Écriture d'un *parser* d'images

Olivier Levillain

M2 FIIL 2018-2019

Introduction

L'objectif de ce TP est d'écrire un *parser* d'images au format MiniPNG, une version simplifiée du format d'images standard PNG (défini dans la RFC 2083).

Le TP se décompose en trois grandes parties :

— la première consiste en l'écriture du *parser* d'images en noir et blanc, en suivant le format décrit ci-dessous ;

— la seconde a pour but de permettre l'affichage des images lues ;

— enfin, la troisième vise à l'extension du *parser* à des images en couleurs, utilisant ou non une palette.

L'objectif de cette séance est de répondre correctement aux cinq premières questions qui rapportent 16 points. Pour les plus motivés d'entre vous, il sera possible de poursuivre le TP jusqu'au bout et afficher une image en couleur.

Vous pouvez écrire votre projet en C, Python, Java ou OCaml. Si vous souhaitez utiliser un autre langage, vous êtes priés de valider avec moi le langage (afin de vous assurer que je puisse vous relire). Étant donné le sujet du cours, il vous est bien sûr demandé de penser au-delà du fonctionnel et de proposer du code sécurisé !

Avant la fin du TP, vous devrez envoyer un message à l'adresse `cours-POS@picty.org` avec votre prénom, votre nom et les fichiers source correspondant à votre travail. Si vous souhaitez poursuivre après ce TP pour casser la primitive, vous pourrez envoyer vos questions et votre code **à la même adresse**.

1 MiniPNG : un format d'images en noir et blanc

Une image au format MiniPNG est constituée d'un marqueur (parfois appelé *magic number*) et d'une liste de blocs. Le fichier commence donc par la chaîne de caractères contenant les 8 octets suivants « `Mini-PNG` ». Ensuite, le fichier est composé de blocs de la forme suivante :

Offset	Nom du champ	Taille du champ	Description du champ
0	Type de bloc	1 octet	Un caractère décrivant le bloc
1	Longueur du bloc	4 octets	Un entier indiquant la longueur du contenu l
5	Contenu du bloc	l octets	Le contenu dépend du type de bloc

Dans un premier temps, nous allons décrire les blocs nécessaires à la description d'images en noir et blanc :

— un bloc d'en-tête `H` *(header)* ;

— zéro, un ou plusieurs blocs de commentaires `C` *(comment)* ;

— un ou plusieurs blocs de données `D` *(data)*.

Description du bloc `H`

Le contenu du bloc `H` doit être le suivant :

Offset	Nom du champ	Taille du champ	Description du champ
0	Largeur de l'image	4 octets	Un entier décrivant le nombre de pixels en largeur
4	Hauteur de l'image	4 octets	Un entier décrivant le nombre de pixels en hauteur
8	Type de pixels	1 octet	0 = noir et blanc, 1 = niveaux de gris, 2 = palette, 3 = couleurs 24 bits

Pour cette première partie, on ne s'intéressera qu'aux images ayant un type 0 (noir et blanc).

Description du bloc `C`

Le contenu du bloc `C` est une simple chaîne de caractères ASCII affichables.

Description du bloc `D`

Le contenu des blocs `D` doit être concaténés pour obtenir un *bitmap* de l'image.

Avec des pixels noirs et blancs, chaque pixel doit être codé sur un bit (0 pour noir, 1 pour blanc). Les autres types de pixels peuvent être ignorés pour l'instant. Les pixels sont rangés de haut en bas, puis de gauche à droite.

2 Premiers pas avec MiniPNG

Tous les programmes qui vous sont demandés doivent prendre en argument un nom de fichier à traiter. Vos programmes devront signaler une erreur par un code de retour non nul, et afficher si possible un message d'erreur explicite.

Question 1. Écrivez un *parser* lisant un fichier au format Mini-PNG et affichant les informations le concernant : largeur et hauteur de l'image, type de pixels.
Voici le transcript attendu résultant de l'appel de votre programme sur le fichier `A.mp` :

```
% ./Q1 A.mp
Largeur : 8
Hauteur : 10
Type de pixel : 0 (noir et blanc)
```

Question 2. Améliorez votre programme pour qu'il affiche également les commentaires présents dans le fichier.
Le transcript attendu est le suivant :

```
% ./Q2 A.mp
Largeur : 8
Hauteur : 10
Type de pixel : 0 (noir et blanc)
Commentaires :
"La lettre A"
```

Question 3. Ajoutez un test pour vérifier que les données obtenues sont conformes au nombre d'octets attendus.

3 Affichage des images en noir et blanc

Question 4. Affichez l'image en mode texte, en affichant les pixels noirs comme des `X` et les pixels blancs comme des espaces.
Le résultat attendu pour `A.mp` est le suivant :

```
  XXXX
 X    X
X      X
X      X
X      X
XXXXXXXX
X      X
X      X
X      X
X      X
```

Question 5. Écrivez un fichier Mini-PNG valide représentant la première lettre de votre prénom.

4 Ajout des autres types de pixels

Niveaux de gris et couleur 24 bits

En mode niveaux de gris, les pixels sont codés sur 1 octet (`0x00` représentant le noir et `0xff` le blanc). En mode couleurs 24 bits, les pixels sont codés sur 3 octets représentant les trois composantes RVB (Rouge, Vert, Bleu).

Question 6. Reprenez les questions 1 à 3 pour supporter les nouveaux modes.

Question 7. Ajoutez le support de l'affichage de ces images en mode graphique. Alternativement, vous pourrez écrire un convertisseur Mini-PNG vers un format *portable pixmap* (`https://fr.wikipedia.org/wiki/Portable_pixmap`), que vous pourrez voir avec un lecteur classique.

Gestion de la palette

Afin de gagner de la place, si le nombre de couleurs affichées est limité, il est possible de décrire une palette contenant jusqu'à 256 couleurs. Pour cela, on utilisera le nouveau bloc de type `P`, dont le contenu est composé de n entrées RVB, chacune codée sur 3 octets.

La première entrée sera l'entrée 0, et les données de l'image seront ensuite composées de valeurs sur un octet représentant l'entrée de la palette correspondante.

Question 8. Ajoutez le support de la palette dans votre *parser* et écrivez un outil capable d'afficher les entrées de la palette.

Question 9. Ajoutez le support de l'affichage des images utilisant une palette.