Vous savez déjà qu'un ordinateur permet de faire pleins de choses totalement différentes : écouter de la musique, lire des films/vidéos, afficher ou écrire du texte, retoucher des images, créer des vidéos, jouer à des jeux vidéos, etc. Pour être plus général, on devrait dire qu'un ordinateur manipule des informations, sous la forme de fichier texte, de vidéo, d'image, de morceau de musique, de niveau de jeux vidéos, etc. Dans ce qui suit, nous allons appeler ces informations par le terme **données**. On pourrait définir les ordinateurs comme des appareils qui manipulent des données et/ou qui traitent de l'information, mais force est de constater que cette définition, oh combien fréquente, n'est pas la bonne. Tous les appareils électroniques manipulent des données, même ceux qui ne sont pas des ordinateurs proprement dit : les exemples des décodeurs TNT et autres lecteurs de DVD sont là pour nous le rappeler. Même si la définition d'ordinateur est assez floue et que plusieurs définitions concurrentes existent, il est évident que les ordinateurs se distinguent des autres appareils électroniques programmables sur plusieurs points. Notamment, ils stockent leurs données d'une certaine manière (le codage numérique que nous allons aborder).

Le codage de l'information

Avant d'être traitée, une information doit être transformée en données exploitables par l'ordinateur, sans quoi il ne pourra pas en faire quoi que ce soit. Eh bien, sachez qu'elles sont stockées… avec des nombres. Toute donnée n'est qu'un ensemble de nombres structuré pour être compréhensible par l'ordinateur : on dit que les données sont codées par des nombres. Il suffit d'utiliser une machine à calculer pour manipuler ces nombres, et donc sur les données. Une simple machine à calculer devient une machine à traiter de l'information. Aussi bizarre que cela puisse paraitre, un ordinateur n'est qu'une sorte de grosse calculatrice hyper-performante. Mais comment faire la correspondance entre ces nombres et du son, du texte, ou toute autre forme d'information ? Et comment fait notre ordinateur pour stocker ces nombres et les manipuler ? Nous allons répondre à ces questions dans ce chapitre.

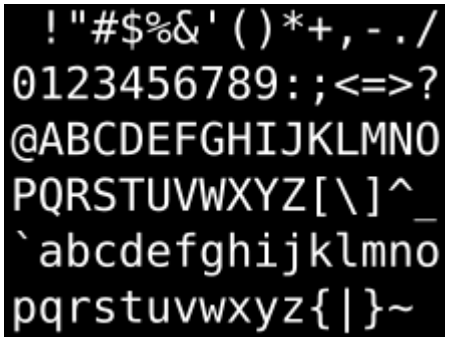
Toute information présente dans un ordinateur est décomposée en petites informations de base, chacune représentée par un nombre. Par exemple, le texte sera décomposé en caractères (des lettres, des chiffres, ou des symboles). Pareil pour les images, qui sont décomposées en pixels, eux-mêmes codés par un nombre. Même chose pour la vidéo, qui n'est rien d'autre qu'une suite d'images affichées à intervalles réguliers. La façon dont un morceau d'information (lettre ou pixel, par exemple) est représenté avec des nombres est définie par ce qu'on appelle un codage, parfois appelé improprement encodage. Ce codage va attribuer un nombre à chaque morceau d'information. Pour montrer à quoi peut ressembler un codage, on va prendre trois exemples : du texte, une image et du son.

Texte : standard ASCII

Pour coder un texte, il suffit de savoir coder une lettre ou tout autre symbole présent dans un texte normal (on parle de **caractères**). Pour coder chaque caractères avec un nombre, il existe plusieurs codages : l'ASCII, l'Unicode, etc. Le codage le plus ancien, l'ASCII, est intégralement défini par une table de correspondance entre une lettre et le nombre associé : la table ASCII. Ce standard ASCII utilise des nombres codés sur 7 bits, et peut donc coder 128 symboles différents. Les lettres sont stockées dans l'ordre alphabétique, pour simplifier la vie des utilisateurs : des nombres consécutifs correspondent à des lettres consécutives. L'ASCII ne code pas seulement des lettres, mais aussi d'autres symboles, dont certains ne sont même pas affichables ! Cela peut paraitre bizarre, mais s'explique facilement quand on connait les origines du standard. Ces caractères non-affichables servent pour les imprimantes, FAX et autres systèmes de télécopies. Pour faciliter la conception de ces machines, on a placé dans cette table ASCII des symboles qui n'étaient pas destinés à être affichés, mais dont le but était de donner un ordre à l'imprimante/machine à écrire… On trouve ainsi des symboles de retour à la

ligne, par exemple.

La table ASCII a cependant des limitations assez problématiques. Par exemple, vous remarquerez que les accents n'y sont pas, ce qui n'est pas étonnant quand on sait qu'il s'agit d'un standard américain. De même, impossible de coder un texte en grec ou en japonais : les idéogrammes et les lettres grecques ne sont pas dans la table ASCII. Pour combler ce manque, de nombreuses autres codages du texte sont apparus, le plus connu étant certainement l'**Unicode**. Pour plus de simplicité, l'Unicode est parfaitement compatible avec la table ASCII : les 128 premiers symboles de l'Unicode sont ceux de la table ASCII, et sont rangés dans le même ordre. Là où l'ASCII ne code que l'alphabet anglais, les codages actuels comme l'Unicode prennent en compte les caractères chinois, japonais, grecs, etc.



Caractères ASCII imprimables.

Image

Le même principe peut être appliqué aux images : l'image est décomposée en morceaux de même taille qu'on appelle des **pixels**. L'image est ainsi vue comme un rectangle de pixels, avec une largeur et une longueur. Le nombre de pixels en largeur et en longueur définit la résolution de l'image : par exemple, une image avec 800 pixels de longueur et 600 en largeur sera une image dont la résolution est de 800*600. Il va de soi que plus cette résolution est grande, plus l'image sera fine et précise. On peut d'ailleurs remarquer que les images en basse résolution ont souvent un aspect dit pixelisé, où les bords des objets sont en marche d'escaliers.

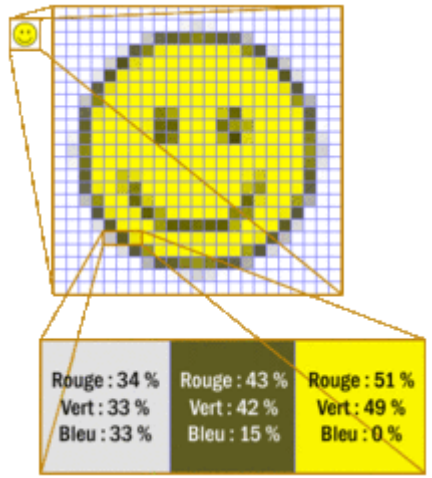


Image matricielle.

Chaque pixel a une couleur qui est codée par un ou plusieurs nombres entiers. D'ordinaire, la couleur d'un pixel est définie par un mélange des trois couleurs primaires rouge, vert et bleu. Par exemple, la couleur est composée à 50 % de rouge et à 50 % de vert. Pour coder la couleur d'un pixel, il suffit de coder chaque couleur primaire avec un nombre entier : un nombre pour le rouge, un autre pour le vert et un dernier pour le bleu. Ce codage est appelé le **codage RGB**. Mais il existe d'autres méthodes, qui codent un pixel non pas à partir des couleurs primaires, mais à partir d'autres espaces de couleur.

Pour stocker une image dans l'ordinateur, on a besoin de connaitre sa largeur, sa longueur et la couleur de chaque pixel. Une image peut donc être représentée dans un fichier par une suite d'entiers : un pour la largeur, un pour la longueur, et le reste pour les couleurs des pixels. Ces entiers sont stockés les uns à la suite des autres dans un fichier. Les pixels sont stockés ligne par ligne, en partant du haut, et chaque ligne est codée de gauche à droite. Les fichiers images actuels utilisent des techniques de codage plus élaborées, permettant notamment décrire une image en utilisant moins de nombres, ce qui prend moins de place dans l'ordinateur.

Son

Pour mémoriser du son, il suffit de mémoriser l'intensité sonore reçue par un microphone à intervalles réguliers. Cette intensité est codée par un nombre entier : si le son est fort, le nombre sera élevé, tandis qu'un son faible se verra attribuer un entier petit. Ces entiers seront rassemblés dans l'ordre de mesure, et stockés dans un fichier son, comme du wav, du PCM, etc. Généralement, ces fichiers sont compressés afin de prendre moins de place.

Les différents codages : analogique, numérique et binaire

Pour pouvoir traiter de l'information, la première étape est d'abord de coder celle-ci. On a vu dans le chapitre sur le binaire comment représenter des informations simples en utilisant le binaire. Mais ce codage, cette transformation d'information en nombre, peut être fait de plusieurs façons différentes. Dans les grandes lignes, on peut identifier deux grands types de codages.

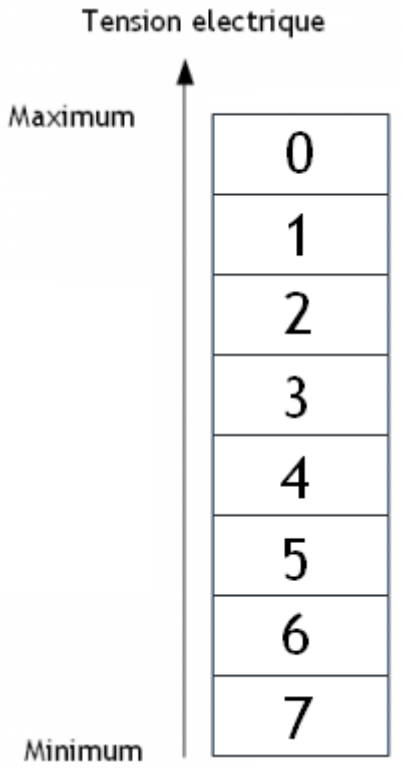
- Le **codage analogique** utilise des nombres réels : il code l'information avec des grandeurs physiques (des trucs qu'on peut mesurer par un nombre) comprises dans un intervalle. Par exemple, un thermostat analogique convertit la température en tension électrique pour la manipuler : une température de 0 degrés donne une tension de 0 Volts, une température de 20 degrés donne une tension de 5 Volts, une température de 40 degrés donnera du 10 Volts, etc. Un codage analogique a une précision théoriquement infinie : on peut par exemple utiliser toutes les valeurs entre 0 et 5 Volts pour coder une information, même des valeurs tordues comme 1, 2.2345646, ou pire… Un calculateur analogique (qui utilise le codage analogique) peut en théorie faire certains calculs avec une précision théorique très fine, impossible à atteindre avec un calculateur numérique : c'est le cas pour les dérivées, intégrations et autres calculs similaires.
- Le **codage numérique** va coder des informations en utilisant des nombres entiers, représentés par des suites de chiffres. Pour donner une définition plus générale, le codage numérique utilise qu'un nombre fini de valeurs, contrairement au codage analogique. Cela donnera des valeurs du style : 0, 0.12, 0.24, 0.36, 0.48… jusqu'à 2. Pour les calculateurs numériques, les nombres manipulés sont codés par des suites de chiffres. Les appareils numériques peuvent être vus comme des calculatrices particulières. Vu que toutes les données sont des nombres, toute manipulation de données peut se réaliser avec une suite de calculs bien précis. Et c'est ce que font les circuits numériques : ils exécutent une suite de calculs de manière totalement autonome. Mais ils se démarquent des calculatrices usuelles sur deux points. Premièrement, ils sont hyper-performants, bien plus puissants que les calculatrices de poche. Ils peuvent faire un grand nombre de calculs par seconde, plusieurs centaines de millions. Deuxièmement, ils sont conçus pour effectuer des calculs bien précis (là où une calculette fait les calculs qu'on lui demande).

Le support physique des chiffres

Mais peu importe le codage utilisé, celui-ci a besoin d'un support physique, d'une grandeur physique quelconque : le codage analogique a besoin de quelque chose pour mémoriser un nombre, tandis que le codage numérique a besoin de quelque chose pour représenter un chiffre. Et pour être franc, on peut utiliser tout et n'importe quoi. Par exemple, certains calculateurs assez anciens étaient des calculateurs pneumatiques, et utilisaient la pression de l'air pour représenter des chiffres ou nombres : soit le nombre encodé était proportionnel à la pression (codage analogique), soit il existait divers intervalles de pression pour chaque chiffre (codage numérique). De nos jours, ce stockage se fait soit par l'aimantation d'un support magnétique, soit par un support électronique. Les supports magnétiques sont réservés aux disques durs magnétiques, destinés à être remplacés par des disques durs entièrement électroniques (les fameux Solid State Drives, que nous verrons dans quelques chapitres).

Pour les supports de stockage électroniques, très courants dans nos ordinateurs, le support en question est une **tension électrique**. Ces tensions sont ensuite manipulées par des composants électriques/électroniques plus ou moins sophistiqués : résistances, condensateurs, bobines, amplificateurs opérationnels, diodes, transistors, etc. Certains d'entre eux ont besoin d'être alimentés en énergie. Pour cela, chaque circuit est relié à une tension qui l'alimente en énergie : la **tension d'alimentation**. Après tout, si un circuit doit coder des bits valant 1, il faudra bien qu'il trouve de quoi fournir une tension de 2, 3, 5 volts : la tension codant notre 1 ne sort pas de nulle part ! De même, on a besoin d'une tension de référence valant zéro volt, qu'on appelle la **masse**, qui sert pour le zéro. Dans tous les circuits électroniques (et pas seulement les ordinateurs), cette

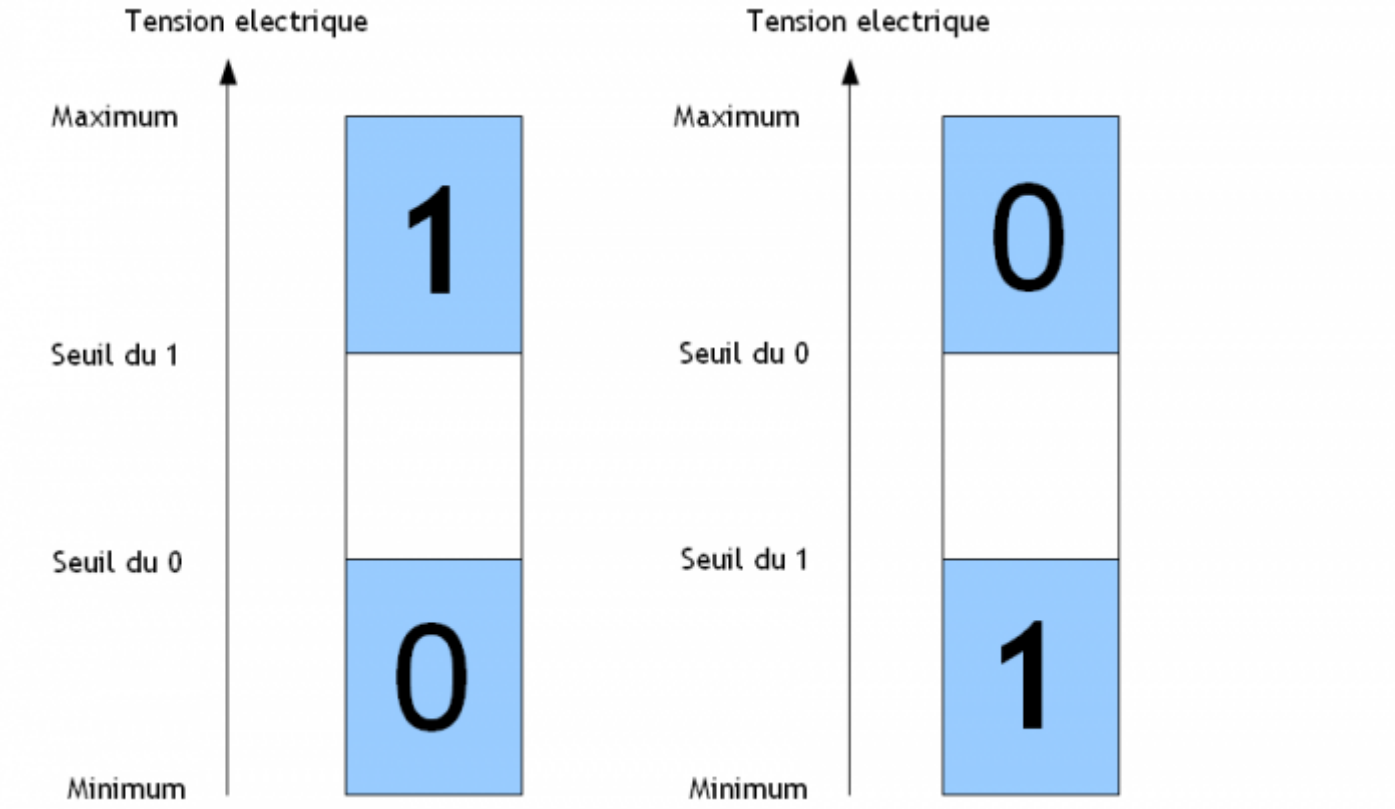
tension d'alimentation varie généralement entre 0 et 5 volts. Mais de plus en plus, on tend à utiliser des valeurs de plus en plus basses, histoire d'économiser un peu d'énergie. Et oui, car plus un circuit utilise une tension élevée, plus il consomme d'énergie et plus il chauffe. Pour un processeur, il est rare que les modèles récents utilisent une tension supérieure à 2 volts : la moyenne tournant autour de 1-1.5 volts. Même chose pour les mémoires : la tension d'alimentation de celle-ci diminue au court du temps. Pour donner des exemples, une mémoire DDR a une tension d'alimentation qui tourne autour de 2,5 volts, les mémoires DDR2 ont une tension d'alimentation qui tombe à 1,8 volts, et les mémoires DDR3 ont une tension d'alimentation qui tombe à 1,5 volts. C'est très peu : les composants qui manipulent ces tensions doivent être très précis.



Le choix d'un codage et de sa base

De nos jours, tous les appareils électroniques et ordinateurs utilisent un codage numérique ! C'est dû au fait que les calculateurs analogiques sont plus sensibles aux perturbations électromagnétiques (on dit aussi ils ont une faible immunité au bruit). Explication : un signal analogique peut facilement subir des perturbations qui vont changer sa valeur, modifiant directement la valeur des nombres stockés/manipulés. Avec signal numérique, les perturbations ou parasites vont moins perturber le signal numérique. La raison est qu'une variation de tension qui reste dans un intervalle représentant un chiffre ne changera pas sa valeur. Il faut que la variation de tension fasse sortir la tension de l'intervalle pour changer le chiffre.

De nos jours, les ordinateurs n'utilisent que deux chiffres, 0 et 1 : on dit qu'ils comptent en binaire. On verra dans le chapitre suivant comment coder des nombres avec des bits, ce qui est relativement simple. Pour le moment, nous allons justifier ce choix de n'utiliser que des bits et pas les chiffres décimaux (de 0 à 9). Avec une tension électrique, il y a diverses méthodes pour coder un bit : codage Manchester, NRZ, etc. Ces diverses méthodes ont chacune leurs avantages et leurs défauts. Autant trancher dans le vif tout de suite : la quasi-intégralité des circuits de notre ordinateur se basent sur le **codage NRZ**. Pour coder un 0 ou 1 en NRZ, la tension est en-dessous d'un seuil donné pour un 0. Et il existe un autre seuil au-dessus duquel la tension représente un 1. Du moins, c'est ainsi dans la majorité des cas : il arrive que ce soit l'inverse sur certains circuits électroniques : en-dessous d'un certain seuil, c'est un 1 et si c'est au-dessus d'un autre seuil c'est 0. Tout ce qu'il faut retenir, c'est qu'il y a un intervalle pour le 0 et un autre pour le 1. En dehors de ces intervalles, on considère que le circuit est trop imprécis pour pouvoir conclure sur la valeur de la tension : on ne sait pas trop si c'est un 1 ou un 0. Il y a deux seuils, car les circuits qui manipulent des tensions n'ont pas une précision parfaite, et qu'une petite perturbation électrique pourrait alors transformer un 0 en 1. Pour limiter la casse, on préfère séparer ces deux seuils par une sorte de marge de sécurité.



L'avantage du binaire par rapport aux autres codages est qu'il permet de mieux résister aux perturbations électromagnétiques mentionnées dans le chapitre précédent. À tension d'alimentation égale, les intervalles de chaque chiffre sont plus petits pour un codage décimal : toute perturbation de la tension aura plus de chances de changer un chiffre. Mais avec des intervalles plus grands, un parasite aura nettement moins de chance de modifier la valeur du chiffre codé ainsi. La résistance aux perturbations électromagnétiques est donc meilleure avec seulement deux intervalles.