Modulo.
Une introduction à l'informatique

Groupe de travail DGEP, EPFL, HEP-VD, UNIL

13 septembre 2024

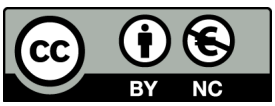


Table des matières

CHAPITRE **1**

Représentation de l'information

1.0 Introduction

Dans ce chapitre on s'intéresse à la manière dont un *ordinateur* représente l'information afin de pouvoir la traiter automatiquement.

Le saviez-vous ?

Le mot **informatique** est la concaténation de «information» et «automatique».

En informatique, l'information est un élément de connaissance (texte, image, son, …) susceptible d'être *numérisé*, *stocké* et/ou *transmis* à l'aide d'un support et d'un mode de codification normalisé.

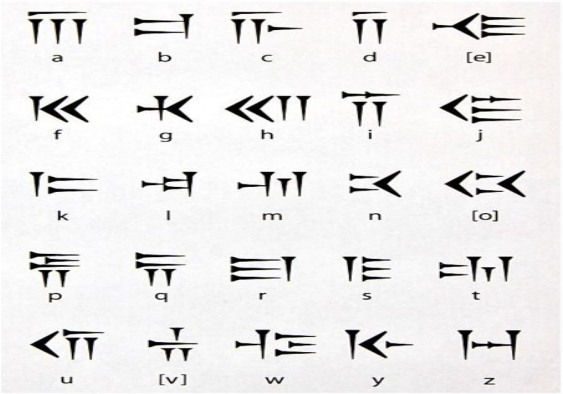
Une des questions centrales de ce chapitre est d'identifier les caractéristiques de la transformation appliquée au réel donnant une représentation suffisamment précise pour permettre aux ordinateurs de la *traiter* de manière fiable.

Mais avant de découvrir le code choisi pour représenter l'information à l'intérieur d'un ordinateur, passons en revue un certain nombre d'alphabets et de systèmes de représentation de l'information qui ont été utilisés au cours de l'histoire.

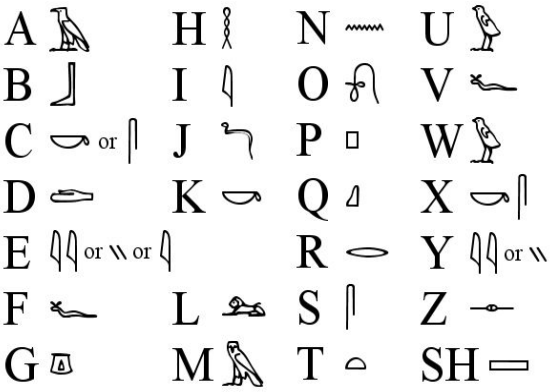
1.0.1 Alphabets anciens et traditionnels

Depuis qu'elle existe, l'espèce humaine a créé de nombreux alphabets, ainsi que de nombreux *systèmes de communication*. Depuis les sumériens [1] qui utilisaient des *pictogrammes* et l'écriture cunéiforme [2], en passant par les égyptiens et leurs hiéroglyphes [3], les chinois et leurs idéogrammes [4] pour arriver aux symboles de nos alphabets actuels, l'espèce humaine n'a eu de cesse de mettre au point des systèmes pour représenter l'information et la *transmettre*.

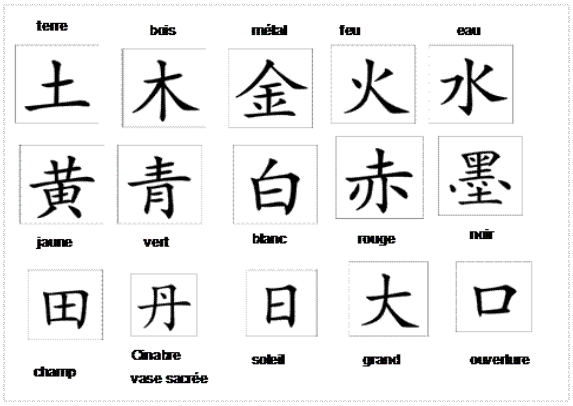
Sumérien



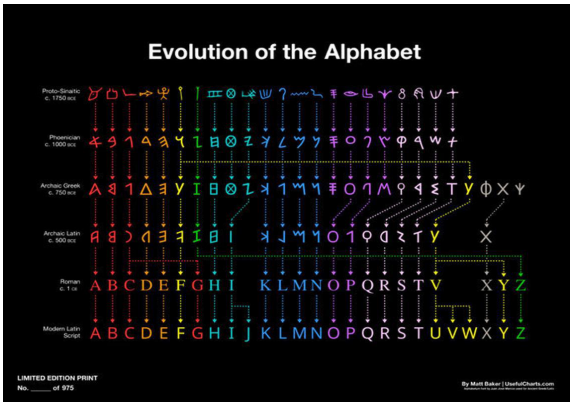
Égyptien



Chinois



Synoptique



Différentes représentations de la même information

- Nombres en chiffres classiques, romains, lettres
- Mot en différentes langues, morse, idéogrammes
- Symboles danger, stop, paix

On trouve des exemples célèbres et bien documentés de *systèmes de communication* depuis l'Antiquité grecque.

1. https://fr.wikipedia.org/wiki/Sum%C3%A9rien
2. https://fr.wikipedia.org/wiki/Cun%C3%A9iforme
3. https://fr.wikipedia.org/wiki/%C3%89criture_hi%C3%A9roglyphique_%C3%A9gyptienne
4. https://fr.wikipedia.org/wiki/Caract%C3%A8res_chinois

1.0.2 Le carré de Polybe

Utilisé en Grèce Antique pour transmettre des messages entre cités voisines, ce système utilisait des torches enflammées en guise de signaux.

Cinq torches «à gauche», cinq torches «à droite», étaient séparées par un espace suffisamment grand pour être identifiables à longue distance. Une torche pouvait être soit allumée, soit éteinte. Le nombre de torches allumées à gauche, de 1 à 5, représentait la ligne d'un tableau de décodage, le nombre de torches allumées à droite représentait la colonne de ce même tableau.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I,J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Fig. 1.1 – Le codage de la lettre «s» dans le carré de Polybe est quatre torches à gauche, trois torches à droite.

Remarque

Dans l'exemple ci-dessus, on utilise les lettres de l'alphabet, mais il est plus probable qu'à l'époque les cités voisines utilisaient des expressions toutes faites dans chacune des cases, comme «l'ennemi est à nos portes» ou «envoyez-nous de l'aide».

1.0.3 Le télégraphe de Chappe

Grâce à l'invention du télescope[5] au XVIIe siècle, les distances avec lesquelles les villes pouvaient communiquer entre elles ont largement augmenté. L'information a commencé à circuler à une vitesse étonnante. Des messages pouvaient être transmis sur une longue distance par l'intermédiaire de relais espacés d'une dizaine de kilomètres et situés sur des hauteurs.

Claude Chappe[6], inventeur français, développe en 1794 un *télégraphe* capable de relier des villes entre elles sur plusieurs dizaines de kilomètres grâce à un système de bras mobiles, qui ressemblent aux signaux que pourrait faire un être humain sur le tarmac d'un aéroport.

5. https://fr.wikipedia.org/wiki/T%C3%A9lescope
6. https://fr.wikipedia.org/wiki/Claude_Chappe

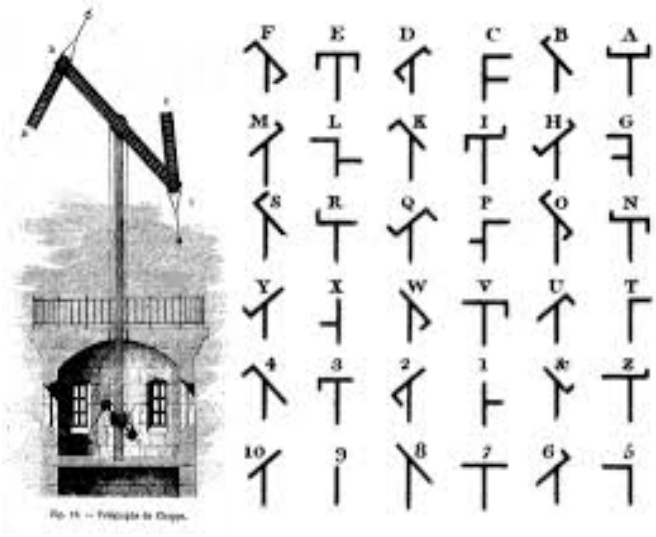


Fig. 1.2 – Le télégraphe de Chappe émet des signaux ressemblant aux bras d'un être humain.

Le saviez-vous ?

Le piratage du télégraphe Chappe [7] est un détournement du réseau de télégraphie optique français entrepris par deux hommes d'affaires bordelais, Louis et François Blanc, entre 1834 et 1836, afin de connaître avant tout le monde le prix de certaines actions à la Bourse de Paris.

Le piratage a été rendu possible par la corruption d'un agent télégraphique de Tours, qui ajoutait discrètement le prix actuel des actions aux messages envoyés par l'État.

7. https://fr.wikipedia.org/wiki/Piratage_du_t%C3%A9l%C3%A9graphe_Chappe

1.0.4 Le Morse

Grâce à la découverte de l'électricité au début du XIXe siècle, et les améliorations techniques faites pour la capturer et la transmettre, on a pu utiliser le réseau électrique pour envoyer des messages. En 1832, nait le code Morse [8], qui s'impose rapidement comme un standard de communication.

Bien sûr, le Morse peut être utilisé aussi avec des signaux lumineux, ou sonores, mais la plupart du temps il est utilisé sur les lignes électriques qui se développent à l'époque.

Vous trouverez ici [9] un traducteur du langage naturel vers le Morse.

Micro-activité

Amusez-vous avec votre assistant vocal en lui demandant par exemple : «Salut Siri. Quel est le code Morse pour *j'ai envie de dormir* ?».

8. https://fr.wikipedia.org/wiki/Code_Morse_international
9. https://morsedecoder.com/

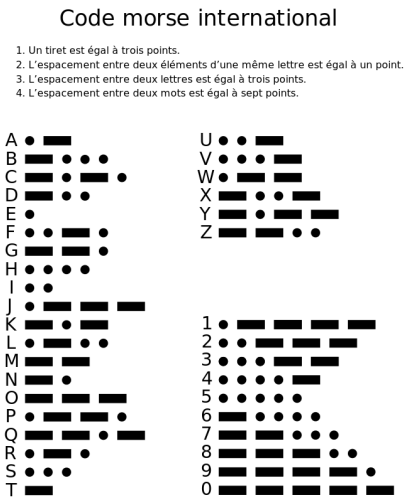


Fig. 1.3 – Le code Morse est le système de représentation de l'information qui se rapproche le plus du langage binaire de l'informatique moderne.

Si vous observez le code Morse [10], vous remarquerez que les signaux utilisés pour représenter les lettres ne suivent pas simplement l'ordre de l'alphabet, puisqu'il est plus économique de coder les lettres les plus fréquentes avec les codes les plus courts.

Le saviez-vous ?

À l'époque où les transmissions télégraphiques en code Morse sont payées à l'unité d'information, donc la lettre, des codex spécifiques sont développés par les utilisateurs pour utiliser le moins de caractères possibles. C'est exactement la même situation qui s'est produite avec l'arrivée des SMS [11] dans les années 1990, où les utilisateurs payaient au caractère. Aujourd'hui, même s'il est rare de payer à l'unité d'information, ce genre de raccourcis existent encore, mais surtout pour un avantage de vitesse.

10. https://fr.wikipedia.org/wiki/Code_Morse_international

CW Abbreviations

73	best regards	MSG	message
88	love and kisses	NR	number
ABT	about	NW	now
AGN	again	OM	old man
ANT	antenna	OP	operator
BK	break	R	are, received, roger
CPY	copy	RCVR	receiver
CQ	general call to any station	RIG	equipment
CUL	see you later	RST	readability, strength, tone report
CU	see you	SIGS	signal
DE	from	STN	station
DX	distance, rare station	TEMP	temperature
ES	and	TKS	thanks
FB	fine business	TNX	thanks
FER	for	UR	you are
FRE	frequency	U	you
Q			
GA	good afternoon, go ahead	WL	well
GE	good evening	WT	watt
GM	good morning	WX	weather
HR	here	XCVR	transceiver
HW	how	XMTR	transmitter
K	go ahead	XYL	wife

Le désavantage de ces codex d'abréviations est leur faible degré de standardisation. Comment savoir quel codex est utilisé ? Et surtout : comment faire pour que tout le monde s'accorde sur le codex ?

La réponse à cette question est l'apport le plus essentiel de l'introduction du code binaire, et des standards de représentation de l'information qui l'ont suivi : un langage pour les contrôler tous.

11. https://fr.wikipedia.org/wiki/Short_Message_Service

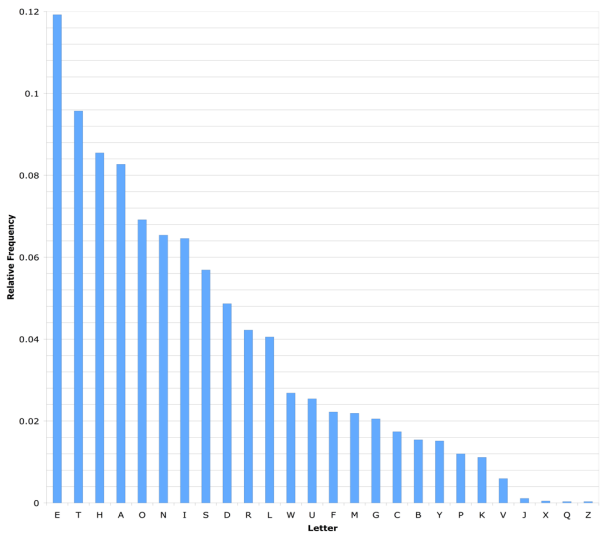


Fig. 1.4 – Ceci est une représentation de la fréquence moyenne de distribution des lettres dans la langue anglaise.

1.0.5 Le binaire

À partir du moment où le Morse [12] a été inventé comme système de *codage* et de *transmission* de l'information par l'électricité, il ne manquait plus que quelques éléments pour commencer à construire les *ordinateurs*.

Une pièce technologique, qui permettrait de *transmettre* pour ainsi dire cette information : le *transistor* (voir architecture des ordinateurs).

Un *code* plus élaboré que le Morse pour pouvoir représenter tous les types d'informations possibles à partir d'une alternative entre deux états : courant ou pas courant ; allumé ou éteint ; vrai ou faux ; 1 ou 0.

Ce *code* est le *code binaire*. Il permet, en utilisant uniquement des 0 et des 1, de représenter n'importe quel type d'information : des chiffres, du texte, des images, du son, des vidéos, etc.

Question 1

Pourquoi la lettre «e», en Morse, est-elle représentée par un seul point ?

1. Parce que c'est la lettre la plus utilisée en anglais.
2. Par hasard.
3. Parce que c'est la lettre la plus rare en anglais.
4. Parce que c'était la lettre préférée de l'inventeur du Morse.

Réponse: 1

Question 2

Que signifie informatique ?

1. Information + quantique.
2. Information + technique.
3. Information + automatique.
4. Information + pratique

Réponse: 3

12. https://fr.wikipedia.org/wiki/Code_Morse_international

1.1 Les entiers

La plupart des civilisations humaines utilise le système décimal. Pourquoi ? Tout simplement parce que nous avons 10 doigts !

L'ordinateur, lui, n'a pas de doigts mais utilise l'électricité. Par conséquent, il ne connaît que deux types d'informations : il y a du courant, il n'y a pas de courant ; allumé, éteint ; vrai, faux ; 1 ou 0.

On dit qu'il travaille dans un système binaire, ou en base deux.

1.1.1 Les systèmes de numération

Le système décimal

Ce système est composé de 10 symboles qui sont les chiffres :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Ainsi, tout nombre écrit dans la base 10 est composé de ces chiffres.

La valeur de chaque chiffre dépend alors du chiffre lui-même et de sa place. Ainsi, le 3 de 1934 et celui de 3008 n'ont pas la même valeur : Le premier vaut 30, alors que le second vaut 3000.On parle alors de **représentation positionnelle en base 10**.

Dans ce système, pour connaître la valeur de chaque chiffre qui compose un nombre, il faut décomposer ce nombre pour identifier chaque chiffre et son coefficient, c'est la **forme canonique**.

Décomposition du nombre 3528 :

- 8 unités
- 2 dizaines
- 5 centaines
- 3 milliers

Sa forme canonique est : $3 \cdot 10^3 + 5 \cdot 10^2 + 2 \cdot 10^1 + 8 \cdot 10^0$

On peut alors vérifier que le nombre 3528 est bien dans la base 10, car tous ces chiffres appartiennent à la base 10. Les nombres de la base 10 ou du système décimal sont des nombres décimaux.

Le système binaire

Le système binaire, ou numération positionnelle en base 2, est représenté à l'aide d'uniquement 2 symboles : 0 et 1.Cette représentation et la représentation décimale sont deux représentations, parmi d'autres, d'un même concept.

Un élément binaire se nomme un *bit* et un ensemble de *bits* peut représenter un entier en utilisant le même principe que pour le système décimal.

La valeur de chaque chiffre dépend toujours de sa place qui représente, cette fois, une puissance de 2.

La forme canonique du nombre binaire $1101_{(2)}$ est : $1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

Le saviez-vous ?

Le *bit* vient de la terminologie anglo-saxonne de *binary digit*. Un ensemble de 8 bits et appelé un **octet**. Un *kilo-octet* (ko) correspond à 10^3 octets soit 1000 octets, donc 8000 bits. Attention à ne pas confondre les préfixes binaires (2^{10}, 2^{20}, 2^{30}, etc.) et les préfixes décimaux (10^3, 10^6, 10^9, etc.). On appelle *kibioctet*, pour kilo binaire, une quantité de $2^{10} = 1024$ octets. On peut remarquer que cette notation, quoique rigoureuse, peine à s'imposer dans le vocabulaire courant des ingénieurs eux-même…

Compter en binaire

On compte en binaire de la même manière que l'on compte en base 10 en ajoutant 1 aux unités (position la plus à droite). Lorsqu'on arrive au dernier chiffre (i.e. 9 en base 10 et 1 en base 2), on le remet à 0 et on augmente de 1 le chiffre à sa gauche.

On répète ces opérations pour tous les chiffres, quelle que soit leur position. Ainsi, en base 10 :

$$0; -; 1; -; 2; -; 3; -; ...; -; 9; -; 10; -; 11; -; ...; -; 99; -; 100; -; 101; -; ...$$

En binaire, on obtient : $0; -; 1; -; 10; -; 11; -; 100; -; 101; -; 110; -; 111; -; 1000; -; ...$

Micro-activité

Comptez jusqu'à 40 en binaire. Que pouvez vous observer au sujet de la parité des nombres binaires ? Pourquoi ?

Conversion du système binaire vers le système décimal

La conversion d'un nombre binaire en nombre en base 10 se fait aisément grâce à la forme canonique.

En effet, il suffit de calculer le résultat de la somme pondérée par les puissances de 2.

Conversion du nombre 10101

$$10101_{(2)} = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 21_{(10)}$$

Le *tableau* ci-dessous permet de convertir un octet en nombre décimal.

Coefficient	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valeur	128	64	32	16	8	4	2	1
Bit	0	0	1	0	1	0	1	0

L'exemple utilisé ici est l'octet $(00101010_{(2)})$ dont la valeur décimale est : $00101010\_(2) = 0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 42\_(10)$

Important

L'utilisation d'un tableau de conversion nécessite d'écrire le nombre binaire de droite à gauche car le bit de poids faible ($= 2^0$) se trouve à droite, de la même façon que le chiffre de poids faible (=l'unité) se trouve à droite en représentation décimale.

Micro-activité

Donnez la conversion en base 10 des nombres binaires suivants :

- 10101101
- 01110010
- 1111
- 1111111
- 1
- 10
- 100
- 1000
- 10000
- 100000
- 1000000
- 10000000

Conversion du système décimal vers le système binaire

L'opération de conversion du système décimal vers le système binaire est moins directe. Cependant, à l'aide d'un tableau de conversion et des instructions suivantes, il est possible d'obtenir la représentation binaire de n'importe quel entier positif.

Tableau de conversion

Coefficient	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valeur	1024	512	256	128	64	32	16	8	4	2	1
Bit											

Instructions de conversion d'un entier du système décimal vers le système binaire

1. Déterminer le coefficient **maximum** dont la valeur est plus petite que l'entier à convertir.
2. Le bit associé à ce coefficient est 1.
3. Soustraire la valeur de ce coefficient à l'entier à convertir pour obtenir le nouveau nombre à convertir.
4. Recommencer à l'étape 1 tant que le nombre est différent de 0.
5. En commençant par le plus grand coefficient utilisé, le nombre binaire correspondant est composé de la suite des bits où des 0 représentent les coefficients non utilisés.

Par exemple, la conversion du nombre 666 en base 10 vers le binaire s'obtient avec les étapes suivantes :

$$666 = 512 + 154 \tag{1.1}$$
$$154 = 128 + 26 \tag{1.2}$$
$$26 = 16 + 10 \tag{1.3}$$
$$10 = 8 + 2 \tag{1.4}$$
$$2 = 2 + 0 \tag{1.5}$$

Valeur	1024	512	256	128	64	32	16	8	4	2	1
Bit		1	0	1	0	0	1	1	0	1	0

Résultat : $(666_{(10)} = 1010011010_{(2)})$

Micro-activité

Donnez la conversion binaire des nombres décimaux suivants :

- 97
- 123
- 256
- 1000
- 511

Pour aller plus loin

Pouvez-vous penser à une autre façon de convertir un entier du système décimal en binaire ?

Le saviez-vous ?

Le 4 juin 1996, le premier vol de la fusée Ariane 5 a explosé 40 secondes après l'allumage. La fusée et son chargement avaient coûté 500 millions de dollars. La commission d'enquête a rendu son rapport au bout de deux semaines. Il s'agissait d'une erreur de programmation dans le système inertiel de référence. À un moment donné, un nombre codé en virgule flottante sur 64 bits (qui représentait la vitesse horizontale de la fusée par rapport à la plate-forme de tir) était converti en un entier sur 16 bits. Malheureusement, le nombre en question était plus grand que 32767 (le plus grand entier que l'on peut coder en tant qu'entier signé sur 16 bits) et la conversion a été incorrecte, induisant un changement de trajectoire fatal.

1.1.2 Représentation des entiers négatifs

Après la représentation des entiers naturels $(\mathbb{N})$, on souhaite évidement pouvoir représenter les entiers négatifs afin d'avoir une représentation des entiers relatifs $(\mathbb{Z})$.Un entier relatif est un entier naturel auquel on a ajoute un signe positif ou négatif indiquant sa position **relative** par rapport à 0.

Bit de signe

La première idée pour représenter un entier relatif est d'utiliser un bit pour représenter le signe. En effet, un bit peut uniquement prendre deux valeurs, 0 ou 1, comme le signe, + ou -. Les bits restants étant utilisés pour représenter un entier positif.Considérons l'encodage sur un octet (8 bits), nous réservons le bit de poids fort (la puissance de 2 la plus grande) pour le signe : 0 indique un nombre positif et 1 un nombre négatif.

Avec cette approche, un entier relatif est représenté par sa valeur absolue (entier naturel) auquel on associe un signe. Le domaine couvert se trouve donc divisé par deux, un bit étant utilisé pour le signe.Ainsi, avec un octet, 7 bits sont utilisés pour encoder la valeur absolue, soit $[0, 127]$, ce qui permet de représenter les entiers relatifs dans l'intervalle $[-127, 127]$.

La représentation de -21 est :

signe	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	0	1	0	1	0	1

Bien que cette approche soit simple et semble convenir, elle pose deux problèmes majeurs :

— le premier est d'avoir deux représentations pour zéro (+0 et -0),

— le second apparait lorsque l'on additionne un nombre et son opposé. Le résultat de cette opération devrait être 0, mais, sans rentrer dans le détail de l'arithmétique binaire qui sera abordée plus tard, l'addition bit à bit avec cette représentation donne $(-2|x|)$. En effet, l'addition des bits de signe donne toujours 1 et le reste des bits représente l'addition de deux fois la valeur absolue.

Pour remédier ces problèmes, une autre représentation des entiers relatifs a été mise au point, il s'agit de la représentation en complément à deux.

Complément à deux

La représentation en complément à deux reprend l'idée du signe encodé par le bit de poids fort et conserve la représentation des entiers positifs. Donc tous les entiers positifs ont une représentation en binaire qui commence par un 0 et le plus grand entier représenté sur un octet est 127.

Les entiers négatifs sont représentés grâce au complément à deux dont voici le principe :

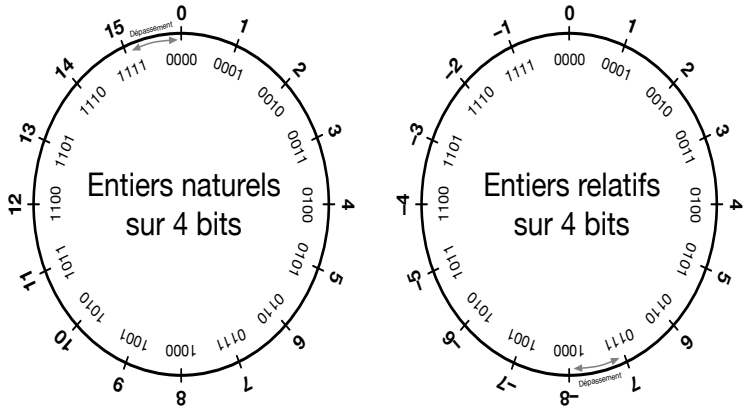
1. Écrire la valeur absolue du nombre en binaire (le bit de poids fort est égal à 0).
2. Inverser tous les bits, les 0 deviennent des 1 et vice-versa. Le résultat obtenu s'appelle le complément à 1 du nombre de départ.
3. On ajoute 1, la dernière retenue est ignorée le cas échéant.

La représentation de -21 en complément à 2 :

1. Valeur absolue en binaire : 00010101

2. Complément à 1 : 11101010
3. Ajouter 1 : 11101011

La représentation de -21 est 11101011, qui additionné à 21, soit 00010101 donne bien zéro : 00000000.



Représentation des entiers avec 4 bits. La figure ci-dessus illustre la différence du domaine couvert avec 4 bits pour la représentation des entiers naturels ou des entiers relatifs. Ainsi, avec 4 bits le domaine couvert pour les entiers naturels est : [0, 15], et pour les entiers relatif : [-8, 7].

À retenir

Puisque le nombre d'entiers relatifs représentés est forcément pair et que le 0 en fait partie, il y a une asymétrie entre les nombres positifs et négatifs représentés. Par exemple, avec 4 bits on peut représenter -8, **mais pas 8**

Quel est le domaine couvert en utilisant la représentation en complément à deux sur un octet ?

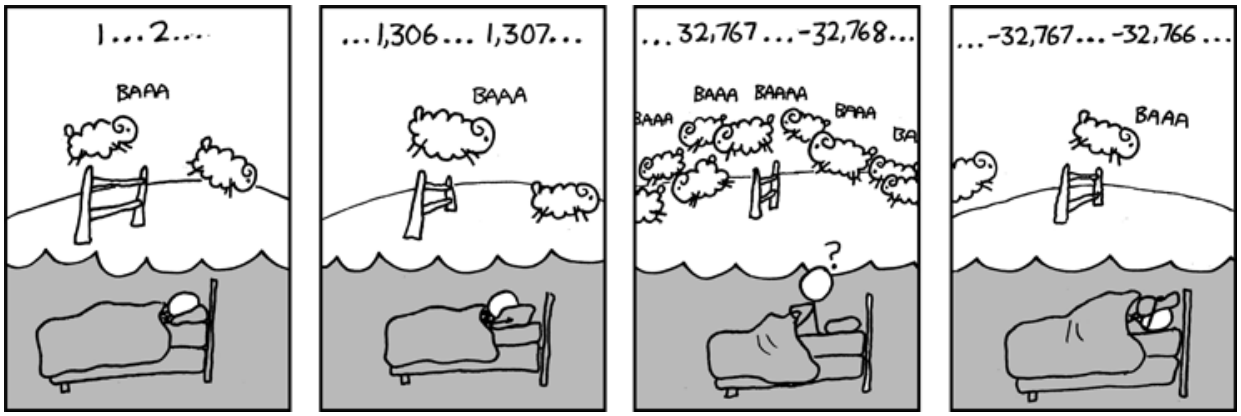
$[-128, 127]$

Micro-activité

Encodez les entiers relatifs suivants sur un octet :

— -99
— -1
— 127
— -128

Vous pouvez maintenant comprendre ce comic d'un robot comptant des moutons pour s'endormir… d'ailleurs, combien de bits utilise-t-il pour compter ? ?



Source : xkcd [13]

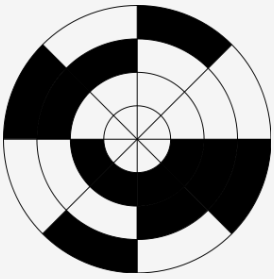
Pour aller plus loin

Le code binaire réfléchi, ou code Gray, est un type de codage binaire ne modifiant qu'un seul bit à la fois quand un nombre est augmenté d'une unité. Ce type de codage est utilisé pour les codeurs rotatifs absolus calibrés et initialisés une seule fois **qui doivent conserver leur valeur** lors de l'arrêt de l'appareil, comme les compteurs kilométriques des automobiles (à la différence du compteur journalier qui peut être remis a zéro par l'utilisateur).

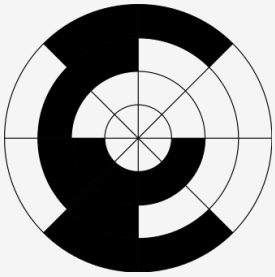
Voici le début de la table de correspondance décimal-binaire-Gray sur quatre bits (huit premières valeurs) :

Codage décimal	Codage binaire	Codage binaire réfléchi
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

Chaque encodage peut être représenté sur un disque divisé en huit secteurs identiques :



13. https://xkcd.com/571/



1 - Etablir la correspondance entre la table binaire et la figure de l'encodage binaire, puis entre la table Gray et la figure correspondante.

2 - En comprenant la construction des huit valeurs données dans la table de correspondance (correspondant donc aux huit secteurs des disques), tenter d'écrire la table complète sur quatre bits.

3 - Combien de secteurs angulaires les disques vont-ils comprendre pour un codage sur quatre bits ? Représenter alors le disque binaire puis Gray sur quatre bits.

1.2 Les caractères

Toute l'information est représentée dans un ordinateur par des nombres encodés sous forme binaire par des 0 et des 1. Se pose alors la question de la représentation des caractères, ne serait-ce que parce que la communication entre les utilisateurs et les ordinateurs s'opère essentiellement sous forme textuelle.

1.2.1 Principe

La solution est simple : on associe chaque caractère à un code binaire.

Caractère	Décimal	Hexadécimal	Binaire
A	65	0x41	01**0**00001
B	66	0x42	01**0**00010
C	67	0x43	01**0**00011
…	…	…	…
Z	90	0x5A	01**0**11010

Chaque caractère frappé sur le clavier est représenté par le code correspondant dans ce tableau.

Chacun des caractères de la phrase que vous lisez (qu'on nomme **chaîne de caractères**) a ainsi été stocké, transmis et manipulé par l'ordinateur sous la forme d'une séquence de 0 et 1.

Lorsqu'il s'agit de représenter ce texte à l'écran ou à l'impression, les logiciels utilisent la table dans l'autre sens pour trouver le caractère correspondant au nombre binaire.

En plus des lettres, les caractères qui représentent les chiffres sont eux-mêmes listés dans la table de conversion. Contre-intuitivement, la valeur binaire du caractère représentant un chiffre ne correspond pas au chiffre lui-même.

Caractère	Décimal	Hexadécimal	Binaire
0	48	0x30	00110000
1	49	0x31	00110001
…	…	…	…
9	57	0x39	00111001

Ces tables donnent également une représentation des caractères de ponctuation et des symboles mathématiques, ainsi que des caractères non-imprimables comme le retour à la ligne.

En réalité, il n'existe pas une table de conversion unique, mais des dizaines de tables de conversion. Certaines tables ont été proposées à l'origine par des constructeurs d'ordinateurs ou des éditeurs de systèmes d'exploitation.

1.2.2 Table ASCII

Le code américain normalisé pour l'échange d'information ASCII (pour American Standard Code for Information Interchange) est apparu dans les année 1960. Malgré sa large acceptation, avec ses **7 bits par caractère**, cette table avait pour principal défaut de ne pas prendre en compte les caractères qui n'existent pas dans la langue anglaise, ne serait-ce que les lettres accentuées.

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*
1	1	1	1	*[START OF HEADING]*
2	2	10	2	*[START OF TEXT]*
3	3	11	3	*[END OF TEXT]*
4	4	100	4	*[END OF TRANSMISSION]*
5	5	101	5	*[ENQUIRY]*
6	6	110	6	*[ACKNOWLEDGE]*
7	7	111	7	*[BELL]*
8	8	1000	10	*[BACKSPACE]*
9	9	1001	11	*[HORIZONTAL TAB]*
10	A	1010	12	*[LINE FEED]*
11	B	1011	13	*[VERTICAL TAB]*
12	C	1100	14	*[FORM FEED]*
13	D	1101	15	*[CARRIAGE RETURN]*
14	E	1110	16	*[SHIFT OUT]*
15	F	1111	17	*[SHIFT IN]*
16	10	10000	20	*[DATA LINK ESCAPE]*
17	11	10001	21	*[DEVICE CONTROL 1]*
18	12	10010	22	*[DEVICE CONTROL 2]*
19	13	10011	23	*[DEVICE CONTROL 3]*
20	14	10100	24	*[DEVICE CONTROL 4]*
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*
22	16	10110	26	*[SYNCHRONOUS IDLE]*
23	17	10111	27	*[ENG OF TRANS. BLOCK]*
24	18	11000	30	*[CANCEL]*
25	19	11001	31	*[END OF MEDIUM]*
26	1A	11010	32	*[SUBSTITUTE]*
27	1B	11011	33	*[ESCAPE]*
28	1C	11100	34	*[FILE SEPARATOR]*
29	1D	11101	35	*[GROUP SEPARATOR]*
30	1E	11110	36	*[RECORD SEPARATOR]*
31	1F	11111	37	*[UNIT SEPARATOR]*
32	20	100000	40	*[SPACE]*
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	*[DEL]*

Tab. 1 La table de représentation des caractères ASCII

Des tables multiples, mutuellement incompatibles, ont alors émergé : une table pour les européens, une autre pour les Japonais et ainsi de suite.

Progressivement, notamment avec l'émergence du Web au cours des années 1990, l'augmentation de l'interconnexion des ordinateurs personnels a amené au début des années 2000 à la mise en place d'une énorme table intégrant le contenu de toutes les tables existantes, via le standard UTF.

1.2.3 Standard UTF

Le standard Unicode [14] UTF (Universal Character Set Transformation Format) s'est imposé pour l'échange, car il permet d'agréger sur 8 bits, 16 bits ou 32 bits par caractère la totalité des caractères utilisés dans toutes les langues humaines… et même extraterrestres, puisque le Klingon [15] est également intégré.

Les caractères liés à l'édition des partitions de musique ou les émojis sont également intégrés.

Variantes

Pour éviter de consommer 32 bits par caractère, des variantes plus compactes ont été mises à disposition.

La plus connue – des européens, puisqu'elle regroupe les caractères qui nous concernent – est la table UTF-8 [16]. Elle se concentre sur les premiers 8 bits de la table UTF complète. Par sa nature, UTF-8 est d'un usage très répandu sur internet et dans les systèmes échangeant de l'information. Il s'agit également du codage le plus utilisé dans les systèmes de logiciels libres pour gérer le plus simplement possible des textes et leurs traductions dans tous les systèmes d'écritures et alphabets du monde. Les navigateurs internet d'aujourd'hui utilisent le codage UTF-8 et les concepteurs de sites prenent en compte cette même norme ; c'est pourquoi il y a de moins en moins de problèmes de *compatibilité*. Toutefois, toutes ces différentes normes et leurs incompatibilités sont la cause des problèmes que l'on rencontre parfois avec les caractères accentués. Il est donc préférable pour la rédaction de courriels à l'étranger, de n'utiliser que des caractères non accentués.

UTF-8 est donc un encodage des caractères basé sur UNICODE, de longueur variable qui utilise de 1 à 4 octets par symbole.

Comment s'opère le codage sur plusieurs octets ?

En UTF-8, chaque point de code de 0 à 127 est stocké dans un seul octet. Seuls les points de code 128 et supérieurs sont stockés en utilisant 2, 3 ou 4 octets. Chaque octet commence alors par quelques bits qui indiquent s'il s'agit d'un point de code à un octet, d'un point de code à plusieurs octets ou de la continuation d'un point de code à plusieurs octets :

0xxx xxxx : ***c'est un code US-ASCII à un seul octet (permettant donc d'encoder les 127 premiers caractères).***

Les points de code multi-octets commencent chacun par quelques bits à 1 du premier octet en partant de la gauche, suivis d'un bit à 0, et qui vont dire si l'on doit lire l'octet suivant, ou les deux ou les trois suivants, pour comprendre l'encodage global. Par exemple, si l'octet le plus à gauche s'écrit :

110x xxxx : ***cela indique que le message global est encodé sur 1+1=2 octets, et donc qu'un deuxième octet suit.***

1110 xxxx : ***cela indique que le message global est encodé sur 1+1+1=3 octets, et donc qu'un deuxième puis un troisième octet suivent.***

1111 0xxx : ***cela indique que le message global est encodé sur 4 octets, et donc qu'un deuxième puis un troisième puis un quatrième octet suivent.***

14. https://home.unicode.org/
15. https://www.kli.org/about-klingon/klingon-history/
16. https://www.utf8-chartable.de/

Enfin, les octets qui suivent ces codes de démarrage sont tous de la forme : 10xx xxxx. Les bits représentés par le caractère «x» représentent ce que l'on appelle la *charge utile*, c'est à dire l'encodage du caractère proprement dit.

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

Tab. 2 Définition du nombre d'octets utilisés

Puisqu'on peut dire quel type d'octet on regarde à partir des premiers bits du premier octet à gauche, alors même si quelque chose est altéré quelque part, la séquence entière n'est pas perdue : ce codage est appelé *codage auto-synchronisant*.

Codage UTF-8 en détail

Le premier octet en partant de la droite sert lui à encoder les caractères ASCII, donnant ainsi au jeu de caractères une **totale compatibilité avec ASCII**.

Chaque caractère non ASCII (c'est à dire dont le point de code - ici le codage décimal - est supérieur à 127) se code nécessairement sur plusieurs octets, entre 2 et 4 octets ; les bits de poids fort du premier octet en partant de la gauche forment, en partant de la gauche également, une suite de 1 de longueur égale au nombre total d'octets utilisés pour coder le caractère ; les octets suivants auront 10 comme bits de poids fort comme on vient de l'écrire.

Reprenons la table ASCII de la figure 8 et la table UTF-8 [17] : on observe que le signe ~ par exemple est sur la table ASCII à l'adresse décimale 126 (01111110 en binaire), et sera donc à la même adresse sur la table UTF-8. Même chose pour le caractère suivant, qui est le caractère de contrôle *del* qui se trouve à l'adresse 127 (01111111 en binaire). En revanche, le caractère suivant, qui est également un caractère de contrôle, bien évidemment n'apparaît plus sur la table ASCII ; sur la table table UTF-8 [18], l'adresse décimale est 194 128.

Si l'on prend à présent, par exemple, le caractère «æ», on lit sur la table UTF-8 : 195 166, soit, en binaire : 11000011 10100110.

On constate bien le passage du codage sur deux octets. L'adresse décimale 195 du premier octet correspond a la valeur binaire 11000011. On retrouve la suite de deux "1" en début de ce premier octet en partant de la gauche, indiquant ce codage total sur deux octets ; il reste 000011 pour la charge utile du premier octet du codage UTF-8. L'adresse décimale de 166 est 10100110 et commence donc bien par 10 comme bits de poids fort ; la charge utile du deuxième octet du codage UTF-8 est donc 100110. L'encodage binaire UTF-8 global s'écrit donc, en concaténant les deux charges utiles : 000011100110, ce qui correspond à 230 en décimal, valeur qu'on peut vérifier sur cette autre table UTF-8 [19] indiquant également le codage décimal.

17. https://www.utf8-chartable.de/unicode-utf8-table.pl?number=512&utf8=dec
18. https://www.utf8-chartable.de/unicode-utf8-table.pl?number=512&utf8=dec
19. https://kellykjones.tripod.com/webtools/ascii_utf8_table.html

Caractère	Numéro du caractère	Codage binaire UTF-8
A	65	01000001
é	233	11000011 10101001
€	8364	11100010 10000010 10101100
𝄞	119070	11110000 10011101 10000100 10011110

Tab. 3 Définition du nombre d'octets utilisés

Par exemple le caractère « € » (euro) est le 8365e caractère du répertoire Unicode ; son index, ou point de code, est donc 8364, il se code en UTF-8 sur 3 octets : 226, 130, et 172 exprimé en décimal (11100010 10000010 10101100 exprimé en binaire).

Type	Caractère	Point de code (hexadécimal)	Valeur scalaire		Codage UTF-8	
			décimal	binaire	binaire	hexadécimal
Contrôle	[NUL]	U+0000	0	0	00000000	00
	[US]	U+001F	31	1·1111	00011111	1F
Texte	[SP]	U+0020	32	10·0000	00100000	20
	A	U+0041	65	100·0001	01000001	41
	~	U+007E	126	111·1110	01111110	7E
Contrôle	[DEL]	U+007F	127	111·1111	01111111	7F
	[PAD]	U+0080	128	*000* 1000·0000	11000010 10000000	C2 80
	[APC]	U+009F	159	*000* 1001·1111	11000010 10011111	C2 9F
Texte	[NBSP]	U+00A0	160	*000* 1010·0000	11000010 10100000	C2 A0
	é	U+00E9	233	*000* 1110·1001	11000011 10101001	C3 A9
	[illegible]	U+07FF	2047	*111* 1111·1111	11011111 10111111	DF BF
	[illegible]	U+0800	2048	*1000* 0000·0000	11100000 10100000 10000000	E0 A0 80
	€	U+20AC	8 364	*10·0000* 1010·1100	11100010 10000010 10101100	E2 82 AC
	[illegible]	U+D7FF	55 295	*1101·0111* 1111·1111	11101101 10011111 10111111	ED 9F BF

Tab. 4 Extrait de la table de représentation UTF-8

1.2.4 Exercices

Exercice 1 – Utilisation de la table ASCII

1 - À l'aide de la table ASCII, codez en binaire la phrase suivante «L'an qui vient !».

2 - Voici maintenant une exclamation codée en binaire : 01000010 01110010 01100001 01110110 01101111 00100001. Retrouvez cette exclamation !

3 - Peut-on coder en binaire la phrase «Un âne est-il passé par là ?» à l'aide de la table ASCII (justifiez la réponse) ?

Exercice 2 – Activité codage et internet

Ouvrez un navigateur Internet (Firefox, …). Dans la barre d'outils, on peut voir à «Affichage», «Encodage des caractères» que c'est le format UTF-8 qui est sélectionné par défaut.

1 - Changez la sélection UTF-8 et choisissez à présent Europe Centrale (Windows). De petits caractères désagréables apparaissent. Que s'est-il passé ?

2 - Utilisez toujours le navigateur web, et allez dans «Affichage», «Source». On lit alors l'entête de la page *html* visitée. Où se situe l'information relative à l'encodage ?

3 - On peut aussi dans «Affichage», «Codage», sélectionner Grec (ISO) et se rendre compte en lisant le texte, que le «à» a été remplacé par un «L» à l'envers dit *Gamma*.

Exercice 3 – Coder en UTF-8

Le symbole Ø correspond à la valeur décimale 8709.

1 - Convertissez cette valeur en binaire.

2 - Combien d'octets doit-on utiliser en UTF-8 pour coder ce nombre convenablement (les moitiés d'octet sont interdites) ?

3 - Donnez le codage UTF-8 correspondant.

Micro-activité – Hexadécimal

Nous avons vu au cours du chapitre précédent deux systèmes de numération, décimal et binaire. Il existe également un troisième système de numération très utilisé, le système hexadécimal, visible par ailleurs sur les tables. Le système binaire permet d'exprimer n'importe quel nombre en base 2 (soit 0, soit 1), le système décimal en base 10 (de 0 à 9) - c'est notre mode de représentation usuel. Le système hexadécimal permet d'exprimer n'importe quel nombre en base 16 : de 0 à 9… puis les lettres A, B, C, D, E, F.

1 - Selon vous, comment s'expriment les nombres décimaux 6, 8, 11, 14 et 16 en hexadécimal ?

2 - Exprimer 34 puis 128 en hexadécimal.

3 - A quels nombres décimaux correspondent les nombres hexadécimaux 80, puis 9A ?

4 - En prenant la valeur décimale 154, essayez de décrire une méthode permettant de passer du système décimal au système hexadécimal.

5 - En reprenant la valeur hexadécimale 9A, essayez de décrire une méthode permettant de passer du système hexadécimal au système décimal.

1.3 Les images

1.3.1 Les images matricielles

Depuis des siècles les humains gardent des traces de leur environnement sous forme d'images. Plus le temps passe, plus ces traces sont fidèles. On découvre par exemple la perspective autour du XVe siècle, les progrès en optique et en chimie permettent ensuite la création de la camera obscura et de la photographie argentique. Enfin l'informatique se développe permettant l'invention de la photographie numérique.

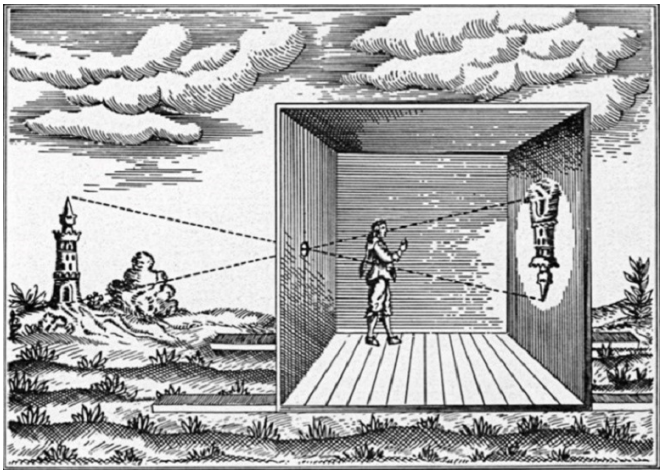


Fig. 1.5 – Principe de fonctionnement de la camera obscura.

Fig. 1.6 – Une caméra obscura.

De la camera obscura à la caméra numérique

Comment fonctionne une caméra numérique ? Une caméra numérique fonctionne en fait d'une manière très similaire à la caméra obscura et aux appareils photographiques analogiques d'un point de vue optique. Imaginez une chambre noire pourvue d'un trou sur l'une de ses parois. La lumière venant de l'extérieur vient se projeter sur le mur opposé.

Dans un appareil analogique, la paroi illuminée est recouverte d'une pellicule chimique photosensible qui permet de capturer l'image.

La différence est que dans un appareil photo numérique cette paroi, le capteur photographique, est recouverte d'une grille de capteurs électroniques photosensibles (photosites) produisant de l'électricité quand ils reçoivent de la lumière. Chaque photosite est recouvert d'un filtre coloré ne laissant passer que les rayons d'une seule couleur (grille de Bayer) : le rouge, le vert ou le bleu. Les filtres sont répartis par carré de quatre : deux verts, un rouge et un bleu. La tension électrique produite par chaque photosite est convertie numériquement et transmise au processeur de l'appareil photo.

L'image numérique ne sera alors rien d'autre que la collection des mesures de tous les capteurs à un temps précis. Comme ces mesures sont organisées sous forme de tableau (grille), on parle souvent d'images matricielles. Plus le nombre de capteurs est grand, plus la définition de cette image le sera aussi.

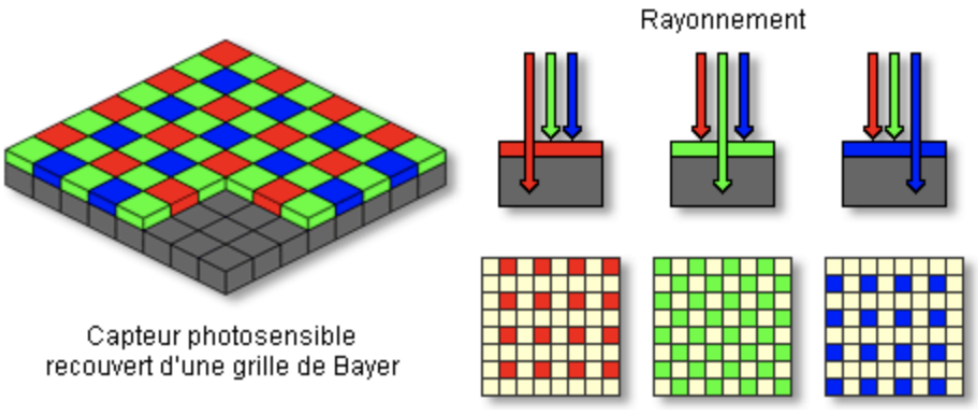
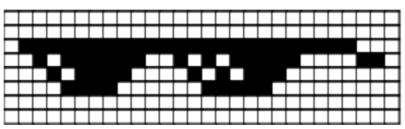


Fig. 1.7 – Principe de la capture numérique d'une image.

1.3.2 Représentation d'une image en noir et blanc

Image



Code

```
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0],
[0,0,1,0,1,1,1,1,1,1,0,0,1,0,1,1,1,1,1,1,0,0,0,0,1,1,0],
[0,0,0,1,0,1,1,1,1,0,0,0,0,1,0,1,0,1,1,1,0,0,0,0,0,0,0],
[0,0,0,0,1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
```

Un bit est l'unité minimale d'information qu'un ordinateur manipule : 1 ou 0, allumé ou éteint. L'image la plus simple qu'un ordinateur puisse afficher est constituée uniquement de noir et blanc. Ainsi, un pixel pourrait être à l'état soit «noir», soit «blanc».

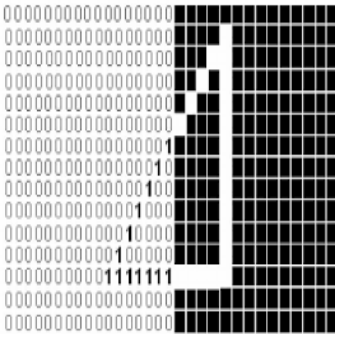


Fig. 1.8 – Tous les pixels marqués d'un 1 s'affichent en blanc, tous ceux marqués d'un 0 s'affichent en noir. Ceci nous permet de construire des images simples, dessinées seulement en noir et blanc.

Un *pixel*, de l'anglais "**pic**ture **el**ement", est le composant minimal d'une image. C'est à dire que c'est le plus petit élément avec lequel on construit une image sur un écran d'ordinateur. Dans notre exemple minimaliste, chaque pixel peut être soit noir, soit blanc, ce qui nous permet de construire une image.

1.3.3 Représentation d'une image en niveaux de gris

Dans ce type d'image seul le niveau de l'intensiteé est codeé sur un octet (256 valeurs). Par convention, la valeur 0 repreésente le noir (intensiteé lumineuse nulle) et la valeur 255 le blanc (intensiteé lumineuse maximale) :

0	*8*	*16*	*32*	*56*	*72*	*90*	*104*	*112*	*128*
136	*144*	*160*	*176*	*192*	*208*	*224*	*244*	*248*	*255*

Fig. 1.9 – Niveaux de gris, codage sur 8 bits.

En général, les images sont représentées sous forme de tableau numérique, aussi appelé format *matriciel*. Une image en niveau de gris sera ainsi représentée par un tableau de valeurs correspondant à la *luminance* de chaque pixel. Les valeurs de luminance sont des nombres allant de 0 (noir) à 255 (blanc). Pour encoder une image en niveaux de gris, chaque pixel nécessite donc 8 bits.

Pour accéder à un pixel particulier, il faut indiquer à quelle ligne et à quelle colonne de l'image ce pixel se trouve. Le pixel (0,0) correspondra normalement au pixel de la première ligne et de la première colonne.

Le saviez-vous ?

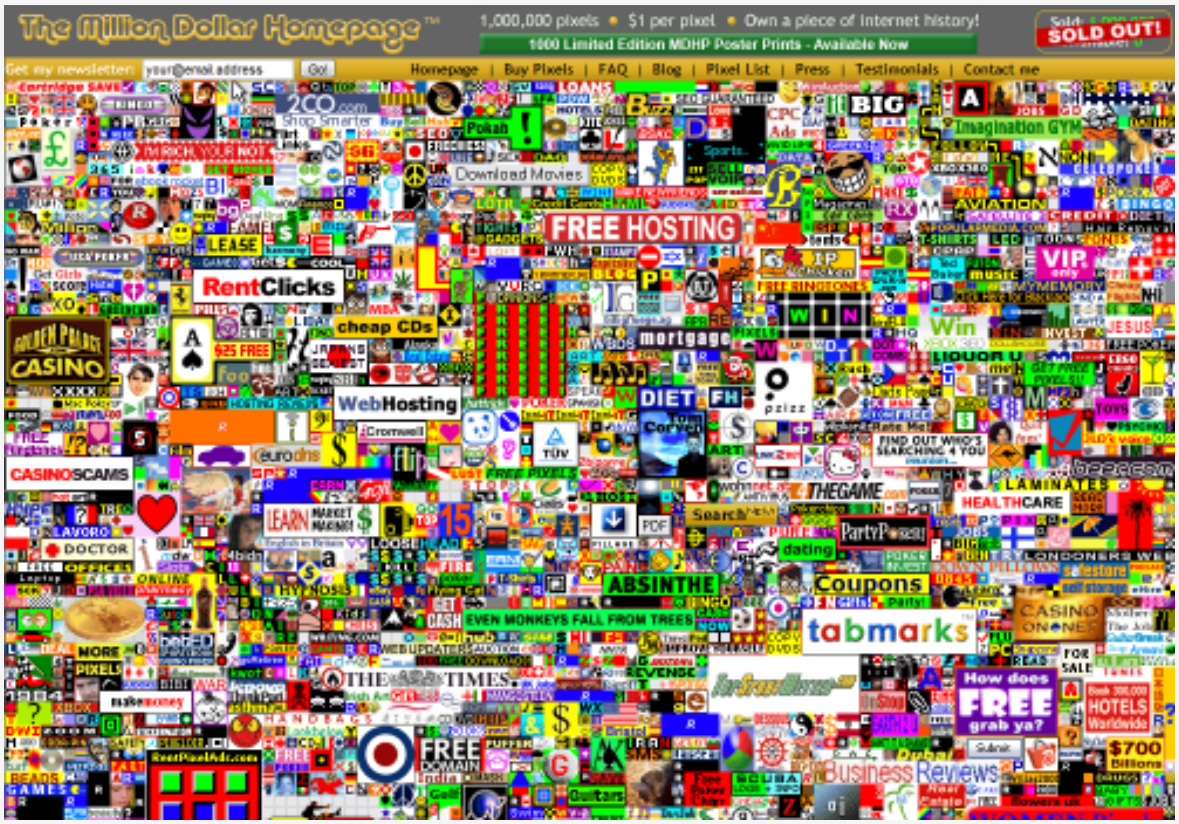
Ce mode de fonctionnement est similaire à celui des tableurs pour lesquels il est possible d'accéder à la valeur d'une case en utilisant sa référence. On pourrait d'ailleurs utiliser le formatage conditionnel pour transformer un tableau de valeurs dans un tableur en image matricielle.

Fig. 1.10 – Image monochrome, pixels et luminance.

1.3.4 Représentation d'une image en couleurs

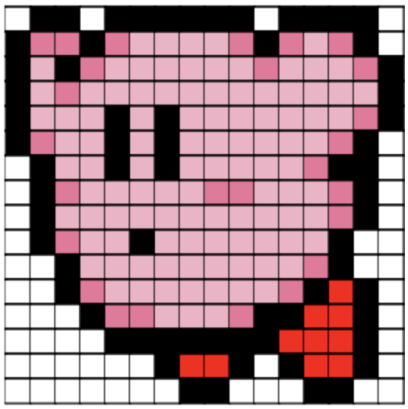
Le saviez-vous ?

The Million Dollar Homepage [20] est un site web conçu en 2005 par Alex Tew, un étudiant anglais, dans le but de financer ses études supérieures. La page d'accueil est une grille de 1000 × 1000 pixels. Chaque pixel était vendu 1$ en tant qu'espace publicitaire. Ils ont tous été vendus…



20. https://fr.wikipedia.org/wiki/The_Million_Dollar_Homepage

Image



Code

```
[0,1,1,0,1,1,1,1,1,1,0,1,1,1,1,0],
[1,(238,116,155),(238,116,155),1,(238,116,155),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(238,116,155),1,(238,116,
155),(243,178,200),(238,116,155),1,0],
[1,(243,178,200),1,(238,116,155),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(238,116,155)
,(243,178,200),(243,178,200),(243,178,200),(238,116,155),1],
[1,(243,178,200),(238,116,155),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),
(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),1],
[1,(243,178,200),(243,178,200),(243,178,200),1,(243,178,200),1,(243,178,200),(243,178,200),(243,178,200),(243,178,
200),(243,178,200),(243,178,200),(238,116,155),1],
[1,(238,116,155),(243,178,200),(243,178,200),1,(243,178,200),1,(243,178,200),(243,178,200),(243,178,200),(243,178,
200),(243,178,200),(238,116,155),1,0],
[0,1,(243,178,200),(243,178,200),1,(243,178,200),1,(243,178,200),(243,178,200),(243,178,200),(243,178,200),(238,116,
155),1,1,0],
[0,1,(238,116,155),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(238,116,155),(238,116,155),(243,178,200)
,(243,178,200),(243,178,200),(238,116,155),1,0],
[0,1,(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200)
,(243,178,200),(243,178,200),(238,116,155),1,0],
[0,1,(238,116,155),(243,178,200),(243,178,200),1,(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,
200),(243,178,200),1,0,0],
[0,0,1,(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,
200),(238,116,155),1,0,0],
[0,0,1,(238,116,155),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(243,178,200),(238,116,
155),1,(255,0,0),1,0],
[0,0,0,1,(238,116,155),(238,116,155),(243,178,200),(243,178,200),(243,178,200),(238,116,155),1,1,(255,0,0),(255,0,0),1,0],
[0,0,0,0,1,1,1,1,1,1,(255,0,0),(255,0,0),(255,0,0),1,0],
[0,0,0,0,0,0,1,(255,0,0),(255,0,0),1,0,1,(255,0,0),(255,0,0),1,0],
[0,0,0,0,0,0,0,1,1,0,0,0,1,1,0,0],
```

En peinture, pour obtenir toutes les couleurs de l'arc-en-ciel, on utilise un mélange de magenta, de cyan et de jaune, qui vont chacune absorber une partie de la lumière ; c'est ce que l'on appelle la *synthèse soustractive* : en ajoutant du pigment à une surface, une partie du spectre lumineux est soustraite.

Pour faire la même chose sur un écran, on utilisera également trois couleurs, mais celles-ci seront le rouge, le vert et le bleu (couleurs primaires). Cela correspond à la *synthèse additive* : en allumant une LED rouge par exemple, on ajoute de la lumière sur la partie du spectre lumineux correspondant.

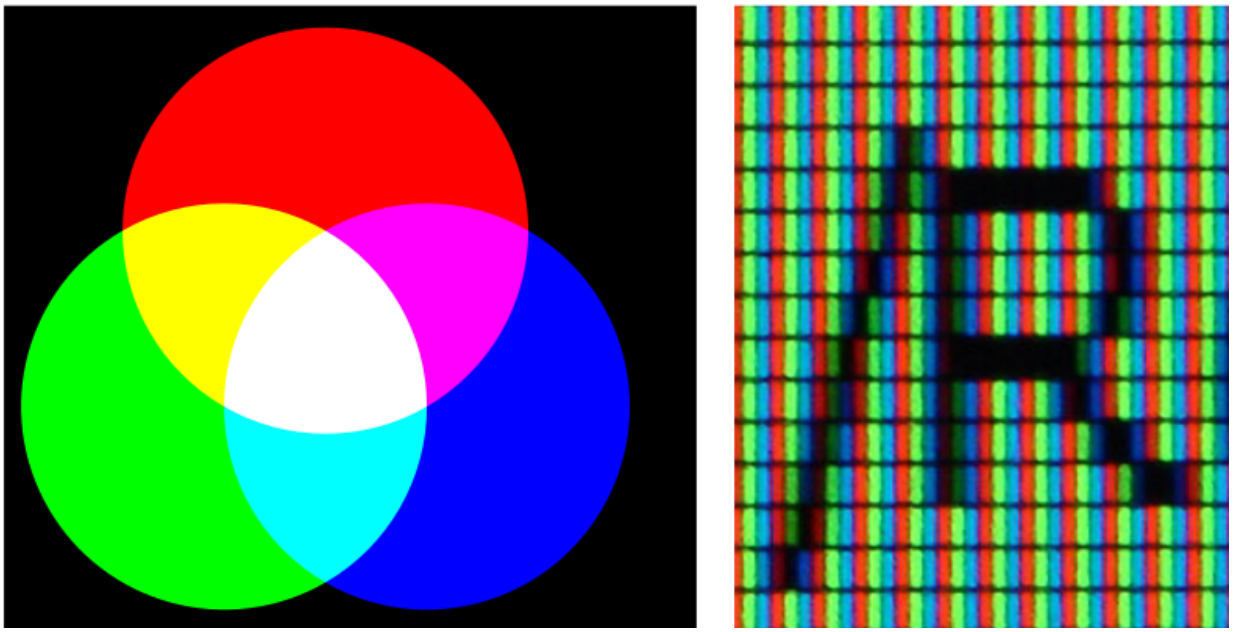


Fig. 1.11 – Système additif et écran au microscope.

Chaque couleur est donc représentée comme un mélange de ces trois couleurs et donc sous forme de trois entiers (triplet). Comme pour les images en niveaux de gris, ces entiers sont généralement représentés sur 8 bits ; les valeurs de luminance sont chacune déclarées comme un nombre allant de 0 (intensité nulle) à 255 (intensité maximale). Pour représenter une image en couleurs il faut donc 8 bits pour le niveau de rouge, 8 bits pour le niveau de vert, et 8 bits pour le niveau de bleu, soit 24 bits.

Dans cette animation [21] vous pouvez zoomer sur chacun des pixels qui constituent l'image totale. Chaque pixel possède trois valeurs allant de 0 à 255. RGB signifie en anglais Red, Green, Blue.

Dans cette autre animation [22] vous pouvez jouer avec la valeur de Rouge, Vert, Bleu, pour créer une couleur finale. L'outil vous permet d'abord de jouer avec des couleurs codées en 24 bits, puis en 8 bits, ce qui illustre bien la précision qu'on arrive à atteindre avec 24 bits.

Les formats matriciels sont Portable Network Graphics (.png), Joint Photographic Experts Group (.jpeg), Tagged Image File Format (.tiff), BITMAP (.bmp), Graphics Interchange Format (.gif) pour citer les plus courants.

Définition et résolution

On appelle *définition* le nombre de points (pixel) constituant l'image, c'est-à-dire sa « dimension informatique » (le nombre de colonnes de l'image que multiplie son nombre de lignes). Une image possédant 640 pixels en largeur et 480 en hauteur aura une définition de 640 pixels par 480, notée 640x480 soit 307200 pixels.

La *résolution*, terme souvent confondu avec la *définition*, détermine en revanche le nombre de points par unité de longueur, exprimé en points par pouce (PPP, en anglais DPI pour Dots Per Inch), un pouce représentant 2.54 cm. La résolution permet ainsi d'établir le rapport entre le nombre de pixels d'une image et la taille réelle de sa représentation sur un support physique. Une résolution de 300 dpi signifie donc 300 colonnes et 300 rangées de pixels sur un pouce carré, ce qui donne donc 90000 pixels sur un pouce carré. La résolution de référence de 72 dpi nous donne un pixel de 1/72 (un pouce divisé par 72) soit 0.353 mm, correspondant à un point pica (unité typographique anglo saxonne).

Les dimensions d'une image sont donc définies par :
- largeur = nombre de colonnes / résolution,
- hauteur = nombre de lignes / résolution.

Compression

La plupart de ces formats utilisent des algorithmes de compression, afin de réduire la taille de l'image sur les mémoires de masse de l'ordinateur (disque durs, …).

On définit alors le taux de compression par : (1 - (taille du fichier image))/(taille de l'image en mémoire)

La compression peut être réalisée avec ou sans perte :
- sans perte : l'image comprimée est parfaitement identique à l'originale,
- avec perte : l'image est plus ou moins dégradée, selon le taux de compression souhaité.

21. https://www.csfieldguide.org.nz/en/interactives/pixel-viewer/
22. https://csfieldguide.org.nz/en/interactives/colour-matcher/

1.3.5 Les images vectorielles

Pour reproduire une image sur une feuille, on peut la diviser en grille et définir un niveau de gris pour chaque case, mais on peut aussi tout simplement dessiner une figure, par exemple un trait d'un millimètre d'épaisseur allant d'un point A à un point B de l'image. De la même manière, en informatique, il est possible de représenter des images sous forme de grilles de pixels, comme nous l'avons vu, mais il est en effet également possible de définir une image comme une collection d'objets graphiques élémentaires (un segment, un carré, une ellipse…) sur un espace plan : c'est le principe des images vectorielles.

L'image vectorielle est dépourvue de matrice. Elle est en fait créée à partir d'équations mathématiques. Cette image numérique est composée d'objets géométriques individuels, des *primitives géométriques* (segments de droite, arcs de cercle, polygones, etc.), définies chacunes par différents attributs (forme, position, couleur, remplissage, visibilité, etc.) et auxquels on peut appliquer différentes transformations (rotations, écrasement, mise à l'échelle, inclinaison, effet miroir, symétrie, translation, et bien d'autres …).

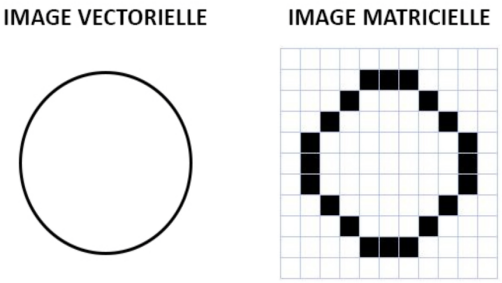


Fig. 1.12 – Un même cercle en représentation matricielle et vectorielle.

À l'inverse de l'image matricielle composée de pixels, l'image vectorielle peut être **redimensionnée** sans pour autant perdre en qualité. Elle est contenue dans un **fichier beaucoup plus léger** qu'une image pixelisée, indépendamment de sa taille et de sa définition. En revanche, chaque forme d'une image vectorielle est remplie d'une seule couleur dite solide ou d'un dégradé de couleurs. Elle reste donc **limitée en termes de réalisme**, et donc inutilisable en photographie par exemple. De plus une image vectorielle ne peut être **créée qu'à partir d'un logiciel dédié**, et n'est pas reconnue par les navigateurs internet.

Les formats vectoriels les plus courants sont Postscript (.ps) et Encapsulé Postscript (.eps), Adobe Illustrator (AI), Portable Document Format (PDF), WMF (format Windows).

Micro-activité

Saisissez le texte suivant dans un éditeur de texte et enregistrer le sous forme de fichier *.svg*. Il vous sera ensuite normalement possible d'ouvrir ce fichier avec un logiciel pour afficher les images.

```
<svg width="100" height="100"  version="1.1" xmlns="http://www.w3.org/2000/svg">
<circle cx="50" cy="50" r="40" stroke="black" stroke-width="2" fill="red"/>
</svg>
```

Modifier le fichier afin de dessiner quatre carrés différents.

Pour aller plus loin

Identifiez et listez les avantages et les inconvénients du format vectoriel en comparaison avec le système matriciel.

1.3.6 Bonus

Une œuvre d'art numérique signée Andreas Gysin …[23]

1.3.7 Exercices

Exercice 1 – Définition

Quelle est la deéfinition d'une feuille scanneée de largeur 6,5 pouces, de hauteur 9 pouces en 400 dpi ?

Exercice 2 – Carte graphique

1 - Calculer, pour chaque définition d'image et chaque couleur, la taille mémoire nécessaire à l'affichage.

Définition de l'image	Noir et blanc	256 couleurs	65000 couleurs	True color
320x200				
640x480				
800x600				
1024x768				

Exercice 3 – Compression

1. Une image de couleur a pour format : 360 × 270. Elle est enregistrée en bitmap 8 bits. Quelle est sa taille sur le disque dur (détaillez les calculs) ?
2. Une image noir et blanc de format 1024 × 1024 est enregistrée en JPG. Le taux de compression est de 50%. Quelle est sa taille sur le disque dur (détaillez les calculs) ?

23. https://play.ertdfgcvb.xyz/#/src/demos/doom_flame_full_color

Exercice 4 – Appareil photo

L'appareil numérique FinePix2400Z (Fujifilm) permet la prise de vue avec trois définitions : a) 640x480 pixels ; b) 1280x960 pixels ; c) 1600x1200 pixels.

Calculez la taille de l'image non-compressée pour chaque définition.

Exercice 5 – Pixelisation

Une image numérique de définition 1024×768 mesure 30 cm de large et 20 cm de haut.

1. Déterminez les dimensions des pixels.
2. On a une photographie de 10 cm sur 5 cm que l'on scanne avec une résolution de 300 ppi. Quelle sera alors la taille de l'image (en nombre de pixels) ?
3. Soit une image 15×9 cm, définie en RVB, que l'on scanne en 72, 300 et 1200 ppi. Quels seront les poids des images, pour une profondeur de 16 bits par couleur ?

1.4 Le son

Un son est une histoire d'énergie et de vibrations. Un son émerge quand des molécules subissent une pression initiale, ce qui va les amener à avancer et entrainer ce mouvement sur les molécules devant immédiatement voisines en leur transmettant une grande partie de cette énergie. Suite à ce nouveau mouvement, elles repartent en arrière pour retrouver leur position d'équilibre ayant transmis cette énergie initiale aux molécules voisines qui à leur tour vont se comporter de la même manière.

Vidéo 1: https://www.youtube-nocookie.com/embed/kW9nwkrfGFw

Toutes ces «tranches» de molécules vont donc osciller successivement, formant une onde qui va se déplacer au sein du milieu matériel : air, eau, caoutchouc par exemple. C'est ce que l'on peut observer lorsqu'un projectile heurte une flaque d'eau : à partir du point d'impact, se forme progressivement une onde circulaire qui s'étend et se propage à la surface de l'eau.

Vidéo 2: https://www.youtube-nocookie.com/embed/Yi3LW5riHfc

Le son est donc une **vibration mécanique**, nécessitant un **milieu matériel** : s'agissant des sons que nous entendons tous les jours, le milieu matériel est bien évidemment l'air ambiant.

On appelle **fréquence** du son, la vitesse avec laquelle ces molécules vibrent. Plus la vibration des molécules est rapide, plus le son est aigu : on parle de fréquence élevée. Inversement, plus la vibration est lente, plus basse est la fréquence. Une corde de guitare détendue vibre moins vite que sa voisine très tendue, elle va produire un son plus grave avec une oscillation bien plus lente.

Le niveau sonore correspond lui à la hauteur de l'oscillation : on parle d'**amplitude**.

Ce phénomène physique d'oscillation des molécules dans l'air est capté par notre oreille en mettant en vibration nos organes qui vont convertir cette pression reçue en signaux électriques transmis au cerveau. Votre musique préférée est donc une addition de sons avec des fréquences et amplitudes différentes qui vont vous fait vibrer au sens propre… comme au figuré !

Entre phénomène physique et organe sensoriel, le son physique (on parle également de son **analogique**) va être un ensemble d'oscillations, de vibrations, définies par des fréquences et des amplitudes.

Vidéo 3: https://www.youtube-nocookie.com/embed/XFyT1bsSnHI

Chaque «son élémentaire» peut ainsi être assimilé à une courbe comme celle décrite dans la vidéo : on parle de courbe sinusoïdale, ou encore de sinusoïde. Les sons ou la musique que vous écoutez n'est autre qu'une somme de ces courbes «convenablement» arrangées.

La question est de savoir comment ramener ces oscillations sinusoïdales combinées ensemble en un ensemble de 0 et 1 pour être stockées numériquement dans un ordinateur, comme les nombres, images et les caractères.

Le saviez-vous ?

Les casques à conduction osseuse transmettent les vibrations directement à l'os temporal du crâne : la cochlée qui est nichée dans cet os va vibrer et transmettre les informations électriques au cerveau, comme le ferait un signal passant par le tympan et le marteau.

Le saviez-vous ?

Vous rappelez-vous l'explosion de l'étoile de la mort dans Star Wars ? Eh bien un son pareil ne peut exister dans l'espace : il n'y a pas assez de molécules à agiter, l'énergie transmise par l'explosion ne peut pas se propager de la sorte.

1.4.1 Numérisation

La conversion d'une grandeur physique analogique continue – température, vitesse du vent, position d'une girouette, etc. – en données numériques digitales est appelée **numérisation**. Elle est réalisée en trois étapes : un **échantillonnage**, une **quantification** puis un **encodage**.

Le processus de numérisation engendre une quantité d'information (des bits) qui vise à représenter, aussi précisément que nécessaire, la grandeur physique sous une forme manipulable par les ordinateurs.

Il s'agit donc d'un compromis entre la qualité de la représentation et les coûts engendrés par un fichier plus grand, qui prend plus de place de stockage, plus de temps à copier, à transmettre sur un réseau et/ou nécessite une puissance de calcul plus importante pour la numérisation (conversion analogique/digitale) et pour la **reconstruction** (conversion digitale/analogique).

Ci-après, un signal continu sera numérisé, mettant en évidence le rôle et les effets des différents paramètres de la numérisation. Il s'agira pour l'exemple de l'intensité sonore telle qu'elle peut être capturée par un microphone.

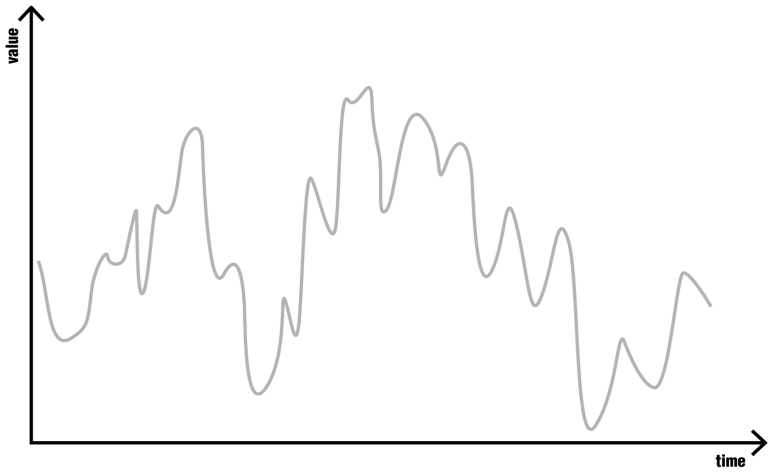


Fig. 1.13 – Signal continu à numériser, par exemple un son.

Le saviez-vous ?

Les sons, tels que perçus par notre ouïe, résultent de la vibration de l'air, prenant la forme d'oscillations cycliques de la pression. Ces oscillations peuvent être capturées par la membrane d'un microphone et générer un signal électrique qui peut être numérisé.

En échange, un signal électrique qui stimule un haut-parleur produit des oscillations de la pression de l'air qui peuvent à leur tour être perçues. Ce signal peut provenir de la reconstruction d'un signal numérisé.

1.4.2 Échantillonnage

L'intervalle temporel entre deux mesures est appelé la période d'échantillonnage. La **fréquence d'échantillonnage** (sampling rate) est le nombre de mesures prises par seconde, exprimée en Hz.

Les limites pratiques d'un échantillonnage sont fixées par la fréquence de Nyquist, qui, de façon très simplifiée, indique que l'information découlant d'un processus dont la fréquence est supérieure à la moitié de la fréquence d'échantillonnage sera perdue lors de la numérisation. Il ne sera donc jamais possible d'avoir une représentation complète d'un processus complexe, tout au mieux une représentation suffisante. Comme toute activité d'ingénierie, la solution retenue résulte d'une pesée d'intérêts et non d'une évidence pointant vers une solution unique.

Sachant que l'oreille humaine ne perçoit globalement que les fréquences comprises entre 20 et 20000 Hz, une fréquence d'échantillonnage supérieure à 40 kHz permettra de restituer l'ensemble de l'information physiologiquement perceptible par l'oreille humaine.

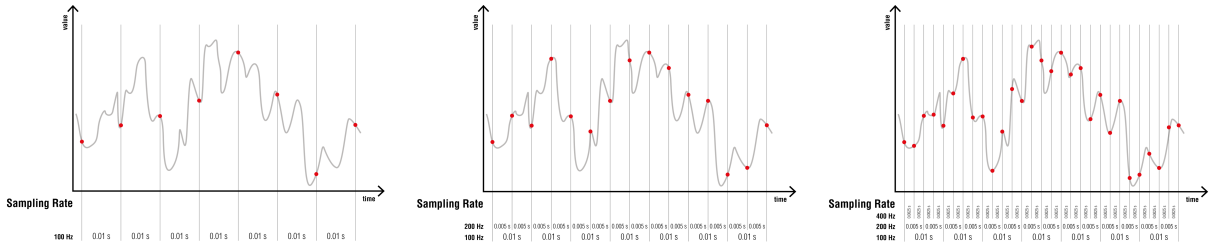


Fig. 1.14 – Effet de la fréquence d'échantillonnage (sampling rate : 100, 200 et 400 Hz) sur la représentation obtenue par numérisation. Plus la fréquence est élevée, plus la quantité d'information collectée est importante. Dans tous les cas, les détails du signal qui se déroulent entre les échantillonnages sont perdus.

Le saviez-vous ?

La fréquence d'échantillonnage de 44.1 kHz a été retenue dans les années 1970 pour permettre l'utilisation des bandes vidéo pour stocker les enregistrements numériques. Ces bandes représentaient le meilleur rapport volume de stockage/prix pour l'époque.

Cependant, les formats vidéos sont cadencés sur la fréquence du système électrique local : 60 Hz pour le NTSC américain (et 30 images par seconde) ; 50 Hz pour le PAL européen (et 25 images par seconde). En choisissant le multiple commun de 44.1 kHz, la norme permettait d'être utilisée avec les deux formats NTSC et PAL comme support de stockage physique pour le transport du "master" stéréo en vue de son impression sur des CDs.

Depuis, avec la disparition des cassettes vidéo comme supports de données, puis l'apparition du support DAT, l'échantillonnage à 48 kHz s'impose progressivement (avec ses multiples 96 et 192 kHz). Ces valeurs ont l'avantage d'être des multiples de huit, ce qui est toujours favorable dans le domaine informatique.

De plus, ces fréquences simplifient la synchronisation avec les enregistrements vidéo en 24, 25, 30, 60, 100 et même 120 images par seconde. Les multiples de 48 kHz sont donc des fréquences d'échantillonnage de choix pour la diffusion HDTV, notamment.

C'est la raison pour laquelle l'échantillonnage de la musique en qualité "CD" est réalisé à 44.1 kHz, en prenant en compte une petite marge pour l'utilisation de filtres passe-bas lors de l'enregistrement.

Une fréquence d'échantillonnage inférieure génère un son dont la qualité est dégradée, à commencer par la précision des sons les plus aigus, aboutissant à une numérisation qui rappelle le son des anciens téléphones analogiques dont les fréquences transmises étaient limitées à 3.4 kHz pour des raisons techniques.

D'ailleurs, les premiers téléphones mobiles ont par la suite répliqué ce niveau de qualité sonore comme base pour leur échantillonnage numérique (norme G.711). Selon la norme utilisée, les téléphones mobiles actuels transmettent quant à eux les fréquences jusqu'à 7 kHz (normes G.722.2 et suivantes).

Une fréquence d'échantillonnage supérieure génère une plus grande quantité d'information, sans ajouter de valeur qualitative pour la très grande majorité des auditeurs.

Le choix de la fréquence d'échantillonnage résulte donc d'une délicate balance entre coûts (taille des données) et bénéfices (qualité de la numérisation).

1.4.3 Quantification

La quantification d'une valeur échantillonnée requiert de déterminer la **précision** de chaque échantillon, ce qui détermine le volume de données générées. Cela découle de la *représentation des entiers* par les ordinateurs.

La précision de l'encodage est donnée par la **profondeur de l'échantillonnage** (bit depth) exprimée en bits (binary digits). Comme pour l'échantillonnage, plus la profondeur de l'échantillonnage est importante, plus la quantité d'information générée est importante.

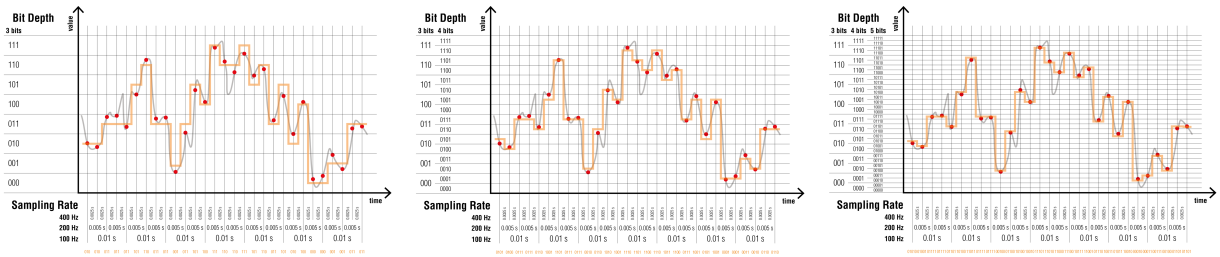


Fig. 1.15 – Effet de la profondeur de l'échantillonnage (bit depth : 3, 4 et 5 bits) sur la représentation obtenue par numérisation. Plus la profondeur est importante, plus la discrimination du signal et la différence entre les basses et les hautes intensités est importante. La quantité d'information générée (le nombre de 0 et de 1) devient également plus importante.

Lorsque l'ensemble de la plage des valeurs possibles est utilisée pour l'encodage, la profondeur de l'échantillonnage définit la **plage dynamique** disponible. Elle est définie entre la valeur encodée la plus petite (0, par exemple) et la valeur encodée la plus élevée ($2^n - 1$ pour une valeur encodée sur n bits, par exemple). Elle correspond également à une idée de précision ou de discrimination des échantillons.

Ici encore, l'oreille humaine ne peut percevoir ni les intensités les plus faibles (inférieures au bruit émis par l'individu lui-même) ni les intensités au-delà du seuil de douleur.

Une précision minimale (environ 8 bits) est ainsi nécessaire pour restituer agréablement un enregistrement respectant les subtilités de l'expression orale (entre voix posée et criée, par exemple).

Au-delà de 16 bits, une profondeur d'échantillonnage supérieure engendre une plage dynamique qui n'a pas d'application pertinente pour la restitution des sons pour la plupart des humains, au coût d'une plus grande quantité d'information collectée.

À l'inverse, il est nécessaire de gérer correctement la plage d'amplitude dans laquelle la numérisation du signal se déroule. Cela s'opère en agissant sur le paramètre de **gain** du signal.

La **distorsion** découle d'un signal dont l'amplitude dépasse les capacités d'encodage du système. Dans ces conditions, un ajustement du gain d'entrée est nécessaire pour rester au plus proche des limites du système, sans les franchir.

La numérisation d'un signal dont l'amplitude serait par trop réduite débouche au contraire sur un encodage qui contient moins d'information, ce qui limite les opérations réalisables numériquement par la suite sans détériorer la qualité du signal.

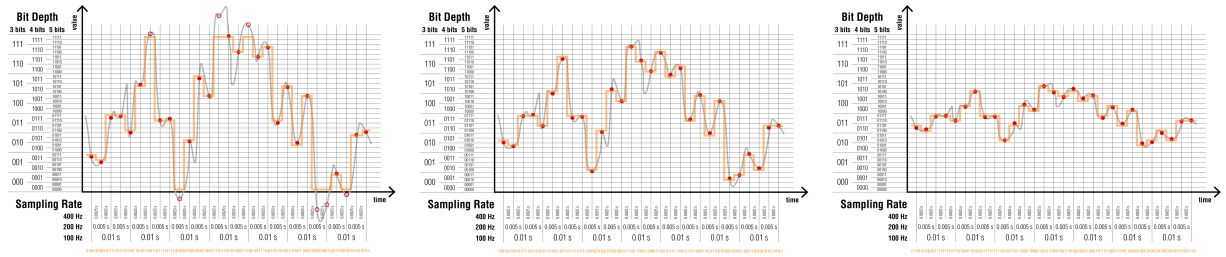


Fig. 1.16 – Effet du gain (trop haut, correct, trop bas) sur la représentation obtenue par numérisation. La distorsion résulte de valeurs très différentes de celles du signal original. Cette aberration du processus de numérisation ne peut plus être corrigée, car de l'information a été perdue au passage. À l'inverse, un gain trop faible nuit à la dynamique de l'information collectée, c'est-à-dire que l'écart entre la valeur retenue la plus faible et la plus élevée n'est qu'une fraction de l'intervalle disponible. Il en résulte une perte de précision.

On notera finalement que l'échantillonnage et la quantification travaillent ensemble pour définir la qualité du signal numérisé. Ces deux paramètres ne sont pas complètement indépendants. Leur choix est réalisé en fonction du résultat escompté et de ce que l'on cherche à réaliser avec le signal numérisé.

Pour l'intensité sonore par exemple, une fréquence d'échantillonnage insuffisante ne peut pas être compensée par une profondeur d'échantillonnage supérieure. La qualité du résultat n'est pas améliorée.

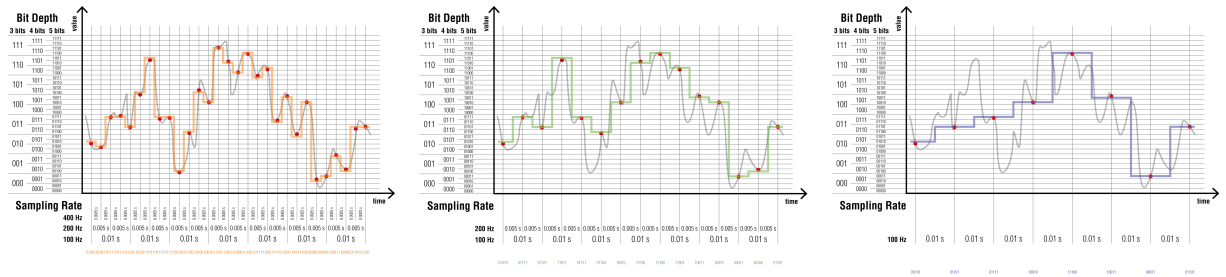


Fig. 1.17 – Effet de la fréquence d'échantillonnage (sampling rate : 400, 200 et 100 Hz) sur la représentation obtenue par numérisation à une profondeur donnée (sampling depth : 5 bits). Une importante profondeur d'échantillonnage ne compense pas une fréquence d'échantillonnage insuffisante.

Les dispositifs électroniques dont la fonction est l'échantillonnage et la quantification des signaux sont appelés des convertisseurs analogique-numérique (CAN) ou **analog to digital converter** (ADC), en anglais.

1.4.4 Encodage

L'encodage de l'information numérisée se fait dans des formats de fichiers spécifiques aux applications.

Dans l'absolu, on pourrait imaginer un format universel de stockage de 0 et de 1. En connaissant la profondeur de l'échantillonnage, il serait aisément possible de reconstruire un signal. Toutefois, cela pose plusieurs problèmes.

Cela ne donnerait aucune indication sur l'interprétation qu'il faut faire de ce signal (est-ce un son ? une image ? la variation de la vitesse du vent ?) ou même les bornes de ce signal (entre 0 et $2^n - 1$? entre $-2^{(n-1)}$ et $2^{(n-1)} - 1$?).

De plus, la quantité de mémoire nécessaire pour stocker et pour manipuler les données serait maximisée. Or, il est possible de construire des formats de fichiers qui exploitent les propriétés spécifiques au signal numérisé pour simplifier dans un deuxième temps le résultat de la numérisation avant de l'enregistrer. Cela débouche sur des fichiers qui sacrifient une partie de la qualité du signal numérisé en échange d'un gain sur la taille des fichiers générés. L'usage de la mémoire est ainsi économisé, mais, en contre-partie, un plus grand nombre de calculs est nécessaire pour manipuler les signaux.

C'est ainsi que des fichiers optimisés différents sont disponibles pour stocker des fichiers d'images (JPG), de vidéo (MP4), de son (MP3), ou de toute autre application. La plupart recourent pour cela à des compressions destructives au-cours desquelles une partie de l'information est abandonnée car, dans le contexte particulier, elles ne sont pas jugées indispensables.

Par exemple, la reproduction exacte des nuances de bleu du ciel importe peu pour un film d'action. Pourtant, ces mêmes nuances sont essentielles pour la reproduction d'un tableau de Monnet…

1.4.5 Reconstruction

On appelle **reconstruction** le processus qui transforme un signal numérisé en une variation continue d'une grandeur physique.

Les dispositifs électroniques dont la fonction est la reconstruction des signaux sont appelés des convertisseurs numérique-analogique (CNA) ou **digital to analog converter** (DAC), en anglais. La sortie du convertisseur est généralement une tension électrique proportionnelle à l'intensité du signal.

1.5 Redondance

Pourrait-on construire un véhicule qui ne tombe jamais en panne ?

Si tous les ingénieurs cherchent à y parvenir, l'aviation commerciale est un domaine dans lequel les résultats sont les plus probants. Les statistiques [24] montrent que, depuis une décennie, seuls 2 à 4 accidents mortels par millions de vols sont enregistrés.

Ce résultat est atteint au prix d'une maintenance extrêmement stricte, une formation exigeante et continue du personnel de bord, une analyse méticuleuse de chaque incident, un contrôle permanent du respect des recommandations tant par les constructeurs que par les opérateurs, et enfin un soin particulier apporté à la redondance des systèmes critiques.

La **redondance** est une technique qui consiste à dupliquer les fonctions critiques d'un système pour augmenter sa fiabilité ou ses performances. Évidemment, la redondance des systèmes a un coût : en complexité, en masse et volume, et en maintenance. En effet, paradoxalement, comme chaque composant a une probabilité de panne (fût-elle faible), plus il y a de composants, plus il y a de pannes. Toutefois, moins leurs conséquences sont graves. On parle alors d'un **système robuste ou résilient**, puisqu'il n'est pas mis en péril par la première panne.

La redondance se retrouve à tous les niveaux des systèmes informatiques, qu'ils soient embarqués dans un avion ou non. Ces redondances ont toujours un coût qui se mesure dans ce cas par une **augmentation** de la **quantité de données** et/ou du **temps de traitement** requis. Ils contribuent en échange à la sécurité des données, en augmentant l'**intégrité** (cohérence) et/ou la **disponibilité** de l'information.

1.5.1 Somme de contrôle (checksum)

Pour s'assurer qu'une information est reçue **intégralement** (sans omissions) et **parfaitement** (sans erreurs), un expéditeur pourrait naïvement l'envoyer deux fois, par deux moyens indépendants, dans un SMS et sur une carte postale par exemple. Le destinataire devrait alors renvoyer une confirmation… par deux moyens indépendants ? !

On convient aisément que cette solution serait atrocement dispendieuse. Une étude attentive montre que, de plus, elle ne permet pas de déterminer laquelle des deux copies est la bonne version lorsqu'elles diffèrent, ce qui indique qu'il y a un problème, mais ne propose pas de solution pour le résoudre.

Pourtant, aucun signal n'étant parfait, l'augmentation de la vitesse de transmission débouche nécessairement sur une augmentation des erreurs, notamment des pertes d'information et des mutations.

Les systèmes informatiques, dès lors qu'ils communiquent continuellement et abondamment, sont particulièrement sensibles à ce problème. Lorsqu'il s'agit d'assurer l'intégrité des informations transmises, des solutions plus élégantes que la proposition naïve présentée plus haut ont été élaborées pour effectuer un contrôle de qualité par redondance.

Les **checksums** par exemple sont une brève représentation d'un bloc d'information plus grand, des sortes d'empreintes numériques. Bien qu'elles soient transmises avec le bloc d'information qu'elles représentent, leur petite taille relative ne surcharge pas démesurément la transmission. En choisissant une représentation qui se calcule et se contrôle facilement, ces checksums n'imposent pas non plus un temps de traitement beaucoup plus long pour les créer et les vérifier.

24. https://www.icao.int/safety/iStars/Pages/Accident-Statistics.aspx

Idéalement, les checksums de deux blocs d'information sont très différentes même lorsque les différences entre les blocs sont infimes. Cela simplifie en effet la détection des erreurs.

La forme la plus simple est utilisée depuis la nuit des temps informatiques : le **bit de parité**. Il permet de contrôler par redondance une erreur sur la transmission d'un paquet de 7 bits, en utilisant 1 bit supplémentaire.

Dans l'exemple qui suit, on donne la valeur 0 au bit de parité lorsque la somme des bits de la valeur à transmettre est paire, et la valeur 1 lorsque cette somme est impaire. On notera que, de cette façon, la somme des 8 bits est toujours paire.

Le bit de parité est habituellement placé à la position de poids le plus faible, ce qui permet de contrôler directement la valeur transmise.

Valeur à transmettre	Somme des bits	Bit de parité	Valeur transmise
0000000	0	0	00000000
0000001	1	1	00000001
0000010	1	1	00000101
0000011	2	0	00000110
0000100	1	1	00001001
0000101	2	0	00001010
0000110	2	0	00001100
0000111	3	1	00001111
…		…	…
1111101	6	0	11111010
1111110	6	0	11111100
1111111	7	1	11111111

On notera que, pour un coût de taille modeste (un huitième des bits transmis) et un calcul rapide à réaliser (une somme et une comparaison), des erreurs de transmission ponctuelles — celles qui ne portent que sur un nombre de positions impair — sont immédiatement détectables. Cela inclut les erreurs qui porteraient sur le bit de parité lui-même.

Le saviez-vous ?

Les contrôles de bit de parité peuvent être intégrés aux composants électroniques.

La mémoire vive RAM (*Random Access Memory*) de type ECC (*Error-Correcting Code*) s'appuie sur des bits de contrôle pour détecter, voire corriger, les erreurs de stockage qui pourraient affecter les données ou le code des logiciels en cours d'exécution.

Cette fonction supplémentaire explique leur coût plus élevé.

1.5.2 Fonction de hachage

L'exemple de bit de parité permet grossièrement de contrôler une communication caractère par caractère. Une forme plus élaborée du même concept prend la forme du **hachage** de l'information qui peut alors s'appliquer à des quantités d'information beaucoup plus importantes, de type et de longueurs variables.

Prenons par exemple un texte représenté par les valeurs décimales de l'encodage ASCII et construisons une empreinte digitale sur la base de calculs simples :

h	a	c	h	a	g	e
104	97	99	104	97	103	101

La somme de toutes ces valeurs totalise 705 (= 104 + 97 + 99 + 104 + 97 + 103 + 101), soit 0x2**C1** en hexadécimal.

La somme des produits de chaque valeur par l'index de sa position totalise quant à elle 2821 (= 1x104 + 2x97 + 3x99 + 4x104 + 5x97 + 6x103 + 7x101), soit 0xB**05** en hexadécimal.

On peut décider de limiter ces deux totaux à leur deux dernières positions hexadécimales, (C1 et 05, respectivement), ce qui permet de construire une empreinte digitale (**digest** ou **hash**) C105 indépendante de la longueur du texte.

Si le texte venait à être modifié, ne serait-ce que très légèrement, l'empreinte numérique ainsi définie serait afectée :

h	a	c	h	a	h	e
104	97	99	104	97	**104**	101

En effet, la somme des valeurs totalise alors 706 (= 104 + 97 + 99 + 104 + 97 + 104 + 101), soit 0x2**C2** en hexadécimal, alors que la somme des produits totalise 2827 (= 1x104 + 2x97 + 3x99 + 4x104 + 5x97 + 6x104 + 7x101) soit 0xB**0B**, ce qui donne un hash de C20B au lieu de C105 précédemment, alors qu'un seul bit diffère entre les deux messages.

On notera que les mots 'hat' et 'fer' débouchent sur la même empreinte (3D86), par exemple.

On notera que la suppression d'une lettre au texte ("hachage" => C105) ne change pas la longueur de cette simple empreinte mais sa valeur ("hachag" => 5C42) et que cette **fonction de hachage** est aussi sensible à la casse ("Hachage" => A1E5).

Le saviez-vous ?

Même si le hachage d'une information est à dessein relativement rapide en soi, des contraintes artificielles provoquant délibéremment la multiplication de ces hachages peuvent être imposées lors de l'ajout des blocs dans une **blockchain**. Cela constitue la preuve de travail (*proof-of-Work*, PoW) des cryptomonnaies que l'on nomme communément **minage des cryptomonnaies**.

La quantité faramineuse de calculs ainsi engendrée pour complexifier artificiellement ce hachage est responsable d'une part mesurable de la consommation électrique mondiale.

Le partage de l'intégralité de la blockchain par l'ensemble des membres du réseau constitue en soi une redondance qui favorise la disponibilité de l'information, mais au prix du stockage d'une quantité mirobolante de copies et de la complexité de leur maintenance.

On notera qu'une empreinte numérique est une simplification de l'information hachée. Il est dès lors envisageable de trouver deux informations, de longueurs possiblement différentes, dont les empreintes sont identiques. En contrepartie, il n'est en principe pas envisageable de reconstruire le texte d'origine sur la base de la seule empreinte.

Toutefois, grâce à leurs propriétés (déterministes et rapides), des **fonctions de hachage** plus complexes (**SHA**, **MD5**…) trouvent des applications dans de nombreux contextes : authenticité (signatures numériques), intégrité (erreurs de transfert, stockage, blockchains…), identification (fichiers, connexions réseau…), authentification (stockage et vérification des mots de passe)…

1.5.3 Disques RAID

Les pannes de disques durs sont très communes et entrainent des pertes de données aux conséquences parfois irrécupérables.

La mise en place de sauvegardes automatiques régulières sur des supports distincts et, de préférence, délocalisés (en soi une forme de redondance sur le stockage de l'information) représente un début de réponse. Toutefois, si le support utilisé pour le stockage n'est lui-même pas résilient, la sécurité de ces sauvegardes n'est pas assurée.

Une solution technique a été proposée dès les années 1980 basée sur la disponibilité de grappes de disques durs relativement bon marché (*Redundant Array of Independent Disks*, RAID). Il est alors possible de créer (entre autres) des disques logiques de taille T (RAID 1), formés d'une grappe de n disques physiques de taille T, sur chacun desquels est stockée une copie complète des données. À chaque écriture, le système maintient automatiquement l'ensemble des n copies, ce qui permet de récupérer l'intégralité de l'information, même si $n - 1$ disques sont endommagés.

Ici encore, en exploitant le principe des bits de parité décrits précédemment, il est par exemple possible de construire une grappe de 3 disques (RAID 5) de taille T capable de stocker 2xT données. La part de stockage perdue (un disque sur trois) y est utilisée de telle sorte que, lorsqu'un des trois disques est perdu – n'importe lequel des trois ! –, aucune information n'est réellement perdue. Mieux, si le disque défectueux est remplacé, son contenu peut être reconstruit automatiquement et la résilience de la grappe rétablie. En outre, les vitesses d'écriture et de lecture sur ces trois disques en grappe est également accélérée.

Ce type d'infrastructure, malgré son coût plus élevé, est à la base des systèmes critiques qui ne peuvent se permettre de perdre des informations, ce qui pourrait inclure, à terme, les copies de sécurité des postes personnels, quand les données traitées sont sensibles.

1.5.4 Cloud computing

Les systèmes informatiques récents sont distribués à l'échelle d'Internet, tant pour leurs parties matérielles que logicielles. On parle de systèmes *cloud* ou **informatique en nuage**.

On trouve ainsi des systèmes de stockage de fichiers distribués sur plusieurs ordinateurs, voire dans plusieurs fermes de stockage. Cette configuration augmente considérablement la sécurité des données en contribuant à leur **intégrité** et à leur **disponibilité**.

1.6 Conclusion

Trois éléments clés sous entendent notre description du monde : la matière, l'énergie et l'information. Transmettre l'informationest sans aucun doute la révolution de ce 20ème siècle achevé et l'enjeu majeur de ce début du 21ème.

La notion d'information est porteuse de sens. Elle s'appuie sur des *données* et un *canal de transmission* ; elle est véhiculée entre un *émetteur* et un *récepteur*.

Dès lors se pose la problématique de la *fidélité* de la représentation, et de l'adaptation au canal de transmission.

Fig. 1.18 – Publicité «iconique» pour la société française Pathé Marconi, du nom d'Émile Pathé (1860-1937), leader du disque phonographique dès le xixe siècle, et de Guglielmo Marconi (1874-1937), pionnier de la radio et prix Nobel de physique en 1909

La mission de la phase d'encodage est d'assurer au mieux cela. Le codage binaire d'entiers ou de caractères s'effectue sans perte, étant effectué entre deux espaces *discrets* ou discontinus : en effet, l'espace des entiers ou des caractères se parcourt par sauts successifs d'une valeur à une autre et trouve une correspondance parfaite avec le codage binaire. Chaque entier distinct va pouvoir trouver sa représentation parfaite comme nous l'avons vu au premier chapitre via son codage binaire et sa représentation décimale associée. De même, chaque caractère trouve son équivalent binaire via la table de représentation ASCCI ou mieux UTF présentées au chapitre deux. Représenter un grand nombre entier, ou une palette plus large de caractères ne dépend dès lors que de la capacité de la machine et du nombre de bits de codage : 4, 8, 16, 32, 64 bits…

Le problème est tout autre dès lors qu'on s'intéresse à la représentation machine des images ou du son, abordée aux chapitres trois et quatre.

Le dialogue entre *discret* et *continu* est caractéristique de la science notamment depuis le début du XXème siècle, en mathématiques, physique, chimie, biologie et… en informatique.

Fig. 1.19 – Günther Uecker, Spirale I 1997page 48, 25

Ce que l'œil perçoit est en réalité une information physique *continue*, tout comme un son. Dès lors, la représentation d'une image dans l'espace *fini* de la machine - l'ordinateur - ne pourra être que partielle ; la puissance de l'ordinateur va définir la précision avec laquelle l'information va pouvoir être transmise, sa *fidélité*.

La chaine complète émetteur- canal de transmission- récepteur participe de cette fidélité, et tout se joue alors dans l'ajustement entre cette chaine et la puissance de l'encodage, c'est à dire la capacité de la machine : la complexité de la représentation matricielle pour l'image (pixellisation), la finesse de discrétisation pour le son (échantillonnage).

Représenter l'information à travers ce que comprend la machine (c'est à dire le mode binaire) assure de pouvoir étendre ses possibilités de transmission : aujourd'hui nous pouvons écrire un texte avec un logiciel de traitement de texte, dessiner une image avec un logiciel ou la capturer avec un appareil photo numérique, enregistrer un son, puis transmettre ces «objets numériques» à une multitude de personnes, à l'autre bout du monde, sans altération significative de l'objet initial. Cette information peut être stockée, modifiée, complétée et s'intégrer à nouveau dans ce flux mondial : c'est toute la puissance et la révolution apportée par la technologie numérique, partagée aujourd'hui par plus de 80% des pays de la planète [26], et 60% des humains [27].

Dès lors, on peut s'interroger sur les évolutions de cette technologie dans le sens d'un accroissement de ces performances de représentation, c'est à dire principalement la fidélité, la rapidité et l'accessibilité. L'accroissement des performances des machines liées à l'adoption des transistors, l'arrivée des

25. https://www.echosciences-grenoble.fr/articles/l-artiste-gunther-uecker-avec-ses-tableaux-de-clous-rencontre-le-discret-et-le-continu-de-la-s
26. http://www.smartaddict.fr/ces-regions-sans-internet/
27. https://www.suricats-consulting.com/fresque-du-numerique/?cn-reloaded=1

microprocesseurs puis à la miniaturisation suit une croissance exponentielle depuis sa prédiction par Gordon E. Moore en 1965 : c'est la fameuse loi de Moore [28]. Les ordinateurs sont devenus de plus en plus petits, de moins en moins coûteux et de plus en plus rapides et puissants.

Les performances calculatoires ont impacté les mathématiques elles-mêmes, science des nombres.

Les outils de traduction parviennent à des niveaux inimaginables il y a encore vingt ans.

La synthèse vocale atteint aujourd'hui une telle qualité qu'il devient difficile pour l'oreille humaine de distinguer une voix humaine de celle d'un robot.

Pour représenter le réel, si l'on peut imaginer des marges de progression encore possibles en terme de fidélité, c'est sans doute sur l'accessibilité, la diffusion, la consommation des ressources et l'impact sur le climat que se situent les enjeux du développement numérique.

28. https://fr.wikipedia.org/wiki/Loi_de_Moore