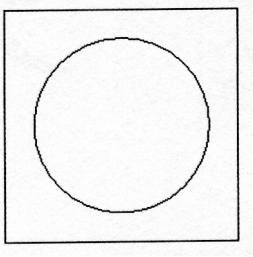
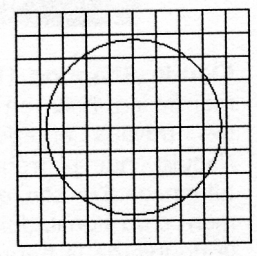
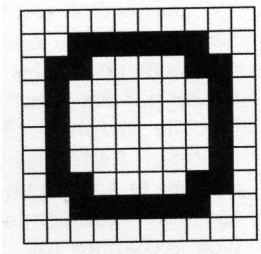
Représentation des images et des sons en informatique.

1. L'image.
   a. *Introduction.*

Pour représenter l'image ci-contre, nous avons vu deux méthodes.



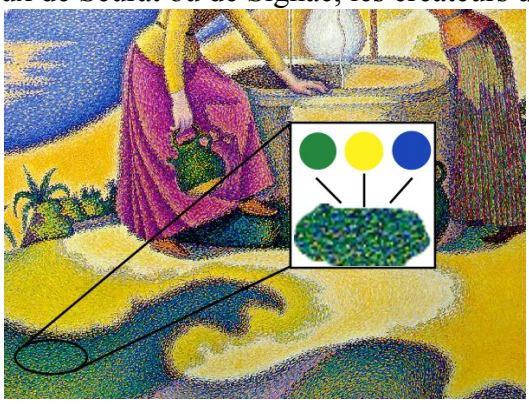
- La première est de dire : « cette image est un cercle de centre Ω et de rayon R = 4 ». C'est la représentation dite vectorielle, ou symbolique, de l'image.
- La deuxième est de superposer à l'image un quadrillage. Chacune des cases est un pixel. On indique alors pour chaque carreau la couleur du pixel : « blanc blanc noir etc… ». Pour gagner de la place, on codera sur un bit blanc et noir, respectivement par 0 et 1. Plus le quadrillage est fin, et plus la description de l'image sera précise ; à partir de quelques millions de pixels l'œil ne fait plus la différence. Cette représentation est dite matricielle, ou « bitmap ».


→

→

0000000000
0011111100
0110000110
0100000010
0100000010
0100000010
0100000010
0110000110
0011111100
0000000000

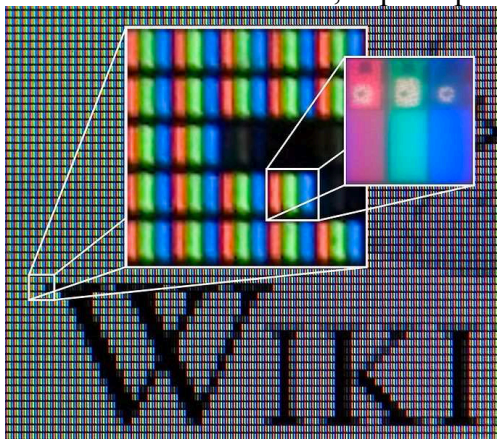
Codage bitmap sur 100 pixels (grille 10×10)

- L'idée est donc de convenir d'un standard pour tous : ici on décrit les pixels de haut en bas et de la droite vers la gauche, et non pas, par exemple, en spirale en partant du centre. Cette dernière méthode n'est pas si idiote qu'elle en a l'air : elle évite à une machine de dessin automatique de faire des grands trajets, et donc gagne du temps. De même il a été convenu que blanc est codé par 0 et noir par 1.
- On peut généraliser ces principes aux images en niveaux de gris ou en couleurs (voir les tableaux de Seurat ou de Signac, les créateurs du pointillisme)



Détail de « Femmes au puits »
Paul Signac 1892

- Sur un écran d'ordinateur, le principe est le même :



Cette image est un agrandissement d'une partie de l'écran, composé d'une grille de pixels. Chaque pixel est composé de trois couleurs. Le premier zoom est un groupe de $5 \times 5 = 25$ pixels. Trois d'entre eux sont presque éteints, ils correspondent au haut du « W ». Le deuxième zoom montre un pixel unique. Le mélange des trois couleurs primaires, suivant leur intensité respective, donne de nombreuses couleurs.

b. Formats d'images matricielles sans compression.

Le suffixe du fichier permet d'identifier le type d'objet codé dans le fichier, par exemple .txt pour du texte, .py pour un programme Python, .odf pour un document OpenOffice, etc…

i. Les images en noir et blanc peuvent être codés dans le format PBM (Portable Bit Map, suffixe .pbm)

Un fichier pbm ASCII se compose comme suit :

- Un nombre magique (P1)
- Un caractère d'espacement (espace, tabulation, nouvelle ligne)
- Largeur de l'image (codée en caractères ASCII)
- Un caractère d'espacement
- Hauteur de l'image (codée en caractères ASCII)
- Un caractère d'espacement
- Données ASCII de l'image :
- L'image est codée ligne par ligne en partant du haut
- Chaque ligne est codée de gauche à droite
- Un pixel noir est codé par un caractère 1, un pixel blanc est codé par un caractère 0
- Les caractères d'espacement à l'intérieur de cette section sont ignorés
- Aucune ligne ne doit dépasser 70 caractères.
- un zéro final.
- Toutes les lignes commençant par # sont ignorées.

Un exemple du codage de la lettre « J » :

```
P1
# exemple de la lettre "J", format de police 6 points !
6 10
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
1 0 0 0 1 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

ii. De la même manière il existe des formats PGM et PPM pour les niveaux de gris et la couleur.

Remarquez que la taille des fichiers devient vite très grande :

- La taille du fichier pour une image de $250 \times 167 = 41750$ pixels en PBM est de 8 Ko.
- En PGM, si l'on code sur $65536 = 2^{16}$ niveaux de gris, il faut deux octets pour coder chaque pixel : on passe à 45 Ko (ce qui fait bien plus que 8×2 …)
- En PPM, chaque couleur est codée par 3 octets (un pour le rouge, un pour le vert et le dernier pour le bleu), soit 127 Ko (même remarque : c'est plus que 8×3…). Remarquez que les trois couleurs de base ne sont pas les mêmes suivant que l'on imprime (ou dessine) en synthèse soustractive, ou bien que l'on affiche à l'écran. Dans le premier cas (feuille blanche), la synthèse des couleurs est soustractive ; les couleurs de base sont cyan, magenta et jaune. Dans le deuxième cas (écran noir), la synthèse est additive ; le couleurs de base sont rouge, vert, bleu (jouez avec les réglages sur votre écran de télévision).

D'autres formats sans compression existent. Par exemple, sous le format BMP (windows bitmap) une image en 1200 × 1600 (1200 pixels de large par 1600 pixels de haut), en 24 bits (16,8 millions de couleurs) aura une taille de (1200 × 1600 × 24) bits soit 5,76 Mo

(5,5 Mio ; 1 Mio = 2^{20} octets). C'est-à-dire sur un dvd de 4,7 Go, il tient 816 images, soit… 33 secondes de film à 24 images par seconde.

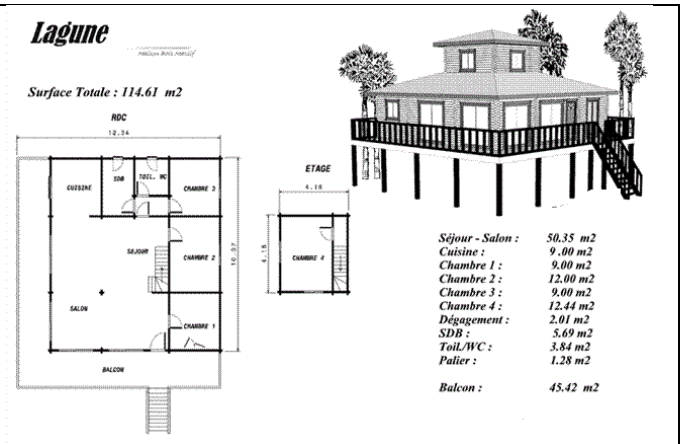
c. *Formats d'images matricielles avec compression*

Les exemples précédents montrent tout l'intérêt de la compression de données. Certains formats, comme le PNG (portable network graphic) compressent l'image sans perte d'information : lors de la conversion de l'image du format BMP en PNG, l'image prend moins de place mais est de même qualité à l'ouverture. D'autres, comme le JPEG (joint photographic experts group), sont des formats avec perte d'information, un des paramètres permettant de régler cette perte. Lors du passage de format, la qualité de l'image est plus ou moins dégradée, et elle peut être alors bien moins lourde.

Sur l'image ci-contre, le taux de compression en jpeg augmente de la gauche vers la droite. On peut alors diviser le poids de l'image par 25.

d. *Formats d'images vectorielles*

Le format matriciel ne permet pas de satisfaire tous les besoins. Les plans d'architectes, les dessins techniques ont besoin de coder des informations très précises pour les besoins de l'utilisateur. Des logiciels comme OpenOffice Draw, ou Inkscape (libres tous les deux) permettent de faire du dessin vectoriel. Au lieu de préciser pixel par pixel la composition de l'image, le fichier contient la liste des éléments géométriques constitutifs de l'image (cercles, rectangles, segments, courbes de Bézier qui permettent de dessiner des tracés complexes,…) ainsi que leurs caractéristiques (par exemple les coordonnées du centre d'un cercle, son rayon, ainsi que sa couleur de remplissage). Il est alors nécessaire de transformer l'image en format matriciel pour pouvoir l'afficher à l'écran. Suivant le type de travail à effectuer sur l'image, on choisira soit le format matriciel soit le format vectoriel. On peut augmenter la résolution d'une image vectorielle sans perte de qualité avec le même poids, contrairement à une image matricielle. Mais on ne peut pas dire qu'une image vectorielle nécessite moins de mémoire en général. PDF, Postscript (pour les imprimantes), SVG et Adobe Flash sont des formats vectoriels.

	
redimensionnement d'une image : à gauche en vectoriel, à droite en matriciel	Un plan d'architecte réalisé en vectoriel, et le rendu en 3d.

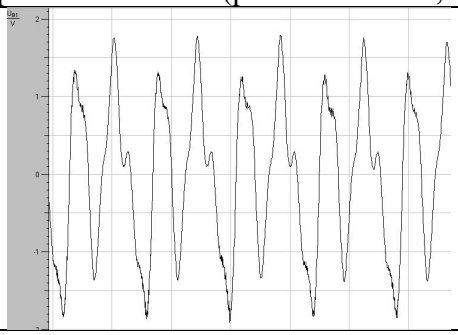
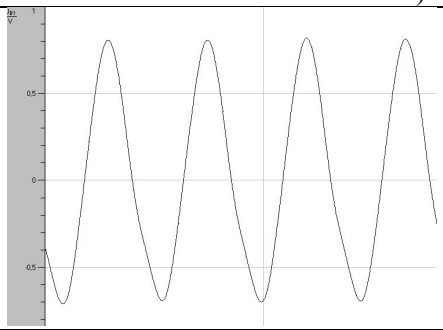
2. Le son.

a. *Définition et caractéristiques*

Un son est une onde produite par la vibration mécanique d'un support fluide ou solide. Il se propage dans le milieu ambiant. Vous pouvez comparer la propagation dans l'air et sous l'eau pour voir que le milieu influence énormément sur la propagation du son. Par extension, le son désigne la sensation auditive.

La façon la plus simple de mesurer un son est de mesurer la variation de dans le temps de la pression acoustique. Si la pression est mesurée en un point de l'espace, le son est « mono », en deux points « stéréo », en plus de deux points « multicanal ».

Pour être perceptible par l'oreille humaine, un son doit posséder certaines caractéristiques. La plus importante est la fréquence. La fréquence d'un son perceptible par un humain est *grosso modo* comprise entre 20 Hz et 22 000 Hz (des graves aux aigus). Les cétacés (dauphins, baleines,…) et les chiroptères (chauves-souris) perçoivent et émettent des ultra-sons jusqu'à 100 Khz (150 Khz pour les cétacés). Les chauves-souris s'en servent comme radar (écholocation) pour capturer les insectes de nuit. Les humains peuvent ressentir les vibrations des infra-sons en dessous de 20Hz par la vibration. Les éléphants, girafes et baleines communiquent à l'aide d'infrasons, qui se propagent très loin. Cela va jusqu'à des centaines de kilomètres sous l'eau, car les baleines et cachalots crient très fort, jusqu'à 230 décibels (pour un humain, le niveau de douleur est entre 120 et 130 décibels).

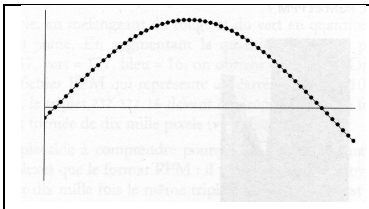
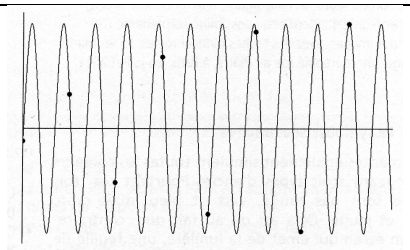
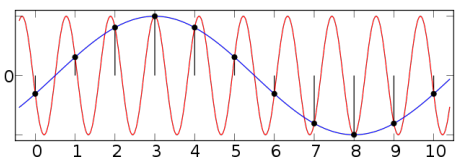
	
Enregistrement correspondant au son « Oooooooo » : plusieurs sinusoïdes superposées	Enregistrement du son émis par un diapason (note : La 440 Hz) : une seule sinusoïde

b. *Représentation et échantillonnage.*

La représentation des ondes sous forme continue, comme sur les schémas ci-dessus, ne convient pas à la représentation discrète des ordinateurs (valeurs « séparées » les unes des autres). On doit donc échantillonner des points sur la courbe, c'est-à-dire que l'on ne prend que quelques points.

Un peu de physique : supposons que l'on échantillonne à 44 000 Hz notre diapason à 440 Hz. On aura donc pour chaque période (chaque oscillation) du diapason 100 points, puisque l'échantillonnage est « 100 fois plus rapide ». On aura une reproduction correcte. En revanche, si l'on échantillonne à 300 Hz, on n'aura plus qu'un point toutes les périodes et demie environ, et il sera impossible de reconstituer la sinusoïde. On peut même prendre une sinusoïde pour une autre avec une fréquence d'échantillonnage trop faible (aliasing ou repli de spectre).

La théorie de l'information (théorème de Nyquist-Shannon) nous dit qu'il faut au moins deux points par oscillation du signal pour pouvoir reproduire fidèlement sa dynamique lors du codage numérique. Dès lors, puisque l'oreille humaine perçoit des sons jusqu'à 22 KHz, il faut 44 000 points par seconde.

		
Echantillonnage à 44 000 KHz du La à 440 Hz	Echantillonnage à 300 KHz du La à 440 Hz	Effet de repli de spectre

c. *Formats non compressés de codage du son*

Il faut 16 bits pour coder correctement la valeur de l'intensité du son à chaque instant, donc pour une minute de son échantillonnée à 44000 Hz il faut plus de 42 millions de bits (soit 5 Mo à peu près).

Il n'existe pas de format standard pour une telle représentation, mais plusieurs formats « propriétaires », lisibles grâce à des logiciels spécialisés appelés « CODECS » (pour codeurs-décodeurs). Par exemple, sous Windows le format est WAV, sous OsX c'est Aiff.

d. *Formats compressés de codage du son*

Les formats compressés sans perte existent, ils sont peu connus. Citons par exemple FLAC (free lossless audio codec), format libre.

Le plus connu des formats compressés avec perte est le MP3. Le poids des fichiers ainsi compressés est divisé par un facteur de 4 à 20. C'est un format propriétaire, des royalties importantes sont à payer pour exploiter cette licence. Vous connaissez certainement le WMA, crée par Microsoft à partir des recommandations du MPEG-4. Il existe également des formats libres, comme le Ogg Vorbis (suffixe .ogg ou .oga). Le format mp3 reste le plus utilisé malgré son coût, car l'arrivée continuelle de nouveaux formats, apportant un avantage assez faible par rapport aux précédents, ne permet pas de mettre en place un standard meilleur que le mp3 et lisible sur tous les baladeurs.

Quelques remarques pour finir :

- Avoir une très grande définition pour une image (un très grand nombre de pixels) ne garantit pas pour autant un bon rendu à l'impression. L'imprimante est aussi limitée (l'unité est le ppp, point par pouce en français ou dpi, dots per inch en anglais). Par ailleurs, la seule notion cohérente pour les logiciels de traitement d'image est celle de PPI (pixels per inch), puisque l'image est traitée à l'écran, en pixels, et non sur une imprimante (en points). Pour une impression en 300 DPI et une photo 10 × 15 cm en 300 PPI, il faut 2,1 Mpixels (1 181 × 1 772). Une photo 20 × 30 cm en 300 PPI nécessite quant à elle 8,4 Mpixels (2 362 × 3 543). Si l'image comporte plus de pixels, un algorithme calcule les pixels pour l'impression ; et cela peut donner un résultat assez moche (testé pour vous !).
- La qualité des fichiers compressés avec perte est bien souvent insuffisante pour les besoins professionnels. En photo, un professionnel préfèrera travailler avec le format RAW (brut sans aucun traitement). En audio, le format mp3 est aux oreilles d'un ingénieur du son ce qu'est pour nous la musique sortie d'un téléphone portable (ie insupportable).

Images (et son) en informatique - exercices.

Ex 1.

Compter le nombre de bits pour coder une image, par exemple 1024×1024, en noir ou blanc, puis avec des niveaux de gris de 0 à 255, puis avec un codage RVB, chaque couleur étant sur un octet.

Ex 2.

Ouvrir un fichier audio quelconque avec un logiciel comme audacity ou tout autre logiciel qui puisse lire les paramètres d'un fichier audio. Regarder la durée D du son en seconde, la fréquence F d'échantillonnage – 44 100 Hz ? Davantage ? – , le nombre C de canaux – mono ? stéréo ? –, et le nombre O d'octets pour coder un échantillon – 16 bits donc 2 octets ? Davantage ? Comparer la taille du fichier en octet au nombre D F × C × O et en déduire le taux de compression.

Ex 3.

Concevoir le même exercice que le précédent mais pour une image.

Ex 4.

Ouvrir une image au choix avec un logiciel d'édition élémentaire d'images tel que gimp ou tout autre logiciel similaire et essayer de modifier la lumière et le contraste : laquelle de ces opérations correspond à ajouter/retirer une valeur constante à la valeur d'un pixel et laquelle correspond à multiplier le pixel par un nombre plus grand/petit que un ?

Exercices : programmation Python de traitements d'images

Tous les programmes à faire se font en adaptant le programme donné dans le notebook « traitement des images ».

Ex 5 : inversion.

1. Dans le TD sur notebook, avez-vous bien compris l'opération à faire sur les pixels pour inverser les niveaux de gris ?
2. Qu'y-a-t-il à modifier dans le programme pour qu'il inverse les images en couleurs en format ppm ? Le tester.

Ex 6 : contraste et luminosité.

Créer un programme qui modifie le contraste, ou bien la luminosité d'une image. L'une de ces opérations correspond à multiplier ou diviser la valeur de chaque pixel par une constante, l'autre à ajouter ou soustraire à chaque pixel une constante. On pourra créer deux programmes, et on n'oubliera pas que les valeurs de gris ou de couleur ne peuvent pas être n'importe quoi, et sont des entiers !

Ex 7 : des niveaux de gris au noir et blanc.

Écrire un programme qui transforme une image initialement en niveaux de gris, en noir et blanc. On pourra créer un réglage afin de choisir le degré de noir (ou de blanc). En testant différentes valeurs du réglage, répondre à la question posée : 0 correspond-il à noir ou bien à blanc ? On peut également répondre à cette question en ouvrant l'image dans l'éditeur hexadécimal, et en comparant les données à l'image.

Les exercices suivants demandent un peu plus de réflexion, puisqu'on travaille sur plusieurs données consécutives.

Ex 8 : de la couleur aux niveaux de gris (à peine plus dur).

Écrire un programme qui transforme une image en couleurs en image en niveaux de gris. Pour cela, on fera simplement la moyenne des trois couleurs, cela correspondra à la valeur du gris.

Ex 9 : couches de couleur

Écrire un programme qui extraie une des couches de couleur, c'est à dire qu'on conserve une des couches intactes, et que les deux autres sont mises à 0.

Encore un peu plus difficile.

Travail préliminaire : écrire les données dans une matrice de taille hauteur de l'image × largeur de l'image. On rappelle que les données sont écrites dans le fichier à la suite les unes des autres, sans préciser que l'on passe d'une ligne de l'image à une autre. Il est bien plus facile de donner les dimensions de l'image au programme directement, plutôt que de les extraire automatiquement de l'en-tête (c'est à dire que vous affichez l'en tête et ré-entrez les dimensions avec un input).

Ex 8 : effet surprise.

Écrire un programme qui transforme un pixel en faisant la moyenne avec ses huit voisins (autour de lui). Si vous ne voyez pas l'effet obtenu, itérez le processus en reprenant à chaque fois l'image transformée comme image de départ.

Ex 9 : contour.

Écrire un programme qui transforme le pixel $P_{i,j}$ soit en noir, soit en blanc, en fonction de la valeur de $\left|P_{i+1,j} - P_{i-1,j}\right| + \left|P_{i,j+1} - P_{i,j-1}\right|$ par rapport à un seuil à choisir.

Ex 10 : réflexion.

Écrire un programme qui transforme l'image en son reflet (inversion droite gauche, ou bien haut/bas).

Ex 11 : vers l'image animée.

Prenons deux photos de suite d'une scène avec un objet fixe et un objet mobile et soustrayons les deux images, c'est-à-dire faisons la soustraction de chaque valeur du pixel de la deuxième image avec le pixel correspondant dans la première. Que s'est-il passé pour l'objet fixe ou mobile ? À quoi peut servir la soustraction d'images alors ?

Vous pouvez faire cet exercice soit en réfléchissant, soit en le programmant.