
  - Image en N/B : 1 bit/pixel
  - Image en niveaux de gris : 8 bits/pixel
  - Image en couleurs de base : 8 bits/pixel
  - Image en couleur : 3*8 = 24 bits/pixel

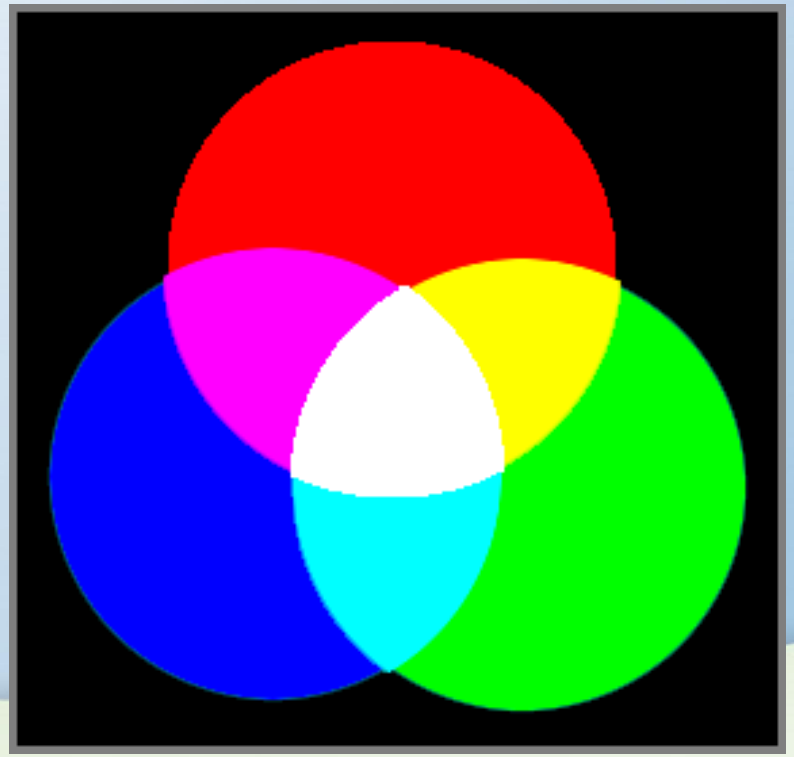
N/B
1 bit/pixel

Niv. gris, 8 bits/pixel

Couleur,
24 bits/pixel

Images

- **Images de type « bitmap »**
- **Représentation RVB (Rouge-Vert-Bleu)** ***RGB***
  - **L'addition des 3 couleurs (avec divers niveaux d'intensité) permet de reproduire toutes les couleurs**
  - **Affichage sur écran : de type RVB**
  - **Addition des 3 couleurs au max de luminosité → blanc**

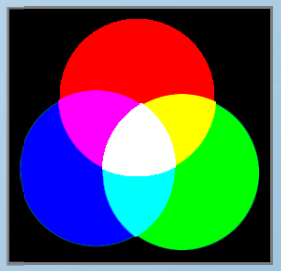


Images

- **Images de type « bitmap »**

- **Représentation RVB (Rouge-Vert-Bleu)**

R	*V*	*B*	*Couleur*
255	0	0	Rouge
255	255	0	Jaune
0	255	0	Vert
0	255	255	Cyan
0	0	255	Bleu
255	0	255	Magenta
255	255	255	Blanc
0	0	0	Noir



- **Avec 3 x 1 octet, on peut coder : 256*256*256 = 16,7 millions de couleurs (l'œil humain en perçoit max. 500 000)**

Images

- **Images de type « bitmap »**

- **Définition de l'image : nb. de pixels = nb colonnes*nb lignes**
- **Résolution : exprimée en PPP = Pixels Par Pouce**

  ou *DPI = Dots Per Inch*

  - **1 inch (pouce) = 2,54 cm**

18 DPI

72 DPI