ministère éducation nationale

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

éduscol

Ressources pour le lycée général et technologique

Ressources pour le cycle terminal général et technologique

Informatique et Sciences du Numérique

Qu'est-ce qu'une image numérique matricielle ?

Juin 2012

Présentation / Qu'est-ce qu'une image numérique matricielle ?

1 / Thème abordé

1.1 Problématique et accroche

Ce parcours pédagogique propose une initiation à la notion d'image numérique (matricielle) vue comme un ensemble de points, avec pour but de mettre en situation la partie du programme « Représentation de l'information » en lui donnant un cadre concret et motivant.

Après une sensibilisation basée sur des images médicales ou météorologiques, la partie pratique propose de découvrir concrètement la structure discrète de l'image et des couleurs, puis d'identifier les informations nécessaires à la conception et au codage d'une image matricielle. La manipulation effective des images est enfin proposée, d'abord pixel par pixel, puis par une approche de programmation.

1.2 Frontières de l'étude et prolongements possibles

Ce scénario dont l'objectif est clairement délimité (définir la notion d'image) peut néanmoins prendre de multiples formes et s'adapter à un public hétérogène dont les besoins sont différents : il est important que chaque élève se sente concerné.

On pourra par exemple, pour des élèves débutants, faire un détour par l'image binaire (en noir et blanc strict) en proposant des activités détachées du support informatique et de tout langage, tandis que des élèves plus performants ou rapides pourront, de leur côté, aborder le codage d'une image en couleur.

Suivant la même idée, il peut être nécessaire de construire progressivement la notion de codage binaire en commençant par des formats d'image très simples (avec des en-têtes réduits et dont le codage est en ASCII) puis de plus en plus complexes.

De nombreux prolongements sont possibles :

- choisir un format d'image selon des critères précis : rapidité d'affichage, taille de fichier, esthétique …
- mettre en place des stratégies pour limiter la taille des fichiers en tenant compte de leurs spécificités (problématique de la compression avec ou sans perte).
- concevoir des filtres mettant en jeu une démarche algorithmique.
- coder un algorithme de manipulation d'images dans un langage de programmation.

2 / Objectifs pédagogiques

2.1 Disciplines impliquées

Les mathématiques interviennent dans la mise en place du codage, notamment binaire. On pourra associer les SVT en travaillant sur des images satellitales ou issues d'IRM : de nombreuses banques d'images libres sont désormais disponibles.

2.2 Prérequis

Quelques notions élémentaires sur la représentation de données sous forme numérique ; l'étude peut donc être lancée dès le début de l'année.

2.3 Éléments du programme

Contenus

Représentation de l'information : représentation binaire, numérisation, formats.

Capacités

- Manipuler à l'aide d'opérations élémentaires les trois unités de base : bit, octet, mot.
- Numériser une image ou un son sous forme d'un tableau de valeurs numériques.
- Identifier quelques formats de documents, d'images, de données sonores.

Compétences mises en jeu

Décrire et expliquer une situation :

- faire preuve de sens critique, vérifier la validité des hypothèses.

Concevoir et réaliser une solution informatique en réponse à un problème :

- imaginer des solutions répondant à l'expression d'un besoin.
- trier et analyser des informations.
- travailler en autonomie, s'organiser dans une exploration empirique.
- faire preuve de créativité dans la mise en place et l'analyse de jeux d'essais.

3 / Modalités de mise en œuvre

3.1 Type et durée de l'animation

L'introduction (problématique et accroche, une demi-heure environ) peut être faite en classe entière, éventuellement suivie d'un travail « sur papier » (voir annexe) qui permet d'introduire la structure matricielle de l'image.

La partie expérimentale de découverte se déroule dans une salle spécialisée avec au maximum un poste pour deux élèves, une ou deux heures selon le niveau d'approfondissement et le nombre de formats étudiés.

Le temps de synthèse peut se faire en classe entière (1 heure au plus), et peut s'enchaîner avec un débat en classe entière pour aborder l'aspect sociétal.

3.2 Mini-projets

De nombreuses pistes s'ouvrent ensuite pour envisager quelques mini-projets :

- des algorithmes de recherche dans l'image : comment rechercher par exemple des pixels d'une couleur donnée (ou dans une plage de couleurs) ?
- des algorithmes simples de compression : sont-il efficaces sur toutes les images ?
- des algorithmes de transformation de l'image : ainsi, le passage en négatif d'une image.

Ces mini-projets seront aussi l'occasion de travailler avec un langage de programmation et permettront de mettre en place les bases méthodologiques requises pour le développement de projets plus conséquents.

3.3 Recherches documentaires

Lors de la mise en place du mini-projet, des recherches seront à conduire sur les formats choisis, le langage envisagé, les algorithmes utilisés.

3.4 Productions d'élèves

À l'issue de la partie pratique, l'élève devra avoir codé une image simple en bitmap (en niveaux de gris). On pourra aussi demander un compte rendu de la démarche expérimentale mise en place pour comprendre le codage : par exemple un jeu d'essais et son analyse.

3.5 Évaluation

Comme il s'agit d'une situation introductive, on peut s'en servir pour évaluer le niveau initial de maîtrise des compétences indiquées ci-dessus, ce qui pourra s'appuyer sur des productions.

4 / Outils

- Éditeur hexadécimal
- Logiciel de visualisation d'images (XnView, GwenView etc.)
- Logiciel de traitement d'images : Photofiltre, The Gimp (http://www.gimp.org), etc.

5 / Auteur

Philippe Jonin, professeur de Mathématiques, académie de Nantes

Ressource / Qu'est-ce qu'une image numérique matricielle ?

1 / Problématique

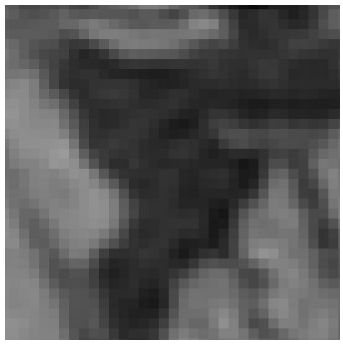
On se propose de travailler dans ce scénario pédagogique la notion d'image numérique (matricielle)[1] vue comme un ensemble fini de points colorés, avec pour but de mettre en situation la partie du programme « représentation de l'information ». On rappelle d'abord qu'un ordinateur ne manipule que des valeurs numériques représentées sous forme binaire[2]. Comment fait-on alors pour coder une image avec des nombres ?

Une idée simple est de la décomposer en parties élémentaires, des points (que l'on appellera des pixels) et de coder l'état de ces points c'est-à-dire leur couleur. Pour simplifier encore, dans ce document nous ne nous intéresserons qu'aux images en niveaux de gris, pour lesquelles une « couleur » n'est autre qu'un niveau de gris compris entre le noir (absence de lumière) et le blanc (luminosité maximale permise par le support physique).

Un petit exemple permet de mieux saisir ces notions.

Image d'une coupe sagittale de cerveau humain obtenue en IRM anatomique[3].

La même image avec un zoom x 4
On commence à distinguer les pixels.



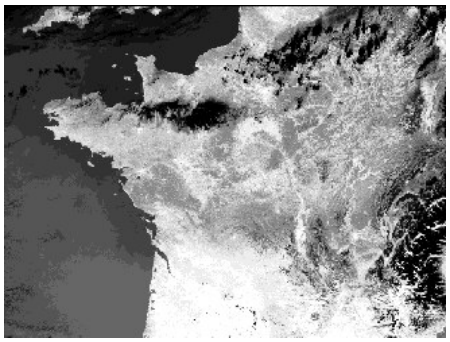
Zoom x 12 : un tableau de pixels...

L'image prend ainsi une structure discrète organisée comme un tableau avec des lignes et des colonnes contenant des cases, dont le contenu est une information liée à la couleur du pixel correspondant.

Le but de l'étude apparaît alors : il s'agit de détailler cette structure d'une image, d'identifier les informations nécessaires à sa conception (taille de l'image, nombre de couleurs) et de réfléchir à la façon dont on peut coder ces couleurs puis ranger tout cela dans un fichier.

Il est utile de mettre rapidement les élèves en activité dans une démarche de découverte et d'expérimentation : on ne se limitera évidemment pas à un cours théorique sur les images et leur codage.

2 / Situation d'accroche : un traitement d'image satellitale en SVT



Dans l'image ci-contre (thermographie NOAA réalisée le 30 mars 1991[4]), le niveau de température est codé par une intensité de gris.
Un gris foncé code-t-il une température élevée ou basse ?
Quel est le type de gris des plaines du sud de la France ? Du massif des Alpes ?
Les cirrus et cumulus ont une température très basses : peut-on les repérer ? Pourrait-on sur ce type d'image repérer automatiquement (avec un programme) les perturbations atmosphériques ou les massifs montagneux ?

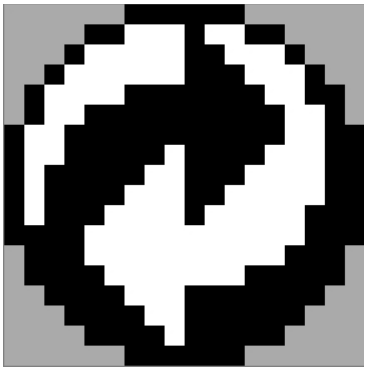
1 Par opposition aux « images vectorielles », définies par un ensemble de tracés de lignes, courbes, et de remplissages de zones.
2 A proprement parler, l'ordinateur ne manipule que des zéros et des uns, lesquels s'assemblent pour former des nombres représentés suivant le système binaire.
3 Image © CNRS Photothèque / DURAND, Emmanuel
4 Image extraite de *Images satellitales NOAA et connaissance de l'environnement terrestre*., © Ministère de l'Éducation nationale, Météo–France, 1994.

3 / Qu'est-ce qu'une image numérique matricielle ?

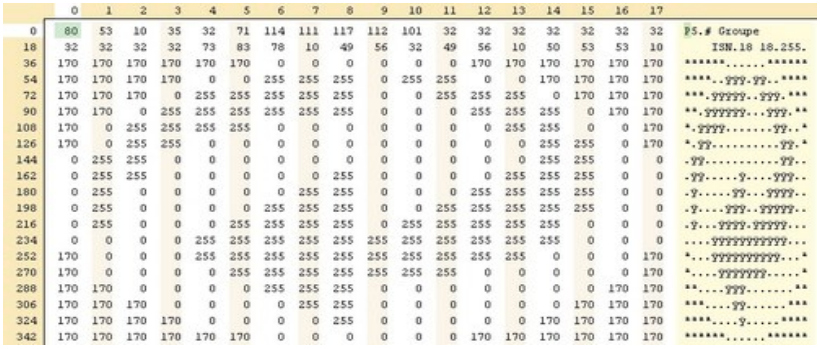
3.1 Démarche de découverte expérimentale du codage

Il existe de nombreux formats d'images avec des niveaux de complexité très variables : on ne cherche pas ici à étudier un format précis, qui deviendra inévitablement obsolète, mais à mettre en avant des notions pérennes comme les caractéristiques d'une image (longueur, largeur, couleur, profondeur, taille), la notion de structure discrète et la notion de codage numérique.

À partir d'une image numérique très simple (une icône[5] de taille 18×18) en niveaux de gris, dont le format aura été soigneusement choisi, on peut visualiser et modifier son code : par exemple en utilisant un tableur, un éditeur de texte ou, pour des formats plus élaborés, un éditeur hexadécimal. Il est ainsi possible d'afficher simultanément dans deux fenêtres juxtaposées l'image (avec un logiciel de visualisation) et son code. On peut alors agir sur l'une et observer le résultat sur l'autre. On peut même travailler dans les deux sens.



L'image initiale (agrandie) Elle est au format PGM (P5)



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
0	80	53	10	35	32	71	114	111	117	112	101	32	32	32	32	32	32	32	P5.# Groupe
18	32	32	32	32	73	83	78	10	49	56	32	49	56	10	50	53	53	10	ISN.18 18.255.
36	170	170	170	170	170	170	0	0	0	0	0	0	170	170	170	170	170	170	******.....******
54	170	170	170	170	0	0	255	255	255	0	255	255	0	0	170	170	170	170	****..yyy.yy..****
72	170	170	170	0	255	255	255	255	255	0	0	255	255	255	0	170	170	170	***.yyyyy..yyy.***
90	170	170	0	255	255	255	255	255	255	0	0	0	255	255	255	0	170	170	**.yyyyyy...yyy.**
108	170	0	255	255	255	255	0	0	0	0	0	0	0	255	255	0	0	170	*.yyyy.....yy..*
126	170	0	255	255	0	0	0	0	0	0	0	0	0	0	255	255	0	170	*.yy.....yy..*
144	0	255	255	0	0	0	0	0	0	0	0	0	0	0	255	255	0	0	-yy.....yy..
162	0	255	255	0	0	0	0	0	255	0	0	0	0	255	255	255	0	0	-yy.....y....yyy..
180	0	255	0	0	0	0	0	255	255	0	0	0	255	255	255	255	0	0	-y.....yy...yyyy..
198	0	255	0	0	0	0	255	255	255	0	0	255	255	255	255	255	0	0	-y....yyy..yyyyy..
216	0	255	0	0	0	255	255	255	255	0	255	255	255	255	255	0	0	0	-y...yyyy.yyyyy...
234	0	0	0	0	255	255	255	255	255	255	255	255	255	255	255	0	0	0	....yyyyyyyyyy...
252	170	0	0	0	255	255	255	255	255	255	255	255	255	255	0	0	0	170	*...yyyyyyyyyy...*
270	170	0	0	0	0	255	255	255	255	255	255	255	0	0	0	0	0	170	*....yyyyyyy.....*
288	170	170	0	0	0	0	255	255	255	0	0	0	0	0	0	0	170	170	**....yyy.....**
306	170	170	170	0	0	0	0	255	255	0	0	0	0	0	0	170	170	170	***....yy.....***
324	170	170	170	170	0	0	0	0	255	0	0	0	0	0	170	170	170	170	****....y.....****
342	170	170	170	170	170	170	0	0	0	0	0	0	170	170	170	170	170	170	******.....******

Sa visualisation avec un éditeur hexadécimal[6] : on retrouve presque l'image. La recherche du codage est ainsi grandement facilitée. Il est par exemple clair que le noir est codé par 0 et le blanc par 255 ; le « gris moyen » des bords de l'image est ici codé par 170.

Cette démarche devrait permettre d'induire des constats et des questions du type :

- Une image est codée par un tableau de valeurs. Combien de cases contient-il ?
- Peut-on relier ce nombre de cases à la taille de l'image ?
- Quelles informations contient alors une case ?
- Où trouve-t-on les caractéristiques de l'image (dimensions, nombre de couleurs mais aussi nom de l'auteur, date de création etc.) ? Il s'agit alors de construire la notion d'en-tête.
- Si on modifie une case du tableau, la machine pourra-t-elle interpréter toute modification ? (y compris dans l'en-tête ?). Y a-t-il un contrôle ou un système de correction d'erreurs ?
- Comment évolue la taille du codage en fonction de la taille de l'image ? On pourra conduire, pour compléter cette interrogation, une petite étude en puisant d'autres exemples dans une banque d'images. L'idée de la compression va peu à peu s'imposer !

Le format du fichier sera une variable didactique à considérer soigneusement : son choix permettra de s'adapter au public et au niveau d'approfondissement visé par le professeur. On prendra en compte les formats déjà connus et utilisés par les élèves et on comparera leurs caractéristiques.
Il s'agit ainsi de rendre les élèves à même de choisir un format en mesurant toutes les implications de ce choix :

- Quelle taille occupera-t-il ?
- Si on utilise une compression ,sera-t-elle destructive ?
- L'image sera-t-elle fidèle (couleur, résolution) ?
- Quels logiciels peuvent le manipuler ?
- Sa lecture utilise-t-elle des ressources du processeur ?

3.2 Démarche de modification et de création

5 Le fichier correspondant est dans l'archive fournie en annexe de cette ressource.

6 Dans un but de simplification de la première approche, l'éditeur hexadécimal a été ici réglé en mode « décimal » ; il convient néanmoins de basculer assez rapidement en mode « hexa » voire en binaire selon les besoins.

On propose pour terminer cette partie de coder « à la main » une image à trois ou quatre niveaux de gris (en incluant le blanc et le noir et un « gris moyen ») : construire l'en-tête, puis le codage lui même.

Pour des élèves débutants on peut proposer dans un premier temps des activités détachées de tout support matériel et travailler sur des activités papier de codage (voir une proposition en annexe).

3.3 Synthétiser la notion d'image

L'idée centrale à dégager est la suivante :

Une image bitmap (matricielle) peut être assimilée à un tableau de pixels de dimensions données (longueur et largeur).

Coder l'image nécessite d'envisager les trois aspects suivants :

- Comment parcourir les pixels : de haut en bas, de gauche à droite ? Par ligne ou par colonne ? La réponse sera variable selon les formats. Les formats Targa ou le PGM[7] procèdent ligne par ligne, de haut en bas et de gauche à droite au sein de chaque ligne ; le premier pixel est donc situé au coin supérieur gauche. Le format BMP, que l'on pourra évoquer en prolongement, travaille sur le même principe mais en partant du coin inférieur gauche et en remontant. Les algorithmes de balayage seront donc différents selon le format.
- Comment coder la couleur d'un pixel ? On restreint ici l'étude à des niveaux de gris ; mais de combien de niveaux distincts a-t-on besoin ? Une nouvelle caractéristique de l'image s'impose : la profondeur de couleur d'une image. On peut alors énoncer le principe suivant :
  - Un pixel noir est codé par la valeur 0, ce qui présente un caractère intuitif : il n'y a aucune lumière.
  - Un pixel blanc est codé par la valeur maximale utilisée pour coder le nombre de gris.
  - Chaque niveau de gris est codé par une valeur entre ces deux extrêmes, proportionnellement à son intensité ; pour suivre les dénominations usuelles en infographie, on a une échelle de gris désignés par un pourcentage, un « gris 10% » étant proche du blanc, un « gris 90% » étant presque noir[8].
  - Il ne reste plus qu'à coder cette valeur, soit en ASCII (format PGM de type P2), soit en binaire (format PGM de type P5 ou Targa), sur un bit, un octet voire plusieurs octets pour des formats dédiés à la photographie professionnelle. Une opportunité s'ouvre alors de faire une brève leçon sur le codage binaire.
- Comment structurer les fichiers contenant des images en vue d'une lecture informative ? Il faut assurément prévoir une zone pour écrire les caractéristiques de l'image : c'est l'en-tête.

Les éléments de cours qui précèdent ne seront pas donnés aux élèves de façon « frontale » mais seront d'abord abordés au travers d'activités expérimentales de découverte puis synthétisés en partant des compte-rendus des élèves.

3.4 Les aspects sociétaux

La généralisation des images numériques dans les ordinateurs, téléphones et autres affichages change notre rapport aux images en accentuant le contraste entre numérique et analogique ; même si la différence est peu perceptible en raison de la qualité de ces images et de leur rendu, elle reste fondamentale et influe sur le regard de chacun (voir en annexe l'expérience sur le tableau de Dali).

L'omniprésence des images dans la société actuelle pose aussi de nombreuses questions, juridiques notamment.

- Modifie-t-on la nature d'une image quand on la numérise ?
- Quelles possibilités ouvre le codage numérique ? Par exemple, compresser une image, l'analyser avec une démarche algorithmique, la marquer (filigrane), la conserver, la classer …
- Des banques d'images se construisent sur l'internet : peut-on y puiser sans restriction ? Modifier ou s'approprier ces images, voire les vendre ? On pourra, comme exemple pratique, étudier une charte d'utilisation comme celle proposée par la communauté Flickr.
- Qu'en est-il du droit des images (droit d'auteur, droit patrimonial, etc.) et du « droit à l'image » (cf. article 9 du code civil) ?

7 Portable Graymap, spécification faisant partie d'une gamme de formats d'image nommée PPM ou NetPBM.
8 Cette terminologie provient de la synthèse soustractive des couleurs.

4 / Références

- Le format PGM : http://fr.wikipedia.org/wiki/Portable_pixmap
- Le format Targa :
  http://www.iut-arles.up.univ-mrs.fr/eremy/Ens/Info2.RepCod/Formats/TGA.pdf
  ou bien :
  http://www.iut-arles.up.univ-mrs.fr/eremy/Ens/Info2.RepCod/Formats/tgaffs.pdf
- Le format BMP :
  http://www.fileformat.info/format/bmp/egff.htm
  http://fr.wikipedia.org/wiki/Windows_bitmap
- Pour aller plus loin, les images HDR :
  http://fr.wikipedia.org/wiki/Imagerie_grande_gamme_dynamique
  http://qtpfsgui.sourceforge.net
- Des activités informatiques sans ordinateur ni langage :
  http://csunplugged.org/image-representation
- Quelques banques d'images :
  - Une banque d'images médicales : http://www.bmlweb.org/image.html
  - Une banque d'image satellitales : http://eo.belspo.be/directory/SensorDetail.aspx
  - Satellites NOAA :
    http://eduscol.education.fr/orbito/system/noaa/noaa3.htm
    http://pedagogie.ac-amiens.fr/svt/info/analyses/titus/noaa/noaa.html#debut
  - Satellite Meteosat :
    http://www.eduspace.esa.int/eduspace/subtopic/default.asp?document=298&language=fr
- Images et SVT :
  - Utilisation des données NOAA-AVHRR dans l'étude de la brise thermique :
    http://cybergeo.revues.org/3132
  - Utilisation d'images IRM en SVT :
    http://acces.inrp.fr/acces/ressources/neurosciences/demarches-pedagogiques/propositions-de-demarches-pedagogiques-pour-la-classe-de-1ere-s
- Les aspects sociétaux :
  - Le portail « Internet responsable » du Ministère de l'éducation nationale :
    http://eduscol.education.fr/internet-responsable
  - Une autoformation bien faite sur le droit d'auteur et le droit à l'image (École des Mines de Nantes) :
    http://imedia.emn.fr/droits/co/droit_web.html

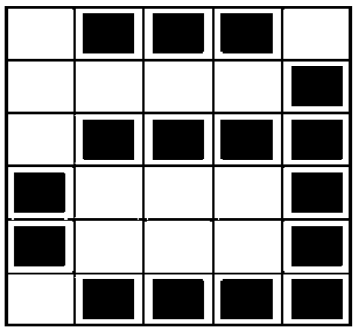
Annexes – d'autres activités

1 / Des activités détachées du support technique et de tout langage

Il est possible de construire des activités détachées de tout support « informatique » (c'est-à-dire, sans ordinateur). Elles permettront par exemple une entrée progressive pour des élèves peu sensibilisés à ces problématiques.

Exemple 1 : étude du tableau de Salvador Dali « *Gala regardant la mer Méditerranée qui à vingt mètres se transforme en portrait d'Abraham Lincoln (Hommage à Rothko), 1976* » (théâtre-musée Dali, Figueres, Catalogne, Espagne)[9]. On montre une reproduction du tableau (vidéoprojecteur) en jouant sur le zoom dans les deux sens ; la vision à mi-distance montre clairement une composition en blocs colorés ou pixels., tandis que la vision lointaine « combine » les pixels pour créer une « image » qui évoque la tête d'Abraham Lincoln (la vision rapprochée donne une autre lecture qui n'a plus de rapport avec les pixels).

Exemple 2 : une fois que le codage matriciel d'une image est mis en place, on peut aborder la compression avec un système ancien mais simple, nommé RLE (*Run Length Encoding* ou codage par longueur des plages) et abondamment utilisé dans les transmissions par fax (aujourd'hui quasiment disparues). L'image ci-contre (qui évoque une lettre « a ») est codée par les nombres figurant à droite. Il s'agit de découvrir le code !



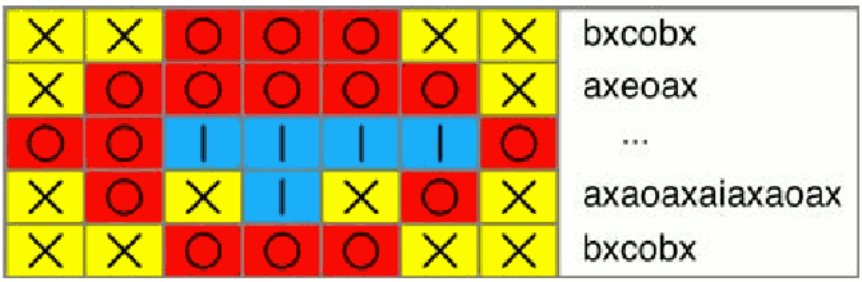
1, 3, 1
4, 1
1, 4
0, 1, 3, 1
0, 1, 3, 1
1, 4

Cette activité est présentée en détail dans les documents et pages indiquées ci-dessous :

csunplugged.org/image-representation
csunplugged.org/sites/default/files/activity_pdfs_other/02_fr_Repr%C3%A9sentation_d_une_image.pdf

Exemple 3 : la même idée, mais en couleurs, posée dans le cadre du concours du Castor informatique 2010[10].

L'image multicolore a été codée par un programme. À droite, tu peux voir le code composé de suites de lettres.



Malheureusement, le code de la troisième ligne a été perdu.

Quelle série de signes est le bon code pour la troisième ligne perdue ?

A. aobobicio
B. bocibo
C. bodiao
D. oociaio

Remarque : les activités présentées ci-dessus introduisent à la thématique de la compression de données sans perte d'une manière qui peut sembler peu efficace mais ouvre la voie à d'autres démarches bien plus efficientes (comme le codage de Huffman) et aussi basées sur l'observation fondamentale suivante : comme les données sont manifestement redondantes, on peut s'appuyer sur cette redondance pour développer un codage adapté à elles.

9 La reproduction de l'œuvre ne peut être incluse ici pour des raisons de droits mais on la trouve facilement sur le web. Le blog suivant présente l'œuvre de manière intéressante : citizenzoo.wordpress.com/2011/01/30/dali-gala-et-lincoln

10 Voir ici : http://castor-informatique.fr/plaquette.pdf

2 / Choisir et s'approprier un outil : l'éditeur hexadécimal

L'usage de l'éditeur hexadécimal est typique d'une problématique apparaissant dans les projets : c'est la recherche de la réponse à l'expression d'un besoin (ici, logiciel) ; on peut structurer la démarche comme suit :

1. identifier le besoin, le formaliser (=un logiciel permettant de manipuler les octets d'un fichier)
2. chercher quels logiciels peuvent éventuellement traiter ce besoin
3. distinguer, dans l'offre, les logiciels libres, non-libres mais gratuits et payants
4. télécharger le logiciel sur le site de l'éditeur (et surtout pas ailleurs)
5. installer le logiciel
6. essayer le logiciel
7. comparer les fonctionnalités au besoin initial
8. évaluer la qualité de la réponse au besoin ainsi que la qualité de l'ergonomie
9. si le bilan est positif, conserver l'outil et se l'approprier
10. si le bilan est négatif ou moyen, désinstaller complètement l'outil
11. [cas des logiciels libres] si après quelque temps l'outil rend de grands services, contribuer à l'effort collectif en participant à la documentation et au repérage des défauts (bugs).

Concrètement, on peut partir d'un tableau synoptique comme celui-ci :

en.wikipedia.org/wiki/Comparison_of_hex_editors

Chacun des logiciels présentés a des points forts et des faiblesses. On recommande particulièrement :

- FrHed (Windows seulement) : frhed.sourceforge.net
- HxD (Windows seulement) : mh-nexus.de/en/hxd
- wxHexEditor (multiplateformes) : http://www.wxhexeditor.org
- XVI32 (Windows seulement) : www.chmaas.handshake.de/delphi/freeware/xvi32/xvi32.htm
- Ghex : (Linux) live.gnome.org/Ghex
- Hexplorer (Windows seulement) : hexplorer.sourceforge.net
- Okteta (Linux) : utils.kde.org/projects/okteta

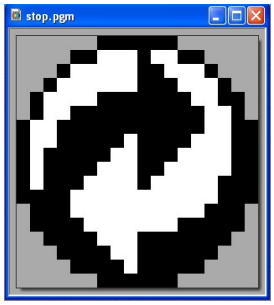
Certains de ces logiciels permettent d'expliciter le lien entre les systèmes binaire et hexadécimal.

3 / Mettre en place une démarche expérimentale de découverte

L'idée est d'afficher simultanément deux fenêtres : l'une contenant une visualisation de l'image et l'autre son codage.

On peut agir sur l'une des fenêtres et observer les changements sur l'autre (quitte à rafraîchir si besoin est).

Exemple 1 : une image au format PGM de type P2.

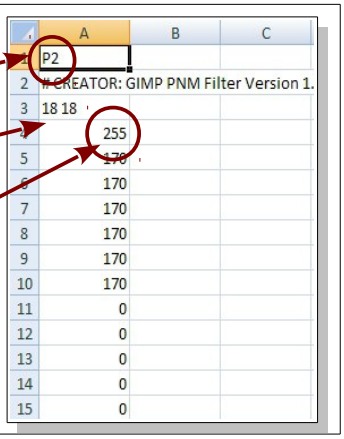


On a ouvert l'image avec PhotoFiltre (en mode RVB pour pouvoir la modifier).

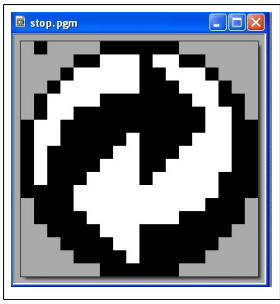
Le code de l'image est ici visualisé avec un tableur (on ouvre l'image comme un fichier CSV).

On observe en début de fichier l'en-tête : comprenant le type de fichier (P2), un commentaire indiquant le logiciel qui a permis de la construire (Gimp), sa taille (18x18) et le nombre maximal servant à coder les gris (on travaille ici avec 256 niveaux de gris).

Le codage suit immédiatement : 170, 170, …



Dans une image PGM de type P2, le niveau de gris est codé en ASCII : un tableur ou un éditeur de texte peu donc l'afficher de façon transparente. Ici l'image commence par des pixels (6 exactement...) dont le niveau de gris est 170.



Agissons sur le code :
en remplaçant le deuxième 170 par un 0 (et en sauvegardant), nous observons le résultat sur l'image (qu'il faut éventuellement rafraîchir) :
un pixel noir apparaît en haut à gauche.
L'image est donc codée de haut en bas et de gauche à droite.

	A	B	C
1	P2		
2	# CREATOR: GIMP PNM Filter Version 1.1		
3	18 18		
4	255		
5	170		
6	0		
7	170		
8	170		

Affichons maintenant le code de cette même image avec un éditeur hexadécimal (ci-dessous).

Pour faciliter la lecture on affiche les données au format décimal.

L'affichage est scindé en deux parties : à gauche on visualise les octets (qui apparaissent comme des cases) et à droite la correspondance au format texte (on retrouve l'affichage du tableur).

Le codage dans l'en-tête du type de l'image, ici P2, occupe donc deux octets dont le premier contient 80, le code ASCII de P et le second 50, le code ASCII de 2.

	00	01	02	03	04	05	06	07	
00000000	80	50	13	10	35	32	67	82	P2..# CR
00000008	69	65	84	79	82	58	32	71	EATOR: G
00000010	73	77	80	32	80	78	77	32	IMP PNM
00000018	70	105	108	116	101	114	32	86	Filter V
00000020	101	114	115	105	111	110	32	49	ersion 1
00000028	46	49	13	10	49	56	32	49	.1..18 1
00000030	56	13	10	50	53	53	13	10	8..255..
00000038	49	55	48	13	10	48	13	10	170..0..
00000040	49	55	48	13	10	49	55	48	170..170

On ne retrouve plus notre 170 : il est codé avec trois octets : 49; 55; 48 (toujours des codes ASCII). Ce format est facile à visualiser mais est très gourmand en taille...

Précédemment le tableur convertissait automatiquement ces trois codes ASCII en 170.

Exemple 2 : une image au format PGM de type P5.

On ne peut plus ici travailler avec un tableur ou un éditeur de texte : les niveaux de gris sont en effet codés en binaire (mais affichés ici en décimal).

Avec notre éditeur hexadécimal, nous obtenons l'affichage ci-contre.

Le contenu de l'octet est bien 170. En revanche le code ASCII affiché à droite devient sans intérêt.

Une astuce didactique pour faciliter le travail va consister à ajuster la taille du tableau d'affichage de l'éditeur hexadécimal avec la taille de l'image.

	00	01	02	03	04	05	06	07	
00000000	80	53	10	35	32	71	114	111	P5.# Gro
00000008	117	112	101	32	32	32	32	32	upe
00000010	32	32	32	32	32	32	73	83	IS
00000018	78	10	49	56	32	49	56	10	N.18 18.
00000020	50	53	53	10	170	170	170	170	255.ªªªª
00000028	170	170	0	0	0	0	0	0	ªª.....
00000030	170	170	170	170	170	170	170	170	ªªªªªªªª
00000038	170	170	0	0	255	255	255	0	ªª..ÿÿÿ.
00000040	255	255	0	0	170	170	170	170	ÿÿ..ªªªª
00000048	170	170	170	0	255	255	255	255	ªªª.ÿÿÿÿ

Il faut aussi jouer sur la taille de l'en-tête, en ajoutant des espaces.

Cette fois la présentation permet un travail très efficace et l'affichage en mode texte retrouve même du sens puisque l'on arrive presque à retrouver l'image !

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	10	11	
00000000	80	53	10	35	32	71	114	111	117	112	101	32	32	32	32	32	32	32	P5.# Groupe
00000012	32	32	32	32	73	83	78	10	49	56	32	49	56	10	50	53	53	10	ISN.18 18.255
00000028	170	170	170	170	170	170	0	0	0	0	0	0	170	170	170	170	170	170	ª ª ª ª ª ª . . . . . ª ª ª ª ª ª
00000036	170	170	170	170	0	0	255	255	255	0	255	255	0	0	170	170	170	170	ª ª ª ª . . ÿ ÿ ÿ . ÿ ÿ . . ª ª ª ª
00000048	170	170	170	0	255	255	255	255	255	0	0	255	255	255	0	170	170	170	ª ª ª . ÿ ÿ ÿ ÿ ÿ . . ÿ ÿ ÿ . ª ª ª
0000005a	170	170	0	255	255	255	255	255	255	0	0	0	255	255	255	0	170	170	ª ª . ÿ ÿ ÿ ÿ ÿ ÿ . . . ÿ ÿ ÿ . ª ª
0000006c	170	0	255	255	255	255	0	0	0	0	0	0	0	255	255	0	0	170	ª . ÿ ÿ ÿ ÿ . . . . . ÿ ÿ . . ª
0000007e	170	0	255	255	0	0	0	0	0	0	0	0	0	0	255	255	0	170	ª . ÿ ÿ . . . . . ÿ ÿ . ª
00000090	0	255	255	0	0	0	0	0	0	0	0	0	0	0	255	255	0	0	. ÿ ÿ . . . . . ÿ ÿ . .
000000a2	0	255	255	0	0	0	0	0	255	0	0	0	0	255	255	255	0	0	. ÿ ÿ . . . . . ÿ . . . . ÿ ÿ ÿ .
000000b4	0	255	0	0	0	0	0	255	255	0	0	0	255	255	255	255	0	0	. ÿ . . . . . ÿ ÿ . . . ÿ ÿ ÿ ÿ .
000000c6	0	255	0	0	0	0	255	255	255	0	0	255	255	255	255	255	0	0	. ÿ . . . . ÿ ÿ ÿ . . ÿ ÿ ÿ ÿ ÿ .
000000d8	0	255	0	0	0	255	255	255	255	0	255	255	255	255	255	0	0	0	. ÿ . . . ÿ ÿ ÿ ÿ . ÿ ÿ ÿ ÿ ÿ . .
000000ea	0	0	0	0	255	255	255	255	255	255	255	255	255	255	255	0	0	0	. . . . ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ . .
000000fc	170	0	0	0	255	255	255	255	255	255	255	255	255	255	0	0	0	170	ª . . . ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ . . . ª
0000010e	170	0	0	0	0	255	255	255	255	255	255	255	0	0	0	0	0	170	ª . . . . ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ . . . . . ª
00000120	170	170	0	0	0	0	255	255	255	0	0	0	0	0	0	0	170	170	ª ª . . . . ÿ ÿ ÿ . . . . . ª ª
00000132	170	170	170	0	0	0	0	255	255	0	0	0	0	0	0	170	170	170	ª ª ª . . . . ÿ ÿ . . . . . ª ª ª

Une petite synthèse sur le format P5 qui pourra être construite par les élèves après observation.

L'image est codée selon la structure suivante :

- Un nombre « magique » (P5)
- 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
- La valeur maximale utilisée pour coder les niveaux de gris, cette valeur doit être inférieure à 65536 (codée en caractères ASCII)
- Un caractère d'espacement
- Données binaires de l'image :
  - L'image est codée ligne par ligne en partant du haut
  - Chaque ligne est codée de gauche à droite
  - Chaque pixel est codé par 1 ou 2 octets selon que la valeur maximale est inférieure ou supérieure à 256. Un pixel noir est codé par la valeur 0, un pixel blanc est codé par la valeur maximale et chaque niveau de gris est codé par une valeur entre ces deux extrêmes, proportionnellement à son intensité.

Toutes les lignes commençant par # sont ignorées (commentaire).

4 / Envisager une progression en travaillant avec plusieurs formats

La diversité des formats et leurs différentes complexités sont autant de possibilités pour envisager une progression fine dans la maîtrise du codage binaire.

- Le format PGM (P2) est très simple d'approche : le niveau gris est codé en ASCII. Il est aussi lourd, puisque trois octets sont nécessaires pour coder un pixel.
- Le format PGM (P5) est plus efficace : le codage en binaire n'occupe qu'un seul octet pour coder 256 niveaux de gris. Mais pourquoi sommes-nous limité à 256 ?
- Une voie s'ouvre alors pour parler de la notion d'octet (jusqu'à présent considéré comme une case) et de bit …
- L'éditeur hexadécimal permet d'ailleurs un affichage en binaire directement (ci-contre).

	0	1	2	3
0	01010000	00110101	00001010	00100011
18	00100000	00100000	00100000	00100000
36	10101010	10101010	10101010	10101010
54	10101010	10101010	10101010	10101010
72	10101010	10101010	10101010	00000000
90	10101010	10101010	00000000	11111111

- Notre gris de niveau 170 prend donc un nouvel aspect : 1 0 1 0 1 0 1 0 = $2^7+2^5+2^3+2$
- On peut aussi envisager de travailler avec le format Targa 8 de type 3 (Uncompressed, Black and White), dont le principe est le même mais l'en-tête plus complexe.

- On pourra enfin en prolongement, aborder le BMP dont le codage part du coin inférieur gauche en remontant vers le haut de l'image : facile à identifier en testant !
- Tout ces formats ont un défaut : leur taille. On peut alors faire une ouverture sur la notion de compression et aborder le principe du RLE dont l'efficacité est très liée au type de l'image.

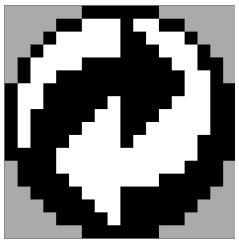
Une fois la problématique des images en niveaux de gris bien comprise, on peut faire aborder les images avec de la transparence (ou canal alpha) ; le codage est différent car il faut distinguer les pixels « transparents » de ceux qui ont une couleur.

5 / Construire des mini-projets, le passage à la programmation

Une fois le principe du codage bien compris, on peut l'exploiter pour transformer un peu notre image.

On se propose par exemple d'inverser les couleurs : un gris codé par x (compris entre 0 et 255) sera transformé en un gris codé par 255-x.

Partons d'une image au format PGM (P5) et de son codage :



L'en-tête occupe les 36 premiers octets qui ne devront pas être modifiés. Le codage commence donc au 37ème octet et se termine à la fin du fichier.

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	10	11	
00000000	80	53	10	35	32	71	114	111	117	112	101	32	32	32	32	32	32	32	P5.# Groupe
00000012	32	32	32	32	73	83	78	10	49	56	32	49	56	10	50	53	53	10	ISN.18 18.255.
00000024	170	170	170	170	170	170	0	0	0	0	0	0	170	170	170	170	170	170	ª.....
00000036	170	170	170	170	0	0	255	255	255	0	255	255	0	0	170	170	170	170	ª..ÿÿÿ.ÿÿ..ª
00000048	170	170	170	0	255	255	255	255	255	0	0	255	255	255	0	170	170	170	ª.ÿÿÿÿÿ..ÿÿÿ.ª
0000005a	170	170	0	255	255	255	255	255	255	0	0	0	255	255	255	0	170	170	ª.ÿÿÿÿÿÿ...ÿÿÿ.ª
0000006c	170	0	255	255	255	255	0	0	0	0	0	0	0	255	255	0	0	170	ª.ÿÿÿÿ.....ÿÿ..ª
0000007e	170	0	255	255	0	0	0	0	0	0	0	0	0	0	255	255	0	170	ª.ÿÿ.....ÿÿ.ª
00000090	0	255	255	0	0	0	0	0	0	0	0	0	0	0	255	255	0	0	.ÿÿ.....ÿÿ..
000000a2	0	255	255	0	0	0	0	0	255	0	0	0	0	255	255	255	0	0	.ÿÿ.....ÿ....ÿÿÿ..
000000b4	0	255	0	0	0	0	0	255	255	0	0	0	255	255	255	255	0	0	.ÿ.....ÿÿ...ÿÿÿÿ..
000000c6	0	255	0	0	0	0	255	255	255	0	0	255	255	255	255	255	0	0	.ÿ....ÿÿÿ..ÿÿÿÿÿ..
000000d8	0	255	0	0	0	255	255	255	255	0	255	255	255	255	255	0	0	0	.ÿ...ÿÿÿÿ.ÿÿÿÿÿ...
000000ea	0	0	0	0	255	255	255	255	255	255	255	255	255	255	255	0	0	0	....ÿÿÿÿÿÿÿÿÿ...
000000fc	170	0	0	0	255	255	255	255	255	255	255	255	255	255	0	0	0	170	ª...ÿÿÿÿÿÿÿÿ...ª
0000010e	170	0	0	0	0	255	255	255	255	255	255	255	0	0	0	0	0	170	ª....ÿÿÿÿÿÿ.....ª
00000120	170	170	0	0	0	0	255	255	255	0	0	0	0	0	0	0	170	170	ªª....ÿÿÿ.....ªª
00000132	170	170	170	0	0	0	0	255	255	0	0	0	0	0	0	170	170	170	ªªª....ÿÿ.....ªªª
00000144	170	170	170	170	0	0	0	0	255	0	0	0	0	0	170	170	170	170	ªªªª....ÿ.....ªªªª
00000156	170	170	170	170	170	170	0	0	0	0	0	0	170	170	170	170	170	170	ªªªªªª.....ªªªªªª

On demande de ne pas modifier le fichier initial mais de créer un nouveau fichier (le traitement peut donc être compris comme un filtre prenant une image en « entrée » et rendant une autre image en « sortie »).

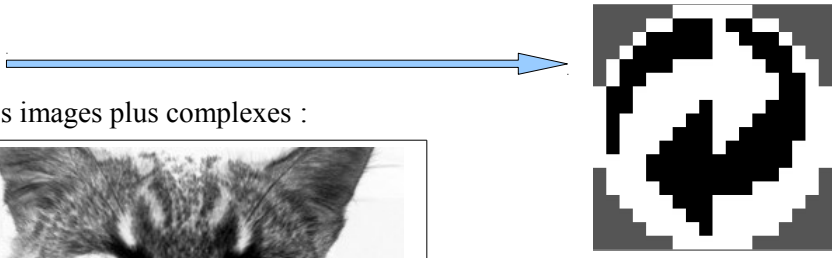
On peut traduire la démarche par l'algorithme suivant :

```
Ouvrir le fichier initial
Créer un fichier de sortie
Recopier l'entête du fichier initial dans le fichier de sortie
Se placer sur le premier octet du fichier au niveau de la zone de codage des données
Pour chaque octet de la zone de codage :
      Appeler la fonction inversion avec pour paramètre le contenu x de l'octet actuel
      elle retourne l'octet modifié(255- x)
      Écrire dans le fichier de sortie l'octet modifié
```

Et voici une traduction possible en langage Python :

```
import os
def inversion (nbr):        # On définit ici la fonction inversion
      return 255-nbr        # On inverse les couleurs
# image initiale en mode binaire, modifier le chemin !
fichier_initial = open("c:/test.pgm", "rb")
# image transformée en mode binaire
fichier_modif = open ("c:/test_sortie.pgm", "wb")
# on recopie l'entête qui ne sera pas modifié (les 36 premiers octets)
fichier_modif.write(fichier_initial.read(36))
# Extraction de chacun des octets dans une liste à partir du 37ème
malistoctet = [ord(i) for i in fichier_initial.read()]
# Les deux affichages qui ne font pas partie de l'algorithme, seulement pour contrôler et mettre au point.
print malistoctet           # Vérification de la liste: on l'affiche
print len(malistoctet)      # On affiche la taille de la liste pour vérifier
# Ecriture dans le fichier de sortie
for i in malistoctet:       # On boucle sur le nombre d'octets de la partie à coder
# On utilise la fonction chr() qui fait le travail inverse de ord()
# et on utilise la fonction d'inversion précédemment définie :
      fichier_modif.write(chr(inversion(i)))
fichier_initial.close()     # Fermeture du fichier source
fichier_modif.close()       # Fermeture du fichier destination (essentiel)
os.system("pause")
```

Le résultat obtenu est montré ci-contre :



Cet algorithme fonctionne très bien avec des images plus complexes :