Représenter l'information

ISN - Chapitre 5

Table des matières

I Représenter l'information

Comment représenter numériquement des informations aussi variées que des nombres, des sons, des images, des vidéos...

La mémoire des ordinateurs est constituée d'une multitude de petits circuits électroniques. Chacun ne peut être que dans deux états. Pourquoi ?

En électronique :
- 0 : pas de courant
- 1 : courant

En mathématiques (Algèbre de Boole) :
- 0 : Faux
- 1 : Vrai

Définition

Une telle valeur, 0 ou 1, s'appelle un **booléen**, un **chiffre binaire** ou encore un **bit** (pour **bi**nary digi**t**).
Une suite finie de 0 et de 1 s'appelle un **mot**.
Sa longueur est le nombre de bits qui le compose.
Un mot de longueur 8 est appelé un **octet**.

Remarque importante :

8 bits	=	1 octet	=	1 byte (en anglais)
8 b	=	1 o	=	1 B

Attention aux confusions :

1 Kilo octet	=	1 Ko	=	2^{10} o	=	1024 octets
1 Mega octet	=	1 Mo	=	2^{20} o	=	1024 Kilo octets

Mémoriser, transmettre et transformer des nombres, des textes, des images ou des sons demande d'abord de les représenter comme des suites de 0 et de 1. C'est ce que nous allons voir dans ce chapitre.

Un petit exercice pour démarrer :
On veut représenter chacune des sept couleurs de l'arc en ciel par un mot, les sept mots devant être de même longueur. Quelle est la longueur minimale de ces mots ?

II Représenter des entiers naturels

Attention : pour débuter, on cherche ici à ne représenter que les entiers **naturels** ! La prochaine partie abordera le cas général (entier **relatif**) qui est celui qui est exploité par les ordinateurs etc.

II 1 Structure du nombre

Depuis le Moyen Âge, la plupart des humains écrivent les nombres entiers naturels en **notation décimale à position**. Seuls dix chiffres sont nécessaires : 0 ; 1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9.
Par exemple, l'écriture 2359 exprime un entier naturel formé de 9 unités, 5 dizaines, 3 centaines et 2 milliers :

$$2359 = 2 \times 10^3 + 3 \times 10^2 + 5 \times 10^1 + 9 \times 10^0$$

La notation décimale à position s'appelle aussi la **notation à position en base dix**.
« à position » car la position d'un chiffre est déterminante. En effet, dans 234 et 253, le chiffre 3 n'a pas la même signification.

Comme la mémoire des ordinateurs est constituées de circuits, qui, chacun, ne peuvent être que dans deux états, il a fallut se placer dans une autre base, la base deux :
- L'un des états est le chiffre binaire 0.
- L'autre est le chiffre binaire 1.

II 2 La base deux

Les nombres exprimés en base deux sont plus difficiles à lire, mais le principe de la numération en base deux est identique à celui de la numération en base dix.

Exemple :

$$11 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Donc la représentation en base deux du nombre 11 est :

$$11 = (1011)_2$$

Remarque :
Selon les sources, on peut noter un nombre en base 2 soit sans indication supplémentaire (1011) si le contexte est clair, soit aussi $1011_{(2)}$ ou encore $\overline{1011}^2$.

Exercice

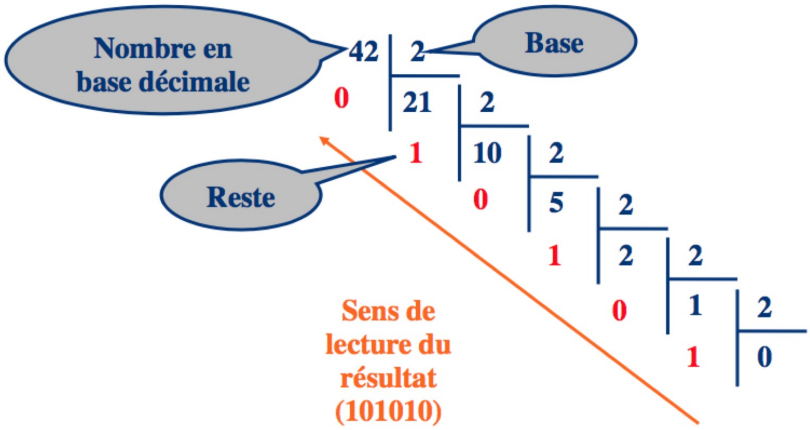
Que valent les nombres suivants (codés en binaire) en décimal ?
- $(00101001)_2$
- $(01110100)_2$

Et dans l'autre sens ?

Méthode pour trouver la représentation en base 2 d'un entier naturel donné en base 10

Pour écrire un nombre en base 2, on dispose de deux chiffres : 0 et 1.
On fait une succession de divisions euclidiennes par 2, jusqu'à obtenir un quotient égal à 0.
Les restes successifs sont les chiffres du nombre en base deux, le premier obtenu étant l'unité.

Exemple :
Déterminer la représentation en base deux du nombre entier 42 :



Exercice

Donner la représentation en base 2 des nombres entiers suivants :
- 1980
- 666

Questions pour finir :

1. Donner la liste des entiers naturels pouvant être codés avec un octet.
2. Généralisation : Combien d'entiers naturels peut-on codé sur un mot de n bits ?

II 3 D'autres bases

On peut généraliser les méthodes vues pour n'importe quelle base b. En informatique, on utilise également fréquemment la base hexadécimale (base 16).

En base 16, on a besoin de 16 chiffres : 0 ; 1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9 ; A ; B ; C ; D ; E ; F.

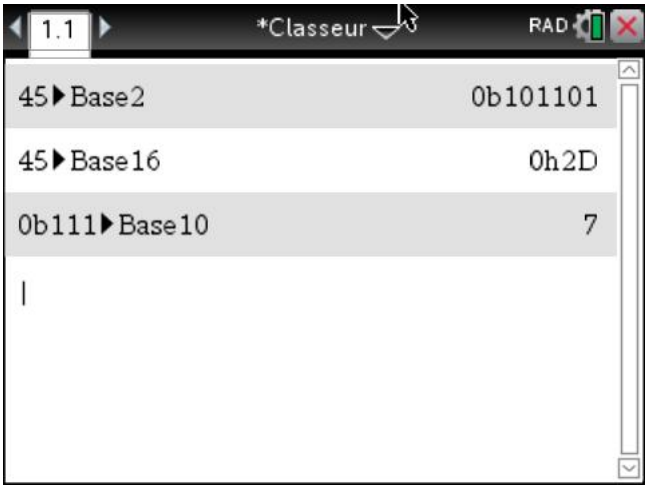
Exercices :

1. Donner la représentation en base 16 des nombres entiers suivants :
- 1980
- 666

2. Donner la représentation en base 10 des nombres suivants :
- $(4E2C)_{16}$
- $(ABCD)_{16}$
- $(281EF)_{16}$

II 4 Sur la calculatrice TI Nspire

La calculatrice peut être utilisée dans les trois bases classiques avec les **réglages du classeur**.
Vérifier les conversions suivantes :



Exercice pour finir :
Que fait le programme suivant ?

```
Define base(n,b)=
    Prgm
        Local a,r,i,p

        a:=n
        i:=0
        p:=0

        While a>0
            r:=mod(a,b)
            a:=((a-r)/(b))
            p:=p+r*10^(i)
            i:=i+1
        EndWhile

        Disp p
    EndPrgm
```

III Représenter des entiers relatifs

III 1 Les limites du système précédent

Une solution est de réserver un bit pour le signe de l'entier à représenter et d'utiliser les autres pour représenter sa valeur absolue.
Ainsi, avec des mots de 16 bits, en utilisant 1 bit pour le signe et 15 bits pour la valeur absolue, combien d'entiers relatifs peut-on représenter ?

Par exemple, $\boxed{1001 \text{ représente } -1}$ et $\boxed{0001 \text{ représente } 1}$.

Mais cette méthode a plusieurs inconvénients, l'un d'eux étant qu'il y a deux zéros, l'un positif, et l'autre négatif. L'autre problème est lié à l'addition qui ne fonctionne plus nécessairement : $0101 + 0100 = ?$

III 2 Le codage à l'aide du complément à deux

La méthode retenue consiste à utiliser un codage que l'on appelle **complément à deux**. Cette représentation permet d'effectuer les opérations arithmétiques usuelles naturellement :

▶ Un entier relatif positif ou nul sera représenté en binaire comme un entier naturel, à la seule différence que le bit de poids fort (le bit situé à l'extrême gauche) représente le signe. Il faut donc s'assurer pour un entier positif ou nul qu'il est à zéro (0 correspond à un signe positif, 1 à un signe négatif). Ainsi, si on code un entier naturel sur 4 bits, le nombre le plus grand sera 0111 (c'est-à-dire 7 en base décimale).
- Sur 8 bits (1 octet), l'intervalle de codage est donc $[-128\,;127]$.
- Sur 16 bits (2 octets), l'intervalle de codage est $[-32768\,;32767]$.
- Sur 32 bits (4 octets), l'intervalle de codage est $[-2147483648\,;2147483647]$.

D'une manière générale, le plus grand entier positif codé sur n bits est $2^{n-1}-1$.

▶ Un entier relatif négatif sera représenté grâce au codage en complément à deux :

Principe :

1. Ecrire la valeur absolue du nombre en base 2. Le bit de poids fort doit être égal à 0.
2. Inverser les bits : les 0 deviennent des 1 et les 1 deviennent des 0. On fait ce qu'on appelle le **complément à un**.
3. On ajoute 1 au résultat (les dépassements sont ignorés).

Cette opération correspond au calcul de $2^n - |x|$, où n est la longueur de la représentation et $|x|$ la valeur absolue du nombre à coder.
Ainsi, -1 s'écrit comme $256 - 1 = 255 = (11111111)_2$.

Exemples :

On désire coder la valeur -19 sur 8 bits. Il suffit :

1. D'écrire 19 en binaire : $(00010011)_2$.
2. D'écrire son complément à 1 : $(11101100)_2$.
3. Et d'ajouter 1 : $(11101101)_2$.

La représentation binaire de -19 sur 8 bits est donc $(11101101)_2$.

Remarque :
En additionnant un nombre et son complément à deux, on obtient 0. En effet, $(00010011)_2 + (11101101)_2 = (00000000)_2$ (avec une retenue de 1 qui est éliminée).

Astuce

Pour transformer de tête un nombre binaire en son complément à deux, on parcourt le nombre de droite à gauche en laissant inchangés les bits jusqu'au premier 1 (compris), puis on inverse tous les bits suivants.
Prenons comme exemple le nombre 20 : $(00010100)_2$.

1. On garde la partie à droite telle quelle : $(00010100)_2$.
2. On inverse la partie de gauche après le premier 1 : $(11101100)_2$.
3. Et voici -20 : $(11101100)_2$.

Exercice

1. Coder les entiers relatifs suivants sur 8 bits (ou 16 bits si nécessaire) : 456 ; −1 ; −56 ; −5642.
2. Que valent en base 10 les trois entiers relatifs suivants codés en binaire (complément à 2) : $(01101100)_2$; $11101101)_2$; $(1010101010101010)_2$.

III 3 Algorithme et programmation

Exercices

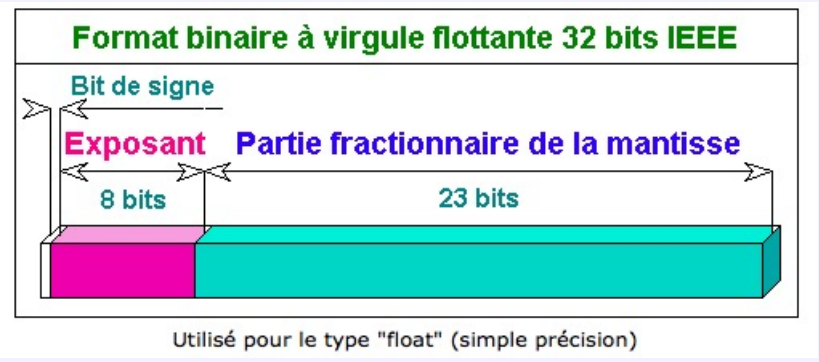
1. Écrire un programme en TI Basic qui prend en entrée deux mots de 8 bits en binaire et qui renvoie la somme de ces deux nombres.
   On pourra prendre comme structure de donnée les listes.
2. Expliquer comment faire une **soustraction** de deux nombres binaires sur huit bits. Calculer ainsi 15 − 7.

IV Représenter des nombres décimaux : la norme IEEE 754

(IEEE, à prononcer à l'anglaise, signifie : Institute of Electrical and Electronics Engineers.)

Fonctionnement :

La norme IEEE 754 définit la façon de coder un nombre décimal. Cette norme se propose de coder le nombre sur 32 bits et définit trois composantes :
▶ Le signe est représenté par un seul bit, le bit de poids fort (+ est représenté par 0 et − par 1).
▶ L'exposant est codé sur les 8 bits consécutifs au bit de poids fort. Il faut rajouter 127, soit $(01111111)_2$ à l'exposant pour une conversion de décimal vers un nombre réel binaire. Les exposants peuvent ainsi aller de −254 à 255.
▶ La mantisse (les bits après la virgule) sur les 23 bits restants. Il s'agit d'un binaire à virgule compris entre 1 inclus et 2 exclu)



A propos de la mantisse : la mantisse est comprise entre 1 et 2, elle a toujours un seul chiffre avant la virgule et ce chiffre est toujours un 1. Il est donc inutile de le représenter et on utilise les 23 bits pour représenter les 23 chiffres après la virgule.

Exemple :
Déterminer le nombre à virgule représenté par le mot en 32 bits :

$$11001001010101100010000001100001$$

▶ Le signe est représenté par 1. Il est donc **négatif**.

▶ L'exposant est représenté par 10010010. $(10010010)_2$ correspond à $2^1 + 2^4 + 2^7 = 146$.
L'exposant du nombre est donc $n = 146 - 127 = 19$.

▶ La mantisse est représentée par 10101100010000001100001.
Cela correspond à $1,10101100010000001100001$, soit $1 + (1/2)^1 + (1/2)^3 + (1/2)^5 + (1/2)^6 + (1/2)^{10} + (1/2)^{17} + (1/2)^{18} + (1/2)^{23} = 14032993/8388608$

Le nombre représenté est donc $-14032993/8388608 \times 2^{19} = -877062,0625$

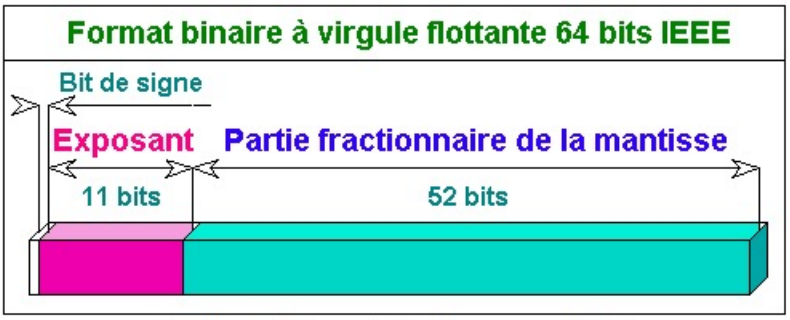
Remarques :
Certaines conditions sont toutefois à respecter pour les exposants :
▶ l'exposant 00000000 est interdit.
▶ l'exposant 11111111 est interdit. On s'en sert toutefois pour trois situations exceptionnelles :
$+\infty$ 0 11111111 00000000000000000000000
$-\infty$ 1 11111111 00000000000000000000000
NaN 1 11111111 11111111111111111111111 : permet de signaler une erreur. (NaN : « *Not A Number* »)

La formule d'expression des nombres décimaux est ainsi la suivante :

$$\boxed{(-1)^S \times 2^{E-127} \times (1+F)}$$

avec :
S le bit de signe.
E l'exposant auquel on doit bien ajouter 127 pour obtenir son équivalent codé,
F la partie fractionnaire.

Remarque :
On peut également utiliser 64 bits :
▶ Toujours un bit pour le signe.
▶ 11 bits pour l'exposant entier relatif compris entre -1022 et 1023, codé $n + 1023$.
▶ 52 bits pour la mantisse.



Utilisé pour le type "double" (double précision)

Exercice :
Déterminer le nombre à virgule représenté par le mot en 64 bits :
1100010001101001001111000011100000000000000000000000000000000000

Une petite anecdote :
Le 4 juin 1996, une 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. A 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 32768 (le plus grand entier que l'on peut coder sur 16 bits) et la conversion a été incorrecte.

V Représenter des caractères et des textes

V 1 La représentation des caractères

Un texte est une suite de caractères : des lettres minuscules et majuscules, des chiffres, des signes de ponctuations, des symboles mathématiques, *etc*
On associe à chacun un nombre, c'est-à-dire un mot binaire pour constituer un codage numérique.

Le code ASCII (American Standard Code for Information Interchange) permet de représenter 128 (2^7) caractères, le premier bit du code est toujours un zéro.

Quel type de mot peut représenter un code ASCII ?

Le code ASCII était à l'origine conçu pour représenter des textes écrits en anglais.
Il n'est pas adapté pour représenter des textes écrits dans d'autres langues. Des extensions sont alors proposées :
- Des accents, des cédilles et autres signes diacritiques : le code **latin-1** (191 caractères).
- Pour les langues d'Europe de l'Est, le code **latin-2**.
- En grec, russe, chinois, japonais, coréen, etc, un format universel : **Unicode**

Unicode recence près de 110 000 caractères et associe un nom et un numéro à chacun. A priori, ce numéro se code sur 32 bits. Mais Unicode existe en plusieurs déclinaisons, parmi lesquelles **UTF-3** et **UTF-8**. Ce dernier a vocation à devenir le standard.

V 2 La représentation des textes simples

Un texte étant une suite de caractères, on peut le représenter en écrivant les caractères les uns derrière les autres.

► Chercher sur le web la table de correspondance ASCII.
► Représenter en binaire ASCII la phrase « je pense donc je suis »

texte	Je pense, donc je suis.
codes ASCII	74, 101, 32, 112, 101, 110, 115, 101, 44, 32, 100, 111, 110, 99, 32, 106, 101, 32, 115, 117, 105, 115, 46
binaire 8 bits	01001010011001010010000001110000011001010110111001110011011001010010110000100000011001000110111101101110011000110010000001101010011001010010000000111001101101010110100101110011001011 10

► Trouver (programmer ? Calculatrice ?) la représentation en binaire ASCII du texte « cet exercice est un peu fastidieux »

Inversement, on peut décoder un texte représenté en binaire ASCII.

▶ On découpe la suite en octets.
▶ On traduit chaque octet en décimal.
▶ On cherche dans la table ASCII le caractère exprimé par chacun de ces nombres.

▶ ▶ Quel est le texte représenté en binaire ASCII par la suite de bits :
01000011001001110110010101110011011101000010000001100110011000010110001101101001011011000
1100101

▶ ▶ Quel est le texte représenté en hexa ASCII par :
4A2741492054524F5556452021

V 3 La représentation des textes enrichis

Les textes en ASCII ou en Unicode sont simplement des suites de caractères. Les éditeurs de texte sont les logiciels qui manipulent ces suites de caractères.
Les seules caractéristiques que l'on peut exprimer en ASCII par exemple sont la *casse* d'une lettre et le découpage en paragraphes, grâce au symbôle *retour chariot*.

Les traitements de texte sont les logiciels qui permettent des mises en page plus élaborées : formats visuels de caractères (appelées *polices de caractères*), tailles des caractères (11 points, 12 points etc), leur forme (romain, italique...), leur graisse (maigre, gras), la mise en valeur des titres des chapitres etc...

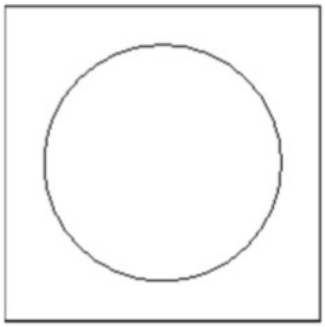
Ceci a amené à enrichir ces formats de manière à qualifier et à structurer.
Ces informations portent sur le texte et ne sont pas le texte lui-même. On les appelle des **meta-données**.
L'un de ces formats enrichis, qui est utilisé en particulier pour écrire des pages web, est appelé le **format HTML** et fera l'objet d'un chapitre plus tard dans l'année.

VI Représenter des images

VI 1 La représentation des images

Comment sont représentées des images ? Il y a deux possibilités généralement :
– Représentation symbolique (celle que nous allons voir)
– Représentation vectorielle

Voici un exemple :

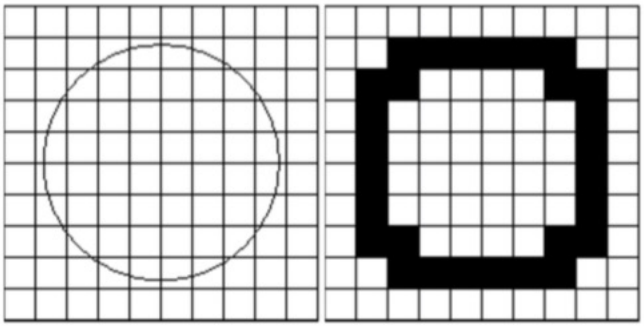


Cette image est formée d'un cercle, défini par son centre et son rayon. A partir de cette description, on pourrait reconstituer le dessin. On peut donc représenter cette image par trois nombres : deux pour les coordonnées du centre et un pour son rayon. Mais comment faire pour une image formée de plusieurs cercles ?

(quel centre, quel rayon) Un dessin formé de cercles et de rectangles ?

Cette méthode trouve rapidement ses limites, par exemple pour représenter cette image :

Pour cela, on va utiliser une représentation en **bitmap**.
On va superposer un quadrillage à l'image :



Chacune des cases de ce quadrillage s'appelle un **pixel (picture element)**.
Le dessin précédent, avec une grille de 10 x 10, se décrit alors par la suite 100 bits :

```
0000000000001111110001100001100100000010010000001
0100000010010000001001100001100011111100000000000
```

C'est une méthode approximative, mais universelle : n'importe quelle image en noir et blanc peut se décrire ainsi.

VI 2 La notion de format

Un texte, une image, un son etc, sont stockées dans un *fichier* auquel on a attribué un nom.
si on trouve un fichier qui contient la suite de bits :

```
0000000000001111110001100001100100000010010000001
0100000010010000001001100001100011111100000000000
```

Il n'est pas *a priori* évident que cette suite exprime une image de 10 pixels sur 10 pixels.
Le nom du fichier portera donc une **extension** indiquant le type de son contenu.

Le format PBM (portable bitmap)

Ce format est l'un des plus simples pour exprimer des images en noir et blanc.
Un fichier au format PBM est un fichier écrit en ASCII :
- Les caractères P1, suivis d'un retour à la ligne ou d'un espace,
- La largeur de l'image, suivie d'un retour à la ligne ou d'un espace,
- La hauteur de l'image, suivie d'un retour à la ligne ou d'un espace,
- La liste des couleurs des pixels, ligne par ligne, de haut en bas, et de gauche à droite : 0 pour blanc, 1 pour noir.

En outre, aucune ligne ne doit dépasser 70 caractères et toutes les lignes commençant par le caractère # sont ignorées.

D'autres exemples sont PGM, PPM, PNG, JPEG, GIF, PS, PICT, TIFF...

Identifier quelques formats d'images

Les différents formats qui permettent d'exprimer des images se distinguent les uns des autres par :
- Le type d'image : noir et blanc, en niveaux de gris, en couleurs, etc.
- La manière d'exprimer ces images : sous forme vectorielle ou de bitmap,
- Le fait que ces images soient compressées ou non,
- Le fait que le format soit public ou secret.

Exemple : Le format PBM est un format noir et blanc, bitmap, non compressé, public et libre.

Exercice

Chercher sur le Web les caractéristiques des formats GIF, PNG et JPEG.

VI 3 La représentation des images en niveaux de gris

Le format PGM (Portable Greymap)

C'est un fichier en ASCII qui se compose comme suit :
- Les caractères P2, suivis d'un retour à la ligne ou d'un espace,
- La largeur de l'image, suivie d'un retour à la ligne ou d'un espace,
- La hauteur de l'image, suivie d'un retour à la ligne ou d'un espace,
- La valeur maximale (par exemple 255) utilisée pour exprimer les niveaux de gris, suivie d'un retour à la ligne ou d'un espace,
- La liste des couleurs des pixels ligne par ligne de haut en bas et de gauche à droite, séparés par des retours à la ligne ou des espaces.

Comme en PBM, aucune ligne ne doit dépasser 70 caractères et toutes les lignes commençant par le caractère # sont ignorées.

VI 4 La représentation des images en couleurs

Pour comprendre comment représenter les images en couleurs, il faut d'abord s'intéresser à la manière dont notre oeil perçoit les couleurs. Ainsi, sur l'écran d'un ordinateur, chaque pixel est composé de trois sources de lumière : rouge, verte et bleue.
En faisant varier l'intensité de chacune de ces sources, on peut simuler n'importe quelle couleur :

Le format PPM (Portable pixmap)

C'est un fichier en ASCII qui se compose comme suit :
- Les caractères P3, suivis d'un retour à la ligne ou d'un espace,
- La largeur de l'image, suivie d'un retour à la ligne ou d'un espace,
- La hauteur de l'image, suivie d'un retour à la ligne ou d'un espace,
- La valeur maximale (par exemple 255) utilisée pour exprimer l'intensité des couleurs, suivie d'un retour à la ligne ou d'un espace,
- La liste des valeurs des couleurs, trois par pixel, dans l'ordre Rouge Vert Bleu, ligne par ligne de haut en bas et de gauche à droite, séparés par des retours à la ligne ou des espaces.

Exemple : un fichier PPM qui représente un carré orange de 100 pixels sur 100 pixels

```
P3
# Mon premier fichier PPM : orange
100 100
255
237 127 16
237 127 16
(...)
237 127 16 (dix mille fois en tout)
```

Il existe des formats d'images plus complexes que le format PPM, notamment pour éviter de recopier dix mille fois le même triplet de nombres. C'est ce qui s'appelle **compresser** un fichier.

Exercices

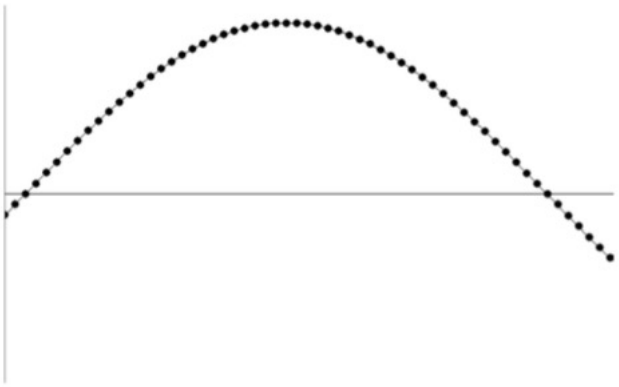
– Lequel des formats PBM, PGM et PPM est adapté pour représenter un carré noir de 10 pixels sur 10 pixels ? Même question pour un carré rouge de même taille.
Comparer les tailles des fichiers obtenus.

– Ecrire les fichiers trouvés dans un éditeur de texte et les ouvrir avec un logiciel de traitement d'images, par exemple Gimp.

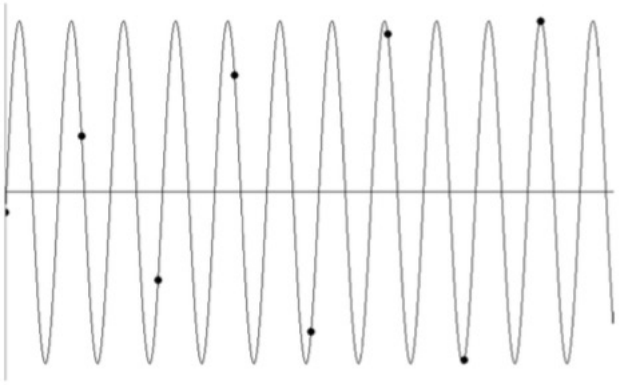
– Rechercher des informations sur la structure des fichiers de type GIF :
Combien de bits occupe la représentation d'un pixel dans ce format ?
Quell information particulièrement importante pour l'affichage de l'image le fichier doit-il contenir et qui n'était pas présente dans les formats PBM, PGM et PPM ?

VII Représenter des sons

Un son est une variation de la pression de l'air au cours du temps.
Une manière simple de représenter un son consiste à l'échantillonner, c'est-à-dire à mesurer la pression à intervalles réguliers et représenter le son comme la suite des mesures obtenues.
L'échantillonnage d'un son est une méthode assez similaire au découpage d'une image en pixels, sauf que le découpage s'effectue, non dans l'espace, mais dans le temps.
Par exemple, si on échantillonne à 44 000 Hz, c'est-à-dire en faisant 44 000 mesures par seconde, une sinusoïde de fréquence 440 Hz, on fait cent mesures par période et on obtient l'échantillon suivant :



En revanche, si l'on échantillonne à 300 Hz cette même sinusoïde, on fait une mesure toutes les période et demie environ, et on obtient l'échantillon suivant :



On comprend qu'il est possible de reconstituer la sinusoïde dans le premier cas, mais non dans le second.
On échantillonne, en général, les sons à 44 000 Hz, car le son sinusoïdal le plus aigu que notre oreille peut entendre est de 22 000 Hz environ. Cette fréquence est relativement élevée, c'est pourquoi il faut plusieurs millions de bits pour représenter une minute de son et les fichiers audio sont souvent compressés. Cette méthode de représentation d'un son par échantillonnage est utilisée dans de nombreux formats :
– **formats simples** : RAW, WAV et BWF
– **formats plus sophistiqués** : MP3, WMA, AAC
A nouveau, ces formats se distinguent les uns des autres par la manière dont le son est représenté, par le fait qu'il est compressé ou non, par le fait que le format est public ou secret et par le fait qu'il est propriétaire ou libre.

Cette manière de représenter les sons par échantillonnage se distingue de la notation musicale, que nous utilisons depuis le XIII siècle et qui permet de représenter la durée, la fréquence et l'intensité de notes de musique, chacune représentée par un symbole sur une portée. Des systèmes intermédiaires entre l'échantillonnage et la notation musicale existent aussi, comme le format MIDI (Musical Instrument Digital Interface) utilisé pour représenter les sons produits par les instruments de musique électronique.