Codage

Laurent Signac

Laurent.Signac@univ-poitiers.fr

Codage numérique de l'information

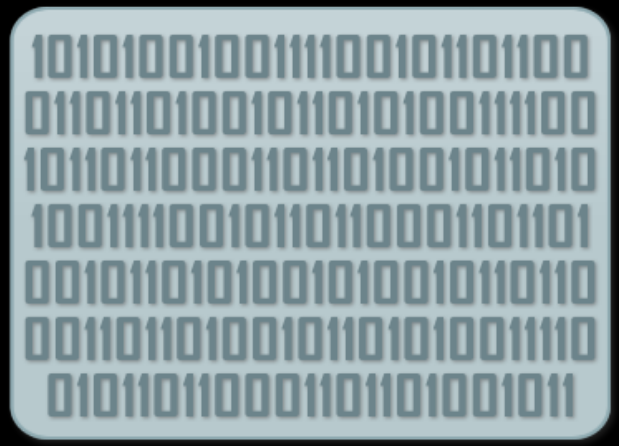
L'information est la matière première de l'informatique.

Les algorithmes, qui sont constitués d'informations :

- stockent,
- manipulent,
- transforment

d'autres informations, à l'aide de machines.

Tout, en informatique, est représenté comme une séquence de 0 et de 1 : les algorithmes, ou plutôt les programmes, le contenu d'un livre, une photo, une vidéo...



Il y a une raison théorique et une raison technique à l'utilisation du binaire :

- on ne peut pas faire plus simple que 2 symboles. Avec un seul symbole, plus rien ne fonctionne (on a un seul code d'une longueur donnée, la taille de l'écriture des nombres, écrit en unaire est proportionnelle au nombre, et non pas à son logarithme).
- les deux symboles 0 et 1 sont transposables électroniquement par : le courant passe ou ne passe pas (robuste)

Tout objet (numérique) est représentable sous forme de bits. La quantité d'information qu'il contient est justement le nombre de bits nécessaires pour le représenter.

Unités

- le bit (*binary digit*) est l'unité d'information : 0 ou 1
- l'octet est un groupe de 8 bits (un *byte* vaut souvent 1 octet)

Pour les multiples, il y a 2 systèmes en concurrence :

- Le Système international :
  - un kilo-octet (ko) vaut 10^3 octets
  - un méga-octet (Mo) vaut 10^3 ko
  - un giga-octet (Go) vaut 10^3 Mo
  - un téra-octet (To) vaut 10^3 Go
  - un péta-octet (Po) vaut 10^3 To
- Norme Commis. Électrotech. Internationale :
  - un kibi-octet (Kio) vaut $2^{10} = 1024$ octets [1]
  - un mébi-octet (Mio) vaut 2^{10} Kio
  - un gibi-octet (Gio) vaut 2^{10} Mio
  - ...
- les débits se mesurent en bits/secondes (ou kbits/s ou Mbits/s avec les mêmes préfixes).

le suffixe B est parfois utilisé pour byte, à ne pas confondre avec le b de bits...

1. http://physics.nist.gov/cuu/Units/binary.html

Écriture en base b

- formée à partir de b symboles, appelés chiffres.
- les 10 chiffres de la base 10 sont : $0, 1, 2, 3, 4, 5, 6, 7, 8$ et 9.

Écriture en base b :

$$c_n c_{n-1} \ldots c_1 c_0$$

Valeur du nombre :

$$v(c_n) \times b^n + v(c_{n-1}) \times b^{n-1} + \ldots + v(c_1) \times b^1 + v(c_0) \times b^0$$

$v(c_i)$ est la *valeur* associée au chiffre (symbole) c_i.

Quels chiffres utiliser ?

Pour une base $b < 10$, on utilise les même chiffres qu'en base 10. Ensuite, on ajoute des lettres. En base 16 : $0, 1, 2, \ldots, 9, A, B, C, D, E, F$.

Les bits situés à droite de l'écriture sont dits de poids faible, ceux situés à gauche, de poids fort.

Nombres entiers positifs : exemples

- Base 10 : $567|_{10} = 5 \times 10^2 + 6 \times 10^1 + 7 \times 10^0 = 567$
- Base 2 : $1101|_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13$
- Base 16 : $4C3|_{16} = 4 \times 16^2 + 12 \times 16^1 + 3 \times 16^0 = 1219$

Exemples de conversion de/vers la base 2

Base 2 (ou 16) ⇒ base 10

$$1011010|_2 = 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 90$$

$$5A|_{16} = 5 \times 16^1 + 10 \times 16^0 = 90$$

Base 10 ⇒ base 2

$$\begin{array}{rll} 90 & = 2 \times 45 & +0 \\ 45 & = 2 \times 22 & +1 \\ 22 & = 2 \times 11 & +0 \\ 11 & = 2 \times 5 & +1 \\ 5 & = 2 \times 2 & +1 \\ 2 & = 2 \times 1 & +0 \\ 1 & = 2 \times 0 & +1 \end{array}$$

La séquence des restes successifs, lue du dernier au premier donne : 1011010 qui est l'écriture en base 2 de 90.

Activités

Numération shadoks

Les Shadoks comptent en base 4. Leurs 4 chiffres sont GA, BU, ZO et MEU (ici classés par ordre croissant de valeur). Combien vaut BUGAZO ? Comment écrire 364 en numération Shadok ?

Il y a 10 sortes de gens, ceux qui savent compter en binaire et les autres.

Non, en fait, il y a 10 sortes de gens, ceux qui comprennent le ternaire, ceux qui ne le comprennent pas, et ceux qui croient que c'est du binaire.

Nombres entiers négatifs

Complément à 2 sur 8 bits

- $00000000|_2$ à $01111111|_2$: 0 à 127
- $10000000|_2$ à $11111111|_2$: -128 à -1

$\Rightarrow$ roue des entiers

Méthode du complément à 2

1. inverser les 0 et les 1
2. ajouter 1

Propriétés

- le complément à 2 est involutif
- l'addition (et la multiplication) fonctionnent

$84 - 102 = 01010100|_2 + CAD(01100110|_2) = 01010100|_2 + 10011010|_2$
$= 11101110|_2 = CAD(00010010|_2) = -18$

Nombres à virgule

Base 2 ⇒ base 10

On peut coder de la même manière les nombres non-entiers. Chaque chiffre placé après la virgule est associé à une puissance négative de la base :

$$101,011|_2 = 1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}+1\times2^{-3} = 5+0,25+0,125 = 5,375$$

Base 10 ⇒ base 2

$$\begin{array}{rll} 0,6875 & \times 2 = & 1,375 \\ 0,375 & \times 2 = & 0,75 \\ 0,75 & \times 2 = & 1,5 \\ 0,5 & \times 2 = & 1 \end{array}$$

Nous prenons les parties entières des résultats dans l'ordre où elles sont obtenues : 1011.
Donc $0,6875 = 0,1011|_2$.

Développement binaires infinis

- Écriture en base 2 finie $\Rightarrow$ écriture en base 10 finie
- Écriture en base 10 finie $\not\Rightarrow$ écriture en base 2 finie

$$\begin{aligned} 0,2 \quad &\times 2 = \quad 0,4 \\ 0,4 \quad &\times 2 = \quad 0,8 \\ 0,8 \quad &\times 2 = \quad 1,6 \\ 0,6 \quad &\times 2 = \quad 1,2 \\ 0,2 \quad &\times 2 = \quad ... \end{aligned}$$

En conséquence : $0,2 = 0,00110\overline{0011}...$

En base b, un nombre $\frac{p}{q}$ (fraction irréductible) a une écriture finie si tous les facteurs premiers de q sont aussi des facteurs premiers de b. En binaire, seuls les nombres de la forme $\frac{p}{2^k}$ ont une écriture finie.

Patriot Missile Failure

Le temps est mesuré par les missiles Patriot par une horloge cadencée au dixième de seconde, dès leur mise sous tension.
Une durée écoulée est donc le nombre de tics $\times \frac{1}{10}$. Le nombre $1/10$ est codé en virgule fixe sur 24 bits :

$$0.1 = 0.0001100110011001100\overline{1100}$$

L'erreur commise sur la fraction est donc :

$$0.0000000000000000000000\overline{1100} \approx 9.54 \times 10^{-8}$$

Une durée de 100 heures de surveillance, tout à fait plausible, implique donc une erreur de $100 \times 3600 \times 10 \times 9.54 \times 10^{-8} = 0,34s$.
Une erreur de $0,34s$ sur la position d'un missile Scud (vitesse $\approx 1\,676 m/s$) correspond à une distance de plus de 500 mètres $\Rightarrow$ hors fenêtre.
Cet incident qui a coûté la vie à plusieurs dizaines de personnes le 25 février 1991.
source : `http://www.fas.org/spp/starwars/gao/im92026.htm` (rapport officiel GAO)

Virgule flottante

Les nombres à virgule flottante ne sont pas les réels (IEEE 754)

$$(-1)^s \times 1, \underbrace{\ldots}_{m} \times 2^e$$

$$25,90625 \Rightarrow 11001,111101 \Rightarrow 1.1001111101 \times 2^4 \times (-1)^0$$

Simple précision : $\underbrace{0}_{s}$ $\overbrace{10000011}^{E:8 \text{ bits } (=e+127)}$ $\underbrace{10011110100\ldots0}_{23 \text{ bits}}$

- Normalisés : $0 < E < 255$
- Dénormalisés : $E = 0$ et $m \neq 0 \Rightarrow 0,m \times 2^{-126} \times (-1)^s$
- Infinis : $E = 255$ et $m = 0 \Rightarrow (-1)^s \infty$
- Indéfini : $E = 255$ et $m \neq 0$

Flottants $\neq$ réels

Il y a des nombres incodables, donc des erreurs d'arondi.

En simple précision :

$$\begin{array}{rll} 2^{127} & \text{se code} & 01111111\,0\overbrace{0\ldots0}^{22}\,0 \\ 2^{127}+2^{104} & \text{se code} & 01111111\,\underbrace{0\ldots0}_{22}\,1 \end{array}$$

Activité

Calculer les bornes des nombres normalisés et dénormalisés, calculer la taille des trous.

- Le plus petit nombre non nul positif vaut :

$$2^{-126} \times 0.\underbrace{0\cdots0}_{22 \text{ fois}}1 = 2^{-149} \approx 1.4013 \times 10^{-45}$$

- Le plus grand nombre positif vaut :

$$2^{127} \times 1.\underbrace{1\cdots1}_{23 \text{ fois}} = 2^{128} - 2^{104} \approx 3.4 \times 10^{38}$$

- Entre deux nombres, il y a un «trou». Par exemple :

$$0\,11111110\,\underbrace{0\cdots0}_{23 \text{ fois}} = 2^{127}$$

- et le nombre suivant est :

$$0\,11111110\,\underbrace{0\cdots0}_{22 \text{ fois}}1 = 2^{127} + 2^{104}$$

Activité : avec Python

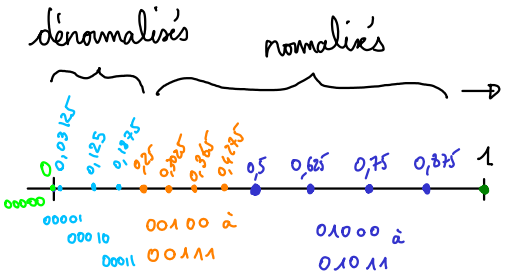
```
>>> a = 1 - 0.2 - 0.2 - 0.2 - 0.2 - 0.2
>>> b = 1 - (0.2 + 0.2 + 0.2 + 0.2 + 0.2)
>>> a, b
(5.551115123125783e-17, 0.0)
>>> c = 1.0 * 2**53
>>> c + 1.0 + 1.0, 1.0 + 1.0 + c
(9007199254740992.0, 9007199254740994.0)
```

$\Rightarrow$ avec les flottants, on n'ajoute pas un petit nombre à un grand nombre.

Activité : Codage vf sur 6 bits

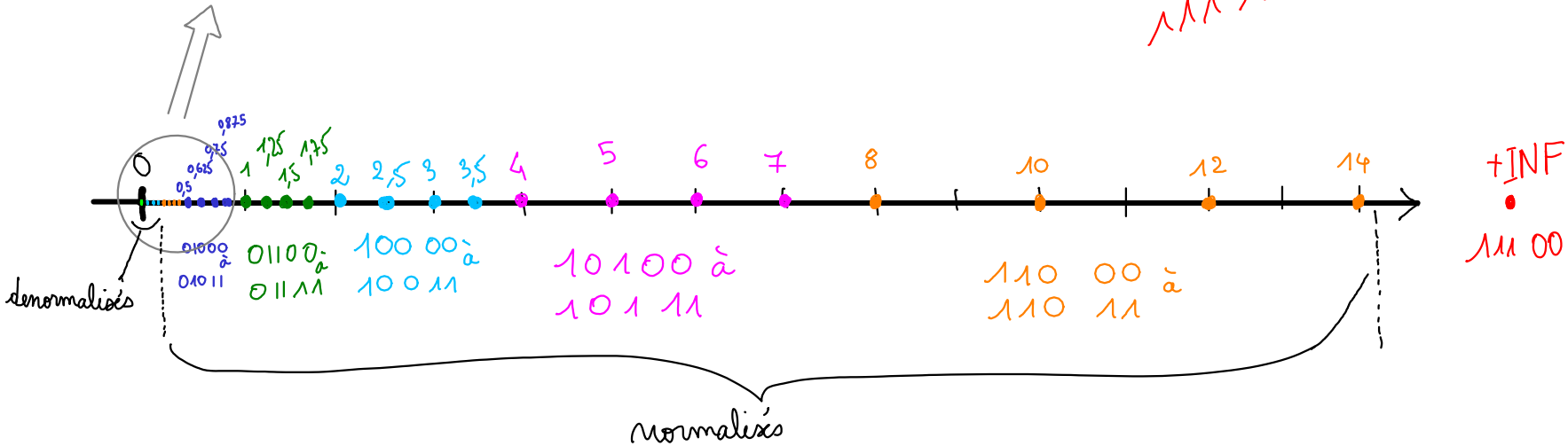
Codage en virgule flottante sur 6 bits

3 bits exposant (E = e + 3)
2 bits mantisse
1 bit de signe (à 0 ci-dessous)



NotaNumber

11101
11110
11111



Ariane 5

4 Juin 1996

le vol 501 (Ariane 5) explose 37 secondes après le décollage.

Le problème vient d'une erreur logicielle.

Un nombre flottant, codé sur 64 bits est convertit en entier signé. Sa valeur dépasse le plus grand entier signé représentable sur 16 bits et la conversion échoue (exception) provoquant une erreur dans le système de référence inertielle.

Source : `http://esamultimedia.esa.int/docs/esa-x-1819eng.pdf`

Exemple du codage des couleurs

- Couleurs sur écran : synthèse additive (addition de la lumière) ($\neq$ impression, synthèse soustractive)
- Couleurs primaires en synthèse additive : rouge, vert et bleu (synthèse soustractive : cyan, magenta, jaune)
- 3 Leds (une rouge, une verte et une bleue) qu'on peut choisir d'allumer ou d'éteindre (on ne peut pas varier leur intensité).

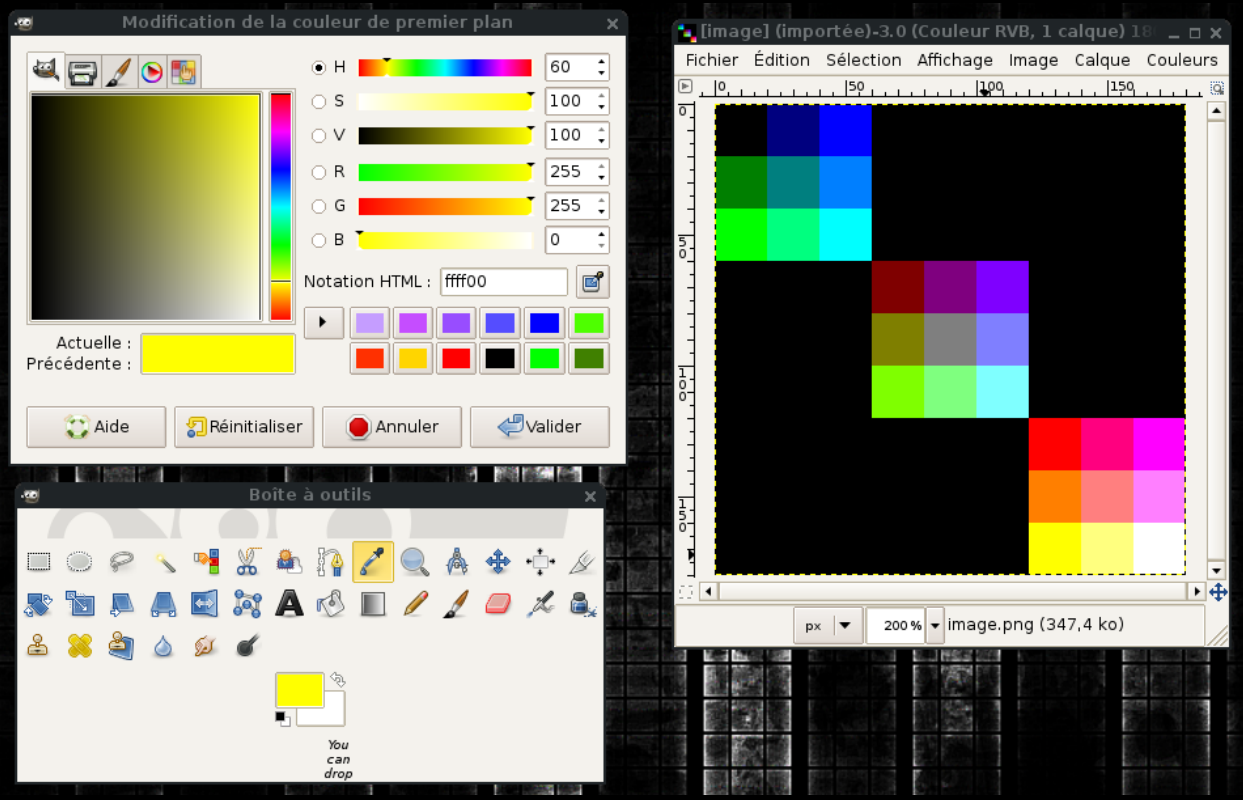
Combien de couleur obient-on ? Lequelles ?

Du noir, si tout est éteint, du rouge, du vert et du bleu, si on en allume une seule, du blanc, si on allume les trois, et du jaune, du magenta et du cyan, si on en allume exactement deux.

Comment représenter ces couleurs ?

Associer un bit d'information à chaque led, indiquant si elle est éteinte (0) ou allumée (1). La donnée d'une des huit couleurs représente dont 3 bits d'information : 000 noir, 001 bleu, 010 vert, 011 cyan, 100 rouge, 101 magenta, 110 jaune, 111 blanc.

Activité : Codage des couleurs dans un logiciel de dessin bitmap



- Observer les différents carrés de couleur
- Trouver la logique de la représentation (axe bleu horizontal, axe vert vertical et rouge diagonal)
- Utiliser Gimp pour accéder aux composantes RVB des couleurs
- Noter la décomposition peu intuitive en synthèse additive (R + V donne jaune)
- Observer le codage en hexadécimal des couleurs
- Noter qu'il existe plusieurs espaces de représentation des couleurs (RVB, HSV...)

Besoin de standardisation

Codage et standard sont complémentaires

- L'ordre des couleurs primaires dans la représentation est arbitraire.
- Le respect de ce choix (adoption d'un standard) facilite grandement les échanges : Deux programmes, par exemple, utilisant ce même standard pourront se communiquer des couleurs. Il permet à d'autres personnes d'écrire aussi des programmes utilisant les données issues des deux autres.
- Si ce standard est publié, c'est un standard ouvert. S'il n'est pas publié, c'est un standard fermé, qui empêche le développement d'applications compatibles (on peut alors pratiquer l'ingénierie inverse pour trouver la convention utilisée).

Il existe de nombreux standards de codage de l'information : pour le texte, les images, la vidéo etc...

Standard Texte

- Plus ancien standard : ASCII contient les caractères latins non accentués et des caractères de contrôle. Codage sur 7 bits.

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

- Rapidement ce codage n'a plus été suffisant. Naissance d'autres codages :
  - extensions : ajout de bits d'information pour avoir des caractères en plus de l'ASCII. Exemples : latin-1 (iso-8859-1). Problème, il y a des extensions pour le chinois, japonais, cyrillique, arabe, etc...
  - unicode : recense tous les caractères dans une table qui en compte plusieurs dizaines de milliers. Ces codes sont ensuite représentés, en utilisant un codage type UTF-8 ou UTF-16

Activité : Encodage des caractères

- Ouvrir le fichier pagewebinfoencodage.html avec un navigateur. Constater qu'il y a un problème avec les accents (qu'on rencontre parfois sur de vraies pages).
- Noter dans le source la déclaration de l'encodage : latin1
- Forcer le navigateur à utiliser un autre encodage (ici Unicode (en fait Utf8))
- Si on dispose d'un moyen de visualiser des fichiers en mode «binaire», rechercher le code des lettres accentuées

```
70 3E 0A C3 80 20 62 69        p > . ? ? . b i
C3 80 en latin 1 : Ã Euro
C3 80 en utf-8   : À
```

https://fr.wikipedia.org/wiki/ISO/CEI_8859-1
https://fr.wikipedia.org/wiki/UTF-8

Texte enrichi

- Le texte enrichi est du texte avec de la mise en forme (gras, italique), des titres, voire des liens etc...
- Il y a de nombreux codages : RTF, HTML, fichier d'une suite bureautique...

Exemple :

```
<html>
<body>
<h1>Algorithmique</h1>
<p>
L'<b>algorithmique</b> est l'ensemble des règles et des techniques
qui sont impliquées dans la définition et la conception
d'<a href="http://fr.wikipedia.org/wiki/Algorithme">algorithmes</a>,
c'est-à-dire de processus systématiques de résolution
d'un problème permettant de décrire les étapes vers le résultat.
En d'autres termes, un algorithme est une suite finie et non-ambiguë
d'instructions permettant de donner la réponse à un problème.
</p>
</body>
</html>
```

Activité : Codage de la mise en forme en html

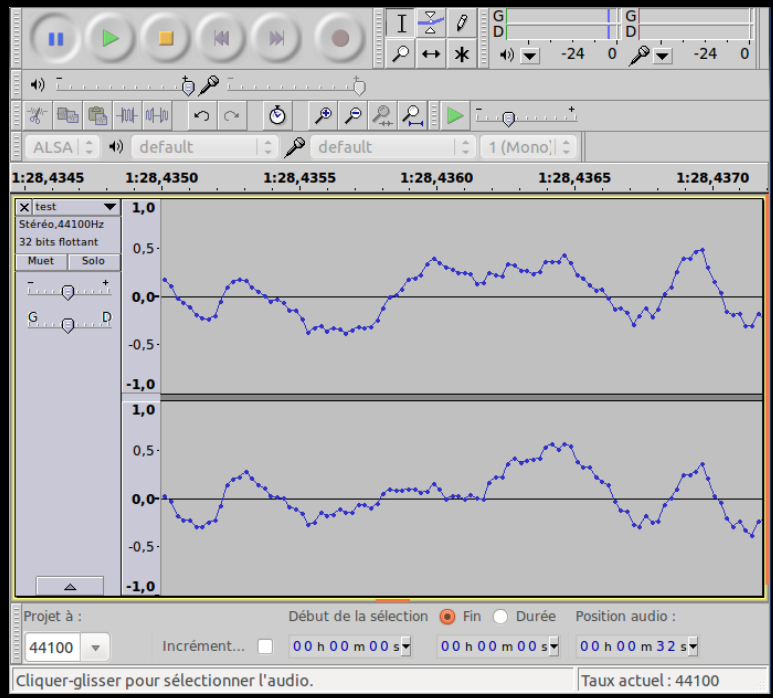
- Ouvrir le fichier ▸ pagewebalgo.html avec le bloc note puis avec un navigateur Web.
- Noter comment le navigateur interprète HTML pour afficher la mise en forme.
- Dans le navigateur, faire : "Voir Source de la page", éventuellement sur une page Web accessible sur Internet

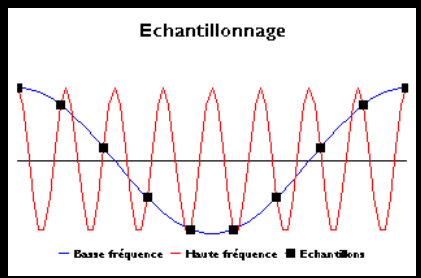
Coder un son (ou une courbe en général)

- choisir correctement la période d'échantillonnage en fonction du signal ;
- échantillonner la courbe ;
- chaque échantillon est un nombre, représenté comme nous venons de l'indiquer (quantification)

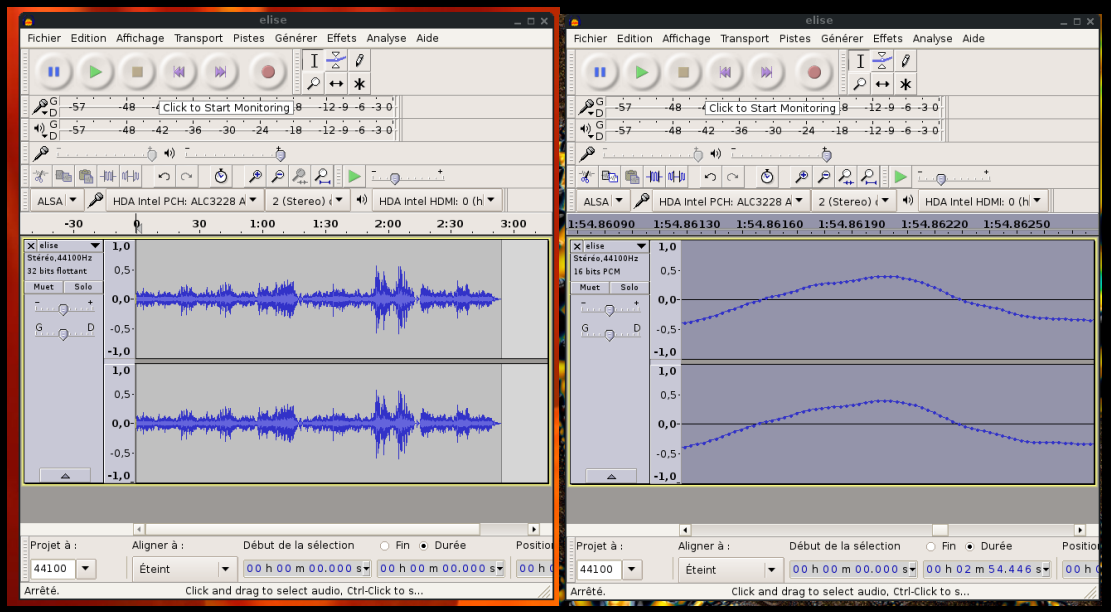
Théorème de Nyquist-Shannon

Il faut une fréquence d'échantillonnage double de la fréquence max du signal pour le reconstruire correctement. L'oreille perçoit jusqu'à 22000 Hz $\Rightarrow$ l'échantillonage de musique est fait à 44100 Hz.





Activité : Visualiser des sons avec Audacity



- Ouvrir le fichier ▸ elise.wav avec audacity
- Zoomer jusqu'à voir un échantillon
- Noter les 2 voies
- Noter la fréquence d'échantillonage
- Noter la durée du morceau
- ! Pas d'info sur le codage de chaque échantillon (`file` ou `exiftool`)

Activité : Taille d'un fichier audio (1)

```
$ file elise.wav
elise.wav: RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, stereo 44100 Hz

$ exiftool elise.wav
ExifTool Version Number         : 10.20
File Name                       : elise.wav
Directory                       : .
File Size                       : 29 MB
File Modification Date/Time     : 2016:09:13 14:18:00+02:00
File Access Date/Time           : 2016:09:13 14:18:38+02:00
File Inode Change Date/Time     : 2016:09:13 14:18:00+02:00
File Permissions                : rw-r--r--
File Type                       : WAV
File Type Extension             : wav
MIME Type                       : audio/x-wav
Encoding                        : Microsoft PCM
Num Channels                    : 2
Sample Rate                     : 44100
Avg Bytes Per Sec               : 176400
Bits Per Sample                 : 16
Duration                        : 0:02:54
```

Activité : Taille d'un fichier audio (2)

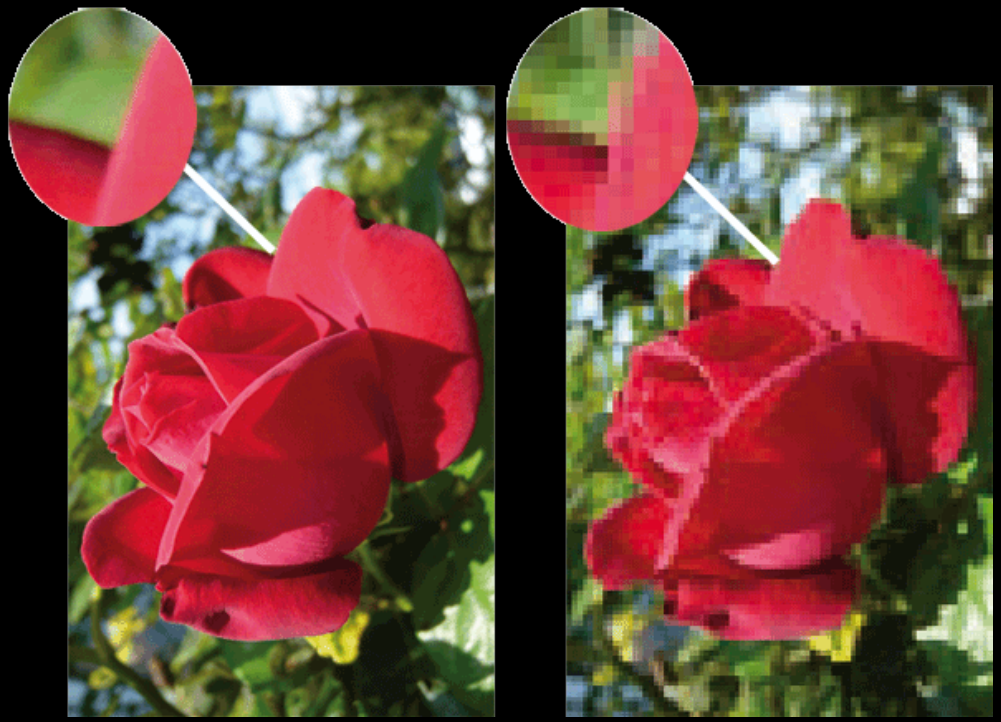
```
(2*60+54) * 44100 * 2 * 2
30693600

$ ls -l
total 45720
-rw-r--r-- 1 signac signac     9156 13 sept. 14:18 elise.mid
-rw-r--r-- 1 signac signac  2792384 13 sept. 14:18 elise.mp3
-rw-r--r-- 1 signac signac 30772324 13 sept. 14:18 elise.wav
```

Pistes...

- Pourquoi le fichier .mp3 est il plus petit ? $\Rightarrow$ compression avec et sans perte (parler de flac).
- Que contient le fichier le plus petit elise.mid ?

Coder une image bitmap

- Tableau bidimensionnels de nombres (on dit pixel).
- Chaque pixel est codé en RGBA (ou autre).
- Chaque composante peut être sur 8 bits (quantification).



Format d'image

les différents formats d'images correspondent à différentes façons de stocker (de manière compressée, avec ou sans perte) un tel tableau de pixels : Jpeg, Gif, Png...

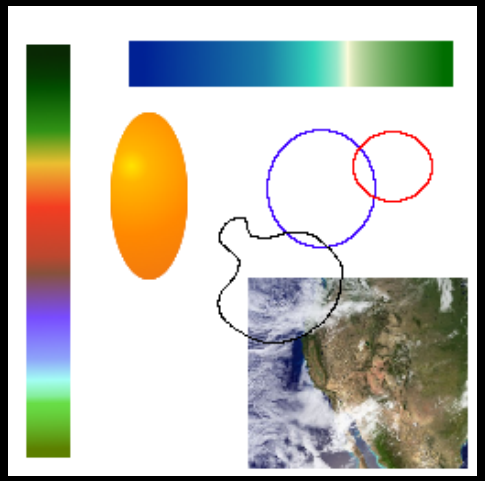
Activité : Résolution / Quantification d'une image



- Quantification :
  - Ouvrir ada.png et ada.gif
  - Vérifier qu'elles ont la même résolution (même nombre de pixels, pas de dpi !)
  - Comparer les couleurs sur un dégradé en zoomant (joue par exemple)
- Résolution :
  - Ouvir ada.png et adasmall.png
  - Les afficher de la même taille à l'écran
  - Comparer les résolution (en pixel)
  - Zoomer

Activité : Formats d'images

Il existe une grande quantité de formats d'images. À partir des images fournies `scene.bmp`, `scene.gif`, `scene.png`, `scene.jpg` et de leur taille en kilo-octets que pouvez-vous déduire des différents formats et de leur utilisation ?



- La bmp est parfaite et non compressées ($256^2 \times 3$)
- La png est parfaite et compressée
- Il manque des couleurs sur la gif (voir les dégradés)
- La jpg est la mieux compressée, mais c'est une compression avec perte (artefacts dans les dessins au trait) (OK pour les photos par contre...)

Codage symbolique

- Coder un son n'est pas coder une partition, coder une image n'est pas coder un dessin
- La partition et le dessin (ou la figure) sont à un niveau supérieur d'abstraction : On ne peut pas représenter tous les sons avec une partition ni toutes les images avec une description géométrique.

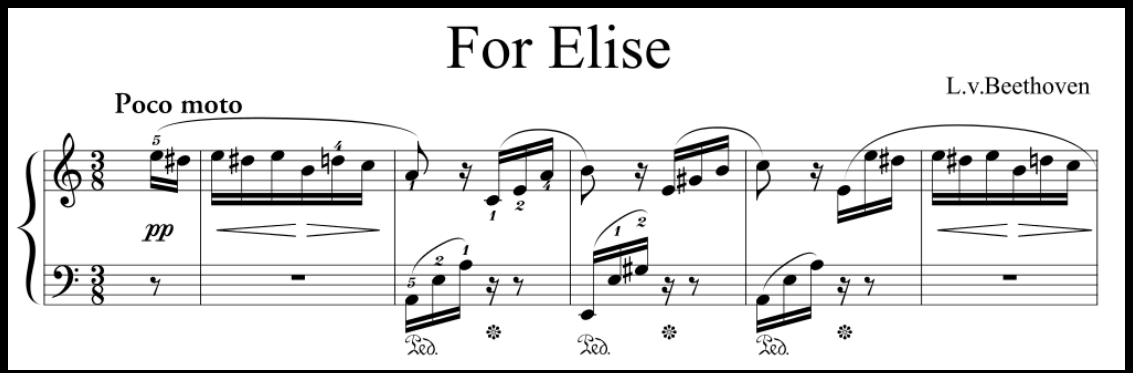
Exemples de descriptions symboliques

- MIDI pour la musique (on donne l'instrument, la hauteur de la note, sa longueur, plutôt qu'une numérisation du son)
- les formats vectoriels (ps, pdf, ai, swf, svg) pour les images

Midi - Son

Exemple de codage MIDI

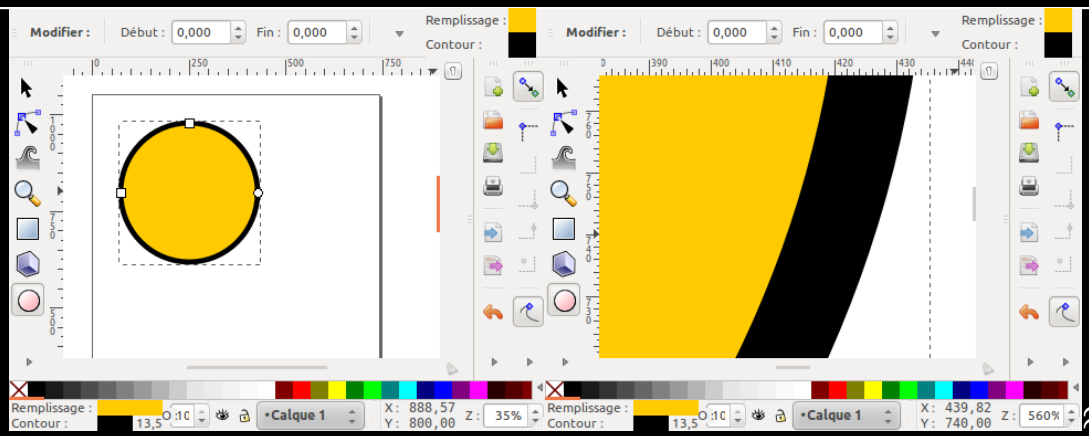
- 10010001 : 1 (status) 001 (note on) 0001 (canal 1)
- 00011000 : 0 (data), 0011000 (Numéro de la note (ici C0))
- 00100000 : 0 (data), 0100000 (Vélocité)



- Durée : 3 min.
- Fichier midi ≈ 10 Ko
- Fichier wav ≈ ...

Image bitmap - Image vectorielle

```
<path
  d="m 268.57143,179.50504 a 98.571426,107.14286 0 1 1 -197.142856,
    0 98.571426,107.14286 0 1 1 197.142856,0 z"
  transform="matrix(1.7969231,0,0,1.6798359,-54.04835,-46.319673)"
  id="path2985"
  style="fill:#ffcb00;fill-opacity:1;stroke:#000000;stroke-width:7.79699993;
   stroke-linecap:round;stroke-miterlimit:3.82500005;stroke-opacity:1;
   stroke-dasharray:none"
  />
```



$\approx$ 400 char

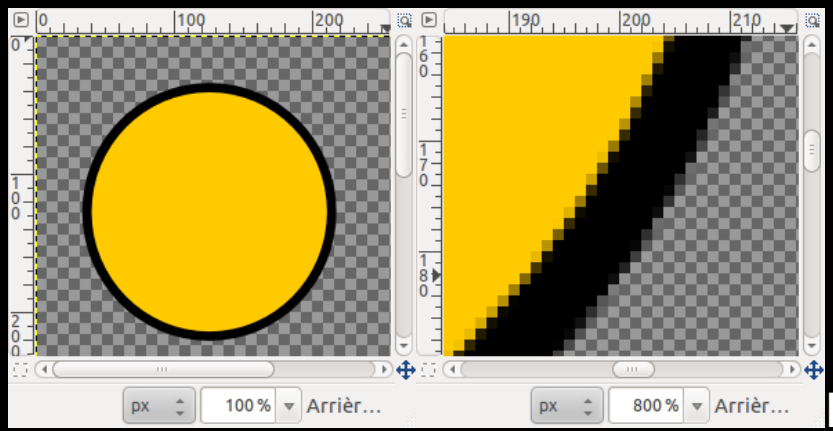


Image 200×200

Activité : images vectorielles ou bitmap

Les fichiers dessin.svg et dessin.png contiennent les mêmes objets en bitmap et vectoriel. Zoomer sur les deux et observer le comportement.
Le format PDF peut contenir du dessin vectoriel, mais aussi embarquer des images bitmap (généralement codées en jpg). Pensez qu'une image en PDF est forcément vectorielle est une erreur.
Comparez les fichiers dessin1.pdf et dessin2.pdf. Expliquez.
Comparez les tailles en octet des 4 fichiers :

```
> ls -l dessin*
-rw-r--r-- 1 dali dali  5996 17 sept. 22:42 dessin1.pdf
-rw-r--r-- 1 dali dali 38286 17 sept. 22:43 dessin2.pdf
-rw-r--r-- 1 dali dali 35871 17 sept. 22:37 dessin.png
-rw-r--r-- 1 dali dali  3849 17 sept. 22:34 dessin.svg
```

Documents fournis

- Diaporama à disposition
- Fichiers d'activités fournis
- Documents complémentaires