PF1 — Principes de Fonctionnement des machines binaires

Jean-Baptiste Yunès
Jean.Baptiste.Yunes@univ-paris-diderot.fr

Version 1.2

Codage

Codage : conversion d'une représentation en une autre

code Morse ?

Le codage va parfois de pair avec la numérisation : procédé consistant à transformer un signal (généralement analogique) en une représentation discrète

échantillonnage de signal ?

Le codage est utilisé à différentes fins :

- représenter quelque chose dans un système numérique
- économiser de l’espace (compression de données)
- rendre illisibles aux non initiés des données (cryptographie)
- résister aux altérations, pertes ou mutations (codes correcteurs)

Bien sûr en informatique, il s’agit de coder de l’information sous la forme d’une suite finie de 0 et 1…

par exemple comment coder l’alphabet latin ?

comment coder le signal numérisé d’une chanson ?

comment coder l’image prise par un appareil photographique ?

comment crypter le contenu de ma clé USB ?

comment assurer que l’image prise par Curiosity sur la planète Mars parvienne sans altération jusqu’à nous ?

Éléments de théorie des codes

Le problème est celui de la représentation de mots écrits dans un alphabet donné en mots sur un autre alphabet

$A = \{ a_0, a_1, a_2, \ldots, a_{n-1} \}$, alphabet de n lettres

un mot fini est une suite finie de lettres $m=m_0m_1\ldots m_{l-1}$, où $m_i \in A$ et l la longueur du mot

on note A^* l'ensemble des mots pouvant être écrits avec l'alphabet A, c'est-à-dire, tous les mots de longueur 0, plus tous les mots de longueur 1, plus tous les mots de longueur 2, … Le mot de longueur 0 est habituellement noté ε

Soit B un second alphabet

on souhaite coder A* sur B*

pour cela on définit le codage des lettres de A en des mots de B* à l'aide d'une fonction $\boldsymbol{\tau}$: A $\mapsto$ B*

le codage par $\boldsymbol{\tau}$ d'un mot de A*, $m=m_0m_1\ldots m_{l-1}$ consiste alors à coder chaque lettre et rabouter les codages dans l'ordre

$\boldsymbol{\tau}(m)= \boldsymbol{\tau}(m_0)\ \boldsymbol{\tau}(m_1)\ldots\ \boldsymbol{\tau}(m_{l-1})$

on identifie $\boldsymbol{\tau}$ sur les mots et $\boldsymbol{\tau}$ sur les lettres…

Attention $\boldsymbol{\tau}$ doit être inversible!

On doit pouvoir retrouver le mot originel à partir de son codage à travers $\boldsymbol{\tau}$!

$\boldsymbol{\tau}$ doit être injective (sur les mots)…

Exemple A={a,b,c} et B={0,1}

$\boldsymbol{\tau}$(a)=00, $\boldsymbol{\tau}$(b)=11, $\boldsymbol{\tau}$(c)=111110

est une fonction de codage acceptable

$\boldsymbol{\tau}$(a)=0, $\boldsymbol{\tau}$(b)=01, $\boldsymbol{\tau}$(c)=10

n'est pas une fonction acceptable, 010 c'est **0**10 ou 01**0 ?**

Si toutes les images par τ sont de même longueur, le code est dit de longueur fixe

- il suffit d'avoir $k \geq \log_{|B|}(|A|)$ caractères dans B pour coder A sur B*

- A={a,b,c}, B={0,1}, il faut k=2 lettres

- A={a,…,z}, B={0,1}, il faut k=5 lettres

- A={A,…Z,a…,z,0,…,9}, il faut k=6 lettres

Le décodage est facile à partir du découpage du mot image en blocs de k lettres…

Codes à longueur variable

ces codes sont plus difficiles à construire…

les codes préfixes sont des codes de longueur variable pas trop difficiles à construire

un code préfixe est un code pour lequel aucune image d'une lettre n'est le préfixe de l'image d'une autre lettre

ex. : 0, 10, 110, 1110

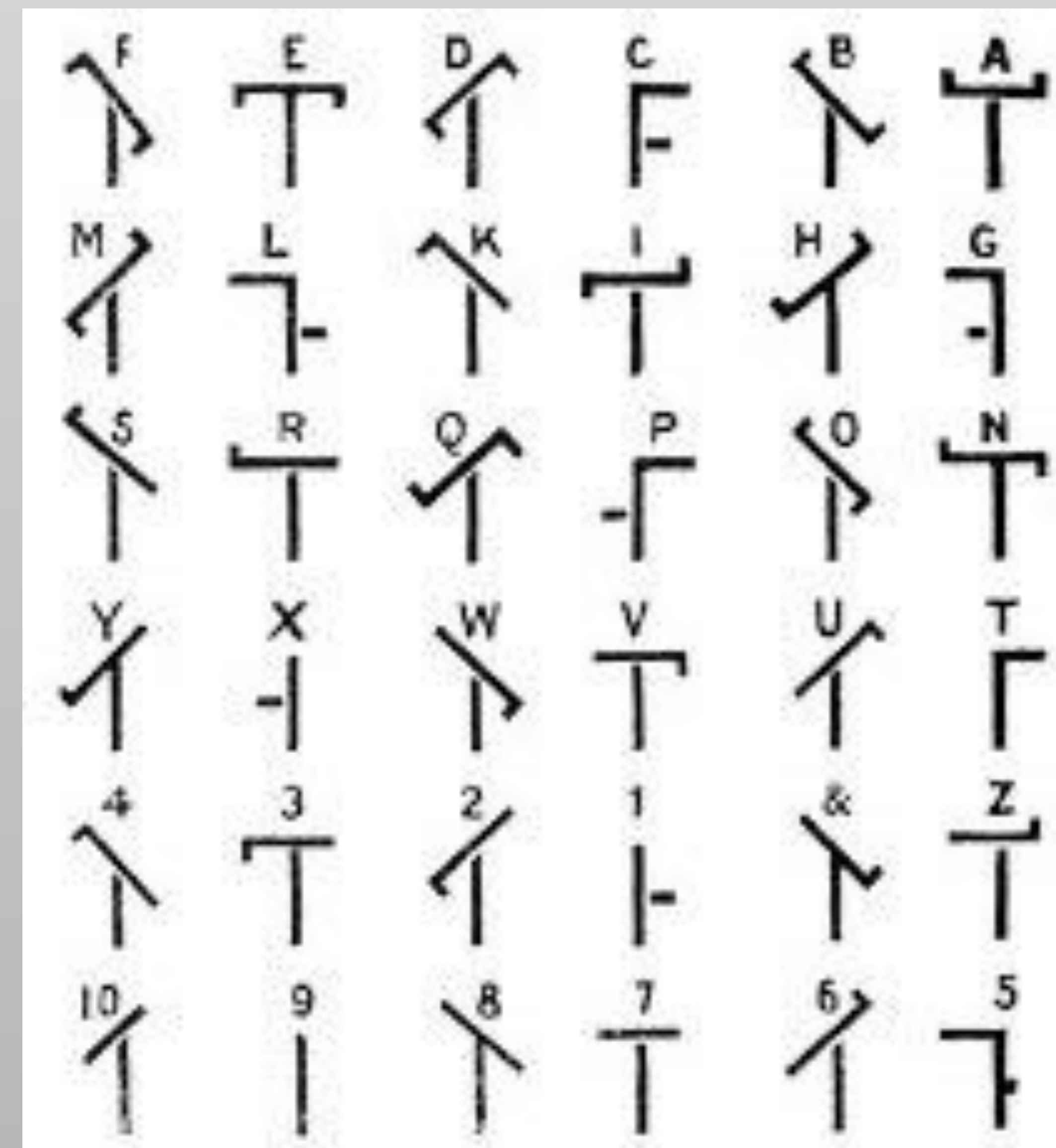
ils sont très utiles si la fréquence d'apparition des symboles de A n'est pas uniforme…

Huffman (on verra bientôt)

Codage de textes (d’alphabets)

Les premiers codes

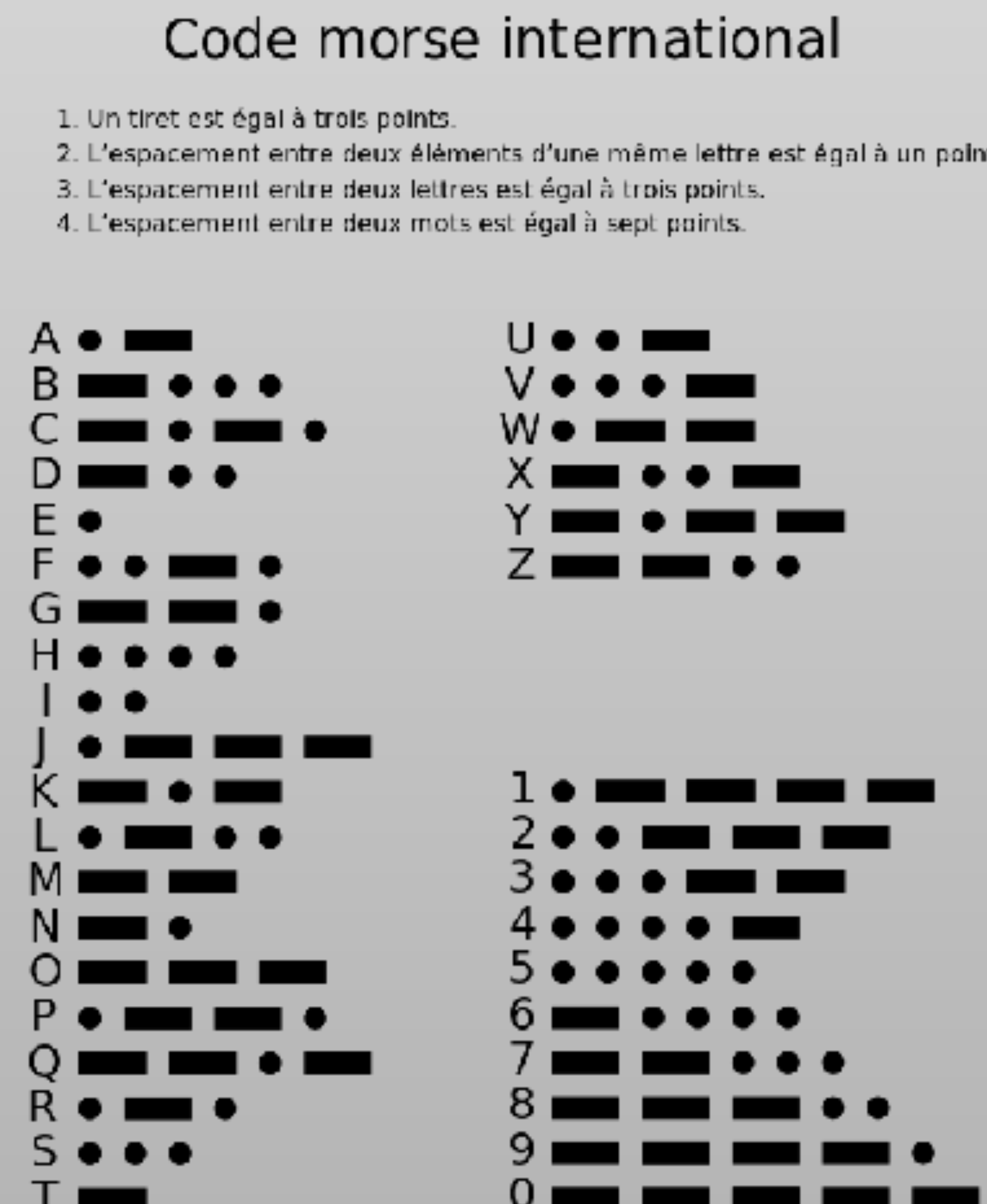
1794, le télégraphe de Chappe (Claude Chappe)
(machine + code + cryptographie)

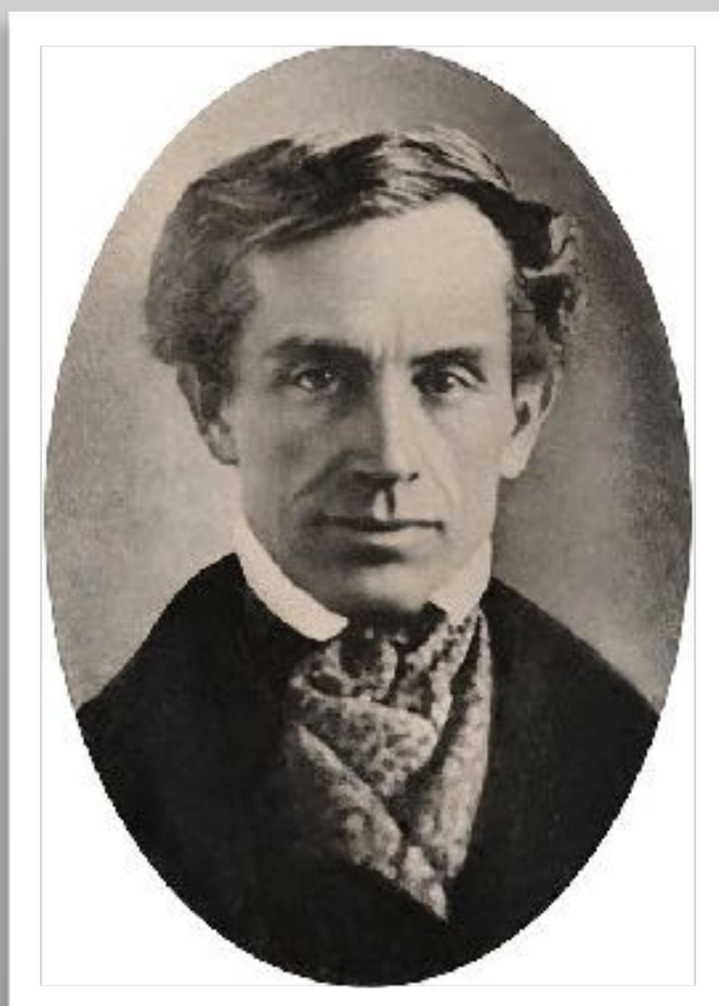


Sources Wikipédia

Les premiers codes

1832, le code Morse (attribué à Samuel Morse) est indépendant du support de transmission, il code dans un alphabet binaire (code de longueur variable)





Sources Wikipédia

Les premiers codes

1874, le code Baudot (Émile Baudot), trois fois plus rapide que le code Morse… Il a constitué la première normalisation d'un alphabet numérique international CCITT n°1

le baud est une unité de mesure en transmission (nombre de symboles / secondes)

Sources Wikipédia

Les demandes de normalisation de codage d’alphabets :

Baudot (5 bits)

TTS (6 Bits)

7-bits ASCII (1972 norme ISO/CEI 646)

8-bits ISO/CEI-8859-1 (1986) dit Latin-1 ou ISO/CEI-8859-15 dit Latin-9 (1998), des extensions au code ASCII

16-bits UNICODE

32-bits UNICODE

Attention, Unicode :

- définit les **jeux de caractères**, leur **numérotation** et **nommage**, etc.
- le **codage** peut-être de longueur variable…
  - UTF-8 (8, 16, 24, 32 bits), UTF-16 (16, 32 bits)
- ou de longueur fixe
  - UTF-32 (32 bits)
  - java utilise le jeu de caractères Unicode encodé via UCS-2 (16 bits)

Le code ASCII originel (American Standard Code for Information Interchange)

- codage des caractères alphabétiques latins non accentués, majuscules et minuscules, chiffres, signes et symboles annexes, caractères spéciaux dits de contrôle. 94 caractères.
- sur 7 bits, ou sur 8 bits avec le bit de poids fort égal à 0. On utilisait parfois ce huitième bit pour réaliser une somme de contrôle, permettant de valider la transmission…

PDF : fr en v · d · m	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
002	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
003	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
004	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
005	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
006	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
007	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

source Wikipédia

ISO/CEI 8859-1																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	*positions inutilisées*															
1x																
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	*positions inutilisées*															
9x																
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Source Wikipédia

ASCII étendu. Extension ? ISO-8859-1

191 caractères

Problèmes : bugs où est œ ? pas encore €…

ASCII étendu

Extension ?

ISO-8859-15

191 caractères

Source Wikipédia

ISO/CEI 8859-15																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	*non utilisé*															
1x																
2x		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	*non utilisé*															
9x																
Ax		¡	¢	£	€	¥	Š	§	š	©	ª	«	¬		®	¯
Bx	°	±	²	³	Ž	µ	¶	·	ž	¹	º	»	Œ	œ	Ÿ	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Code sur 8 bits, combien de caractères au maximum ?

Ne permet pas de supporter plusieurs langues en même temps

- passage à un codage plus long
- le type `char` en Java est sur 16 bits (encodage UCS-2)

Démonstration

- utilisation d’un éditeur(par exemple **`emacs`**) afin d’éditer du texte et essayer plusieurs encodages

- utilisation de l’outil **`od`** permet de *dumper* (c’est-à-dire examiner le contenu brut) d’un fichier quelconque

Les pages webs d’Internet :

utilisent le langage de description HTML pour structurer le contenu (basiquement : du texte)

le texte est nécessairement encodé, c’est pourquoi HTML permet de préciser quel encodage a été utilisé

HTML permet de spécifier l'encodage via

pour HTML4

```
<meta http-equiv="Content-Type"
content="text/html;charset="ISO-8859-1"/>
```

pour HTML5

```
<meta charset="UTF-8"/>
```

pour XHTML5

```
<?xml version="1.0" encoding="UTF-8"?>
```

si l'on indique rien c'est UTF-8 pour HTML5 et ISO-8859-1 pour HTML < 5 par défaut.

Java et les caractères

- le type `char` est en fait un type entier 16 bits non-signés dont la représentation utilise l'encodage UNICODE UCS-2
- un **littéral de caractères** est :
  - directement le caractère entouré de simples apostrophes comme `'A'` ou `'z'` ou encore `'ặ'`
  - le code UNICODE du caractère entouré d'apostrophes et préfixé par `\u` comme `'\u1BE7'`

```java
char c = 0x1EB7;
System.out.println(c);

char d = 'ặ';
System.out.println(d);

char e = '\u1EB7';
System.out.println(e);
```

dans certains cas il est utile de préfixer le caractère souhaité à l'aide du caractère `\`

comme pour le littéral de caractère représentant l'apostrophe ou le caractère `\` lui-même!

i.e. : `'\''` ou `'\\'`

ou pour représenter des **caractères spéciaux** comme le passage à la ligne ou la tabulation

i.e. : `'\n'` ou `'\t'`

Java autorise l'utilisation de certains caractères UNICODE UCS-2 y compris pour les identificateurs!

```
int i = 0; // légal

double π=3.1415926; // légal!

int âgeDuCapitaine = 77; // légal

double \u0394=b*b-4*a*c; // légal
```

Attention, il ne faut pas trop en abuser…

Attention, certains symboles ont différents encodages…

∆ : e2 88 86 (U+2206 - INCREMENT) ou ce 94 (U+0394 GREEK CAPITAL LETTER DELTA)

seul le dernier est valide (UCS-2).

L’insertion dans un fichier de caractères UCS-2 est dépendante du logiciel et du système hôte…

emacs permet la saisie de codes UCS via [ALT]-x ucs-insert [RET] *code-hexadécimal*[RET]

La transmission électronique nécessite parfois l’encodage de fichiers contenant du «binaire» en suite de caractères alphabétiques :

- uuencode
- Base64
- quoted-printable

uuencode encode sous la forme de texte ne comprenant que les 65 caractères suivant :

es	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`															

L'idée est de transformer chaque suite de 24 bits en suite de 4 caractères. Chaque sous-mot de 6 bits (2^6=64) x est codé en le caractère de code ASCII x+32

Ex.: `000011110000111100001111`

`000011➟32+ 3=35➟#`

`110000➟32+48=80➟P`

`111100➟32+60=92➟\`

`001111➟32+15=47➟/`

base64 repose sur le même principe mais encode vers les 64 caractères

`A-Za-z0-9+-`

Une suite de 24 bits est découpée en sous-mots de 6 bits.

Ex.: `000011110000111100001111`

`000011`➟`D`

`110000`➟`w`

`111100`➟`8`

`001111`➟`P`

quoted-printable encode chaque octet :

- si l'octet correspond à un caractère ASCII imprimable ($33 \leq code \leq 126$), il est peut-être représenté par lui-même

- sinon la représentation héxadécimale est utilisée (deux chiffres de la base 16) préfixée par le caractère =

Example : extrait du contenu d'un mail contenant en pièce jointe un «.zip»

```
--Apple-Mail=_1761FE5C-6D12-45B4-819B-68FE2FED470A
Content-Disposition: attachment;
	filename="dossier sans titre.zip"
Content-Type: application/zip;
	x-mac-auto-archive=yes;
	name="dossier sans titre.zip"
Content-Transfer-Encoding: base64

UEsDBAoAAAAAAIxTS0sAAAAAAAAAAAAAAAAATABAAZG9zc2llciBzYW5zIHRpdHJlL1VYDAAy1t1Z
KNbdWfUBFABQSwECFQMKAAAAAACMU0tLAAAAAAAAAAAAAAAAEwAMAAAAAAAAAAAABA7UEAAAAAZG9z
c2llciBzYW5zIHRpdHJlL1VYCAAy1t1ZKNbdWVBLBQYAAAAAAQABAE0AAABBAAAAAAA=
--Apple-Mail=_1761FE5C-6D12-45B4-819B-68FE2FED470A
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain;
	charset=utf-8



--
M. Jean-Baptiste Yun=C3=A8s (y=3D>)
http://www.irif.univ-paris-diderot.fr/~yunes/
```

Pour d'autres formats «exotiques» de transmission consultez Internet

Binary to Text encoding

Codage d’images

Deux types d’images numériques

matricielles : description sous la forme d’une matrice dans laquelle chaque élément correspond à un point de l’image. La description est discrète (résolution)

vectorielles : description « idéale » sous la forme d’opérations géométriques appliquées à des formes décrites à l’aide d’équations mathématiques (ex: certaines polices de caractères)

Images matricielles (bitmap/raster)

- on décompose l'image en points
  - un point est un « atome » de la représentation
    - la valeur d'un point est la composition des valeurs des caractéristiques intéressantes (en général la couleur)
      - codage de la couleur ? (plus loin)
      - les valeurs aussi sont discrètes
- un tel point est appelé **pixel** : **pic**ture **el**ement

Codage des couleurs

Attention : les dispositifs de reproduction de la couleur ne peuvent représenter l’ensemble des couleurs que l’œil humain peut percevoir

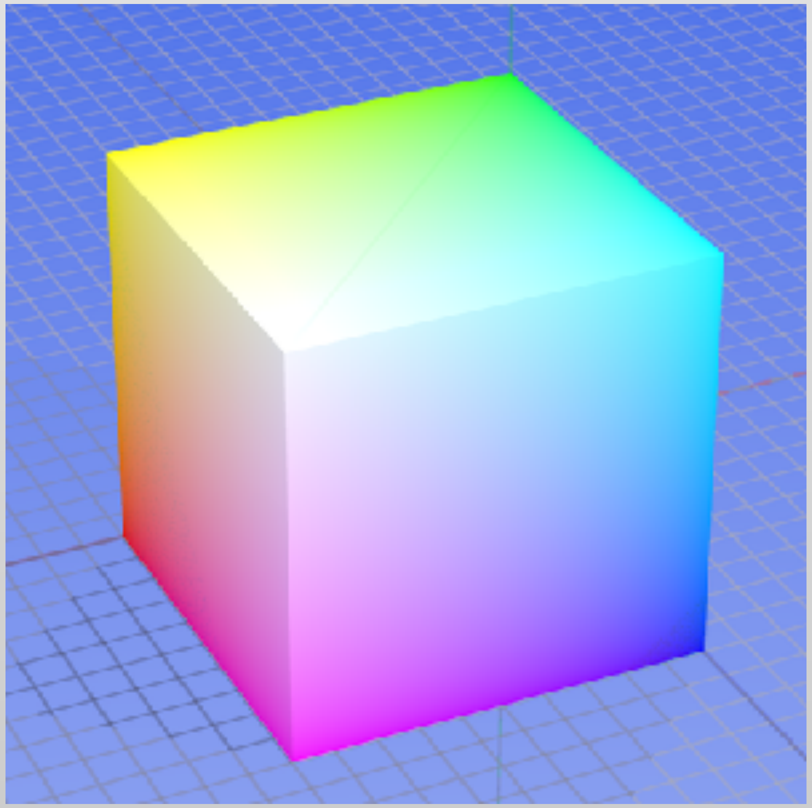
L’ensemble des couleurs reproduites par un appareil donné est son **gamut**

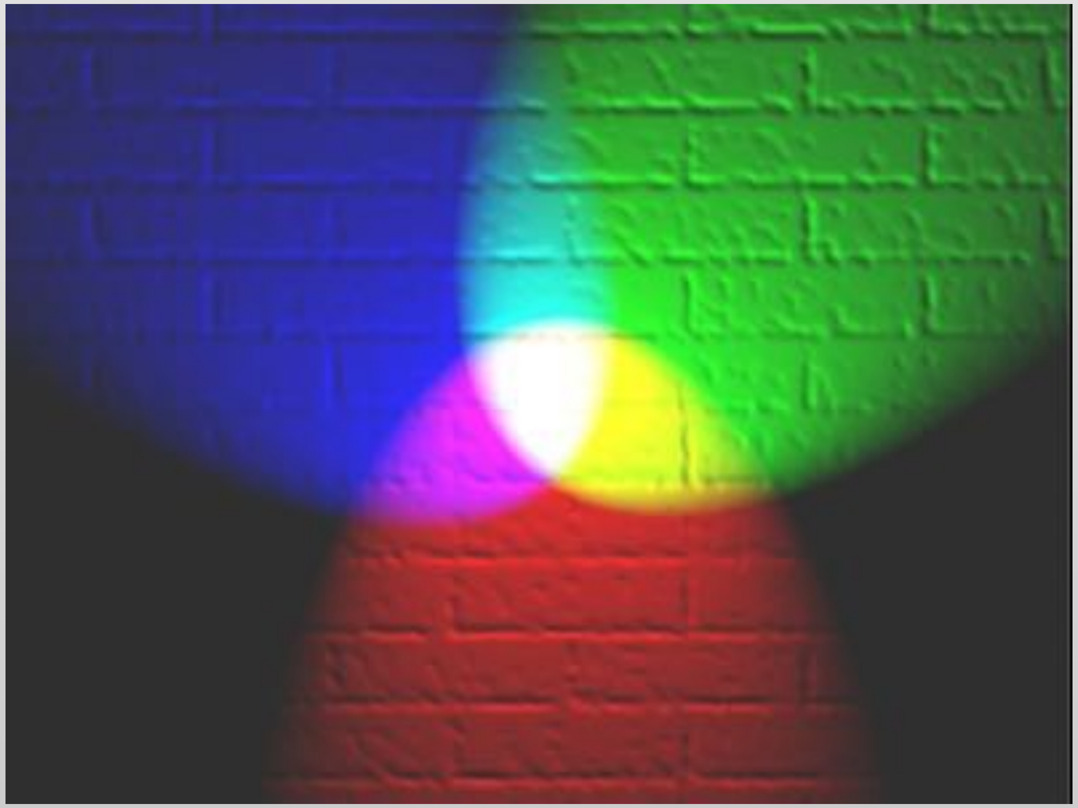
Attention : un dispositif de reproduction de la couleur n’est adapté qu’à un type d’œil

Codage des couleurs

- il existe de nombreuses façons de coder les couleurs numériquement, dont les «classiques» :
  - le **noir et** blanc - black and white
  - les **niveaux de** gris - grayscale : palette permettant de représenter différents gris, du noir au blanc (16 gris ou 256 gris sont courant)
  - en **RVB**/RGB: système additif qui compose une couleur par addition de couleurs primitives (rouge, vert, bleu)
  - en **HSB**/TSV : système de coordonnées dans un espace contenant des couleurs

il existe des modes avec une valeur de transparence : canal alpha





Synthèse additive (ex. : écrans)

Synthèse soustractive (ex. : imprimante)

La valeur associée à un pixel représente :

soit une valeur « directe » (*direct color*)

soit une indirection dans une table qui contient les valeurs (*color table*)

En mode direct ou absolu comment coder une couleur RBG ?

- un nombre pour le R, un pour le V, un pour le B
  - mode arithmétique [0.0,1.0] (avec des flottants!)
  - mode pourcentage 0%-100% (des entiers suffisent)
  - mode numérique, un nombre sur n-bits
    - valeur pour n ?
    - comme d'habitude, 1, 2, 4, 8, 16 ou 32…
  - le nombre est ensuite codé en vue de son stockage

Caractéristiques d’une image

- définition
- résolution
- poids

Les dimensions de la matrice qui représente l’image.

Combien de lignes et colonnes…

C’est ce que l’on appelle la **définition** de l’image, c’est-à-dire sa finesse intrinsèque.

C’est exprimé en pixels (**pic**ture **el**ements) comme un produit de deux nombres. ex : 1024x768 px

Un pixel doit-il être **représenté** par une grande surface ou une petite surface ?

C’est ce que l’on appelle la **résolution**.

Elle indique la finesse de représentation de l’image.

C’est en général exprimé sous la forme d’une densité linéaire : nombre de **pixel par pouce** / **point par pouce** / *dot-per-inch* / *pixel-per-inch*

Il peut y avoir deux densités, une horizontale et une verticale.

Le nombre de bits utilisés pour coder une image est appelé son **poids**.

Sans compression c’est le produit de la définition par le nombre de bits utilisés pour coder un pixel.

résolutions communes:

- affichage public type barco : quelques unités
- écran : de l’ordre de la centaine
- imprimante : plusieurs centaines (voire milliers)

c’est lié à la distance moyenne d’observation, à la capacité du dispositif et à la qualité souhaitée…

On rappelle que l’œil a un pouvoir de résolution ou encore pouvoir de séparation

c’est le cas de tous les dispositifs optiques

pour l’œil c’est environ 1’ d’arc c’est à dire 1/60° de degré

Dans un amphi (au fond) qu’êtes-vous êtes capable de distinguer ?

Dans un amphi (au fond) qu’êtes-vous êtes capable de distinguer ?

disons 20m de distance

$\sin(\pi/180/60)*20 \approx 5$ mm

Format de fichier codant une image

Ex. : Le format netpbm

http://netpbm.sourceforge.net/doc/ppm.html

Ce format est particulièrement inefficace en terme de poids mais est très simple

Netpbm (sous format plain ppm)

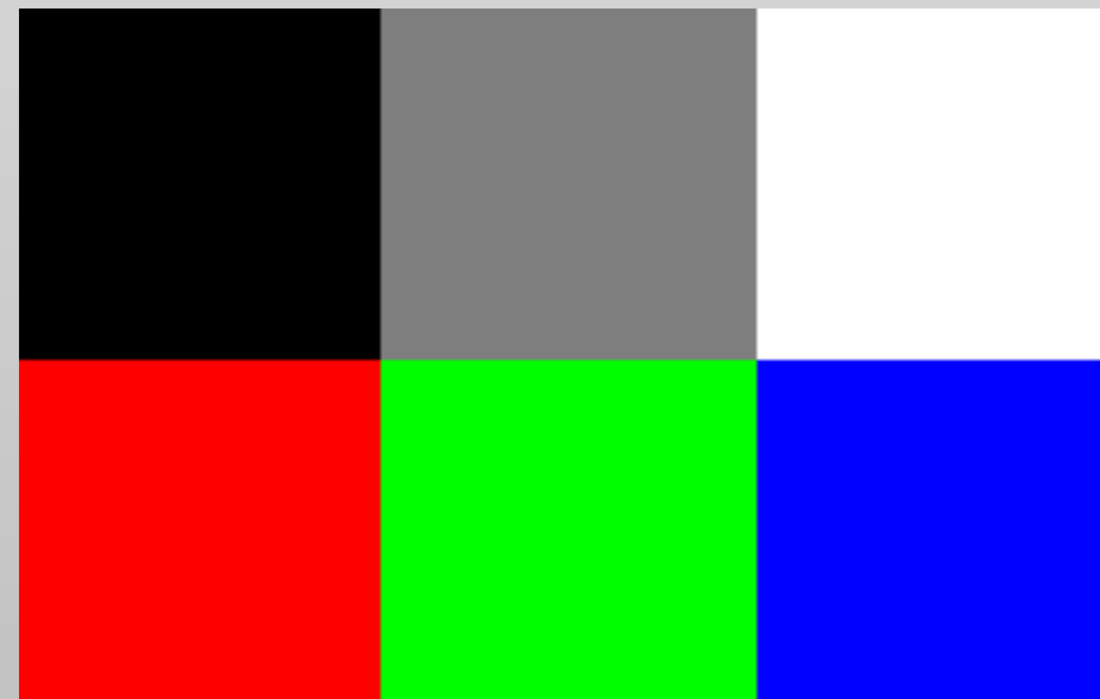
les nombres nécessaires au codage d’une image sont représentés par le codage sous forme ASCII de leurs valeurs (i.e. le nombre douze s’écrit 12)

Netpbm (sous format plain ppm)

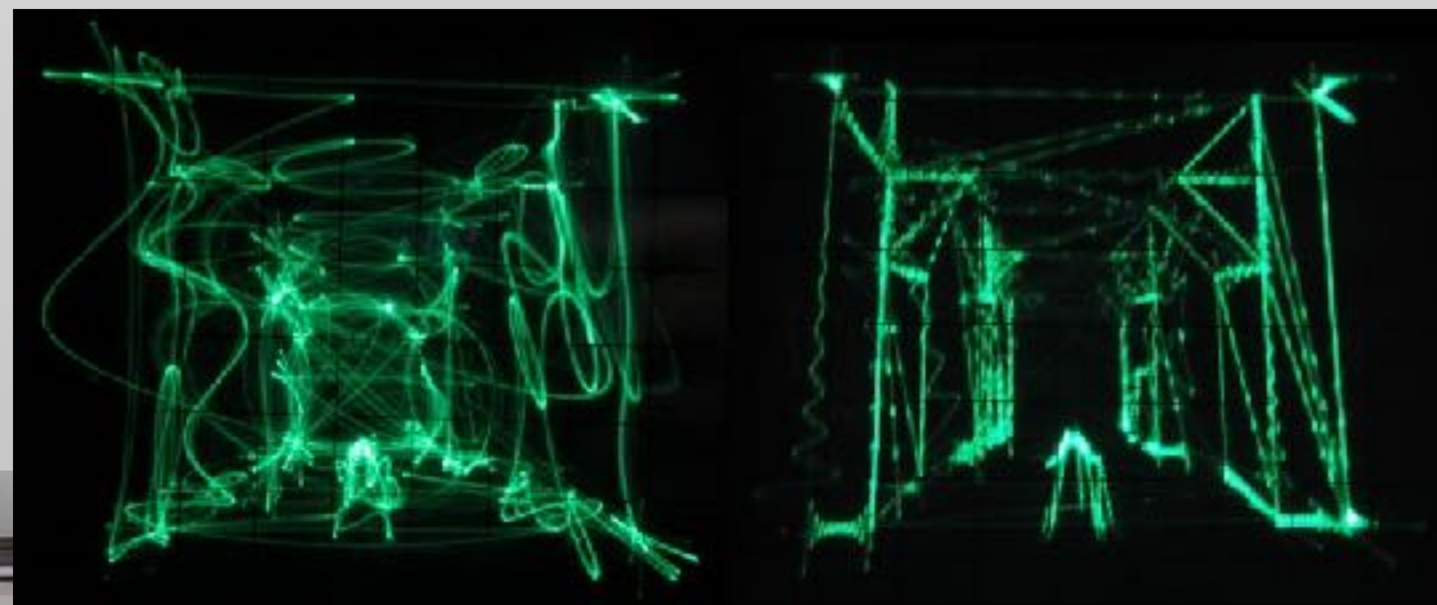
- l’image est décrite par la suite de nombre représentant les caractéristiques suivantes :
  - les caractères magiques P3
  - la définition (largeur puis hauteur)
  - la valeur maximale d’un canal de couleur RGB
  - les triplets RGB, de gauche à droite puis de haut en bas

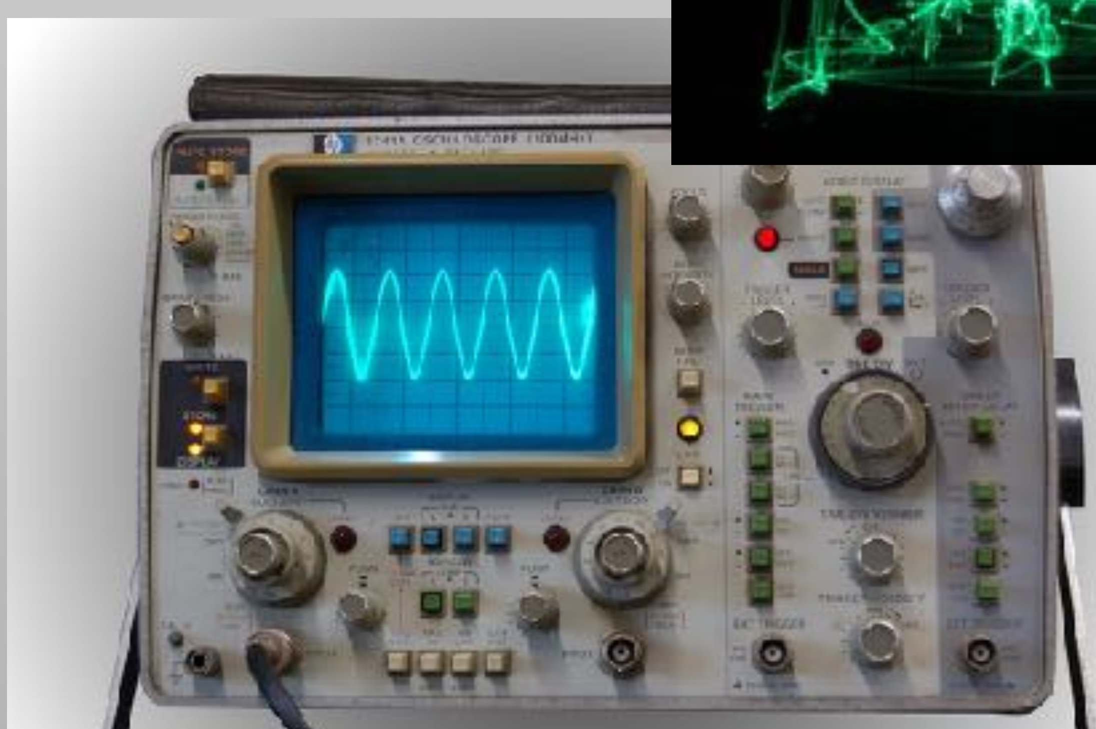
Netpbm (un exemple)

```
P3 3 2 256
0 0 0
127 127 127
255 255 255
255 0 0
0 255 0
0 0 255
```



Il a existé par le passé des dispositifs d’affichage vectoriel : téléviseurs cathodiques, écrans vectoriel
Il existe encore certains appareils de mesure de ce type comme les oscilloscopes





Les images vectorielles :

SVG

DVI, PS, PDF

SVG (Scalable Vector Graphics)

un langage de description d’image indépendant de la résolution…

```
<svg width="200"
     height="200"
     xmlns="http://www.w3.org/2000/svg">
   <desc>Exemple SVG - Cercle</desc>
   <circle cx="100"
           cy="100"
           r="50"
           fill="none"
           stroke="black"
           stroke-width="2" />
</svg>
```

un langage puissant…

```
<svg width="2000" height="2000"
   xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   version="1.1">
  <desc>Exemple SVG - Cercle</desc>
  <path id="monChemin"
     d="M 150 150 C 150 50 300 50 300 150"
     fill="none" stroke="blue"
     stroke-width="5" />
  <text font-family="Helvetica"
     font-size="30" fill="red">
    <textPath xlink:href="#monChemin">Coucou
     me voil&#x00E0;!
    </textPath>
   </text>
</svg>
```