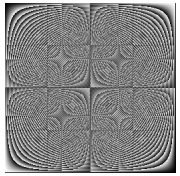
Une application : Traitement d'images

Christian Lasou, Nour-Eddine Oussous, Éric Wegrzynowski

Licence ST-A, USTL - API1

31 mars 2008



Objectif

- découvrir un format simple d'image ;
- une structure de données pour représenter une image dans un programme en Pascal ;
- et programmer quelques traitements d'image.

Les images numériques

- Les images sont constituées de pixels, un pixel ayant une couleur.
- Les pixels sont disposés dans un rectangle ayant une largeur et une hauteur. La largeur et la hauteur sont souvent exprimées en nombre de pixels. Une image de dimension $L \times H$ possède ainsi LH pixels.
- La couleur d'un pixel est codée par un nombre entier.

Formats d'images

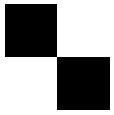
Il existe beaucoup de formats pour stocker les images numériques dans des fichiers.

- format `BMP` (bitmap) ;
- format `JPEG` (Joint Photographic Experts Group) ;
- format `PNG` (Portable Network Graphics) ;
- format `PGM` (Portable Gray Map) ;
- et bien d'autres...

Format PGM

```
P2  # "nombre magique"
# CREATOR: API1
2 2 # Largeur x hauteur
255 # valeur du blanc
0   # un pixel noir
255 # un pixel blanc
255 # un pixel blanc
0   # un pixel noir
```

Format texte très simple à comprendre permettant de coder des images en nuances de gris.



Limitations

On se fixe les limitations suivantes :

- limitation des dimensions horizontale et verticale à 1024 ;
- limitation du nombre de nuances de gris à 256 ; ainsi les nuances seront codées par des entiers compris entre 0 (noir) et 255 (blanc).

le type IMAGE

```
const
  DIMMAX = 1024 ;
type
  COULEUR = BYTE ;
  IMAGE   = record
      largeur, hauteur : 1..DIMMAX ;
      pixels : array[1..DIMMAX,1..DIMMAX] of COULEUR ;
  end {record};
```

Objectifs

Pouvoir

1. entrer en mémoire la description d'une image au format PGM ;
2. écrire le contenu d'une variable de type IMAGE dans un format PGM.

Procédure d'entrée

```
procedure lireImage(out img : IMAGE);
var
   i,j : CARDINAL;
begin
   readln;
   readln;
   readln(img.largeur, img.hauteur);
   readln;
   for i := 1 to img.hauteur do
      for j := 1 to img.largeur do
         readln(img.pixels[i,j]);
end {lireImage};
```


Procédure de sortie

```
procedure ecrireImage(const img : IMAGE);
var
   i,j : CARDINAL;
begin
   writeln('P2');
   writeln('# CREATOR: API1 (Licence ST-A S2 USTL)');
   writeln(img.largeur,' ',img.hauteur);
   writeln(high(COULEUR));
   for i := 1 to img.hauteur do
      for j := 1 to img.largeur do
         writeln(img.pixels[i,j]);
end {ecrireImage};
```


Redirection des entrée/sortie

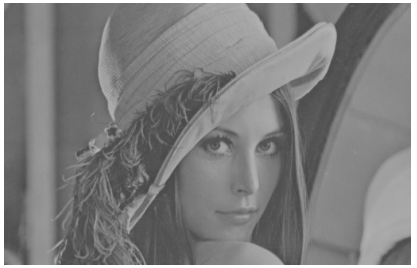
Redirection de l'entrée standard

```
./copier-image < image.pgm
P2
# CREATOR: API1 (Licence ST-A S2 USTL)
256 256
255
1
2
3
4
...
```

Redirection de la sortie standard

```
./copier-image < image.pgm > image2.pgm
```

Adoucir une image



La transformation

```
function adoucie(img : IMAGE) : IMAGE;
var
   res : IMAGE;
   i,j : 1..DIMMAX;
begin
   res.largeur := img.largeur;
   res.hauteur := img.hauteur;
   for i := 1 to img.hauteur do
      for j := 1 to img.largeur do
         res.pixels[i,j] := (img.pixels[i,j]
               + (high(COULEUR) div 2)) div 2;
   adoucie := res;
end {adoucie};
```


Le programme principal

Le code

```
var
   img1, img2 : IMAGE;
begin
   lireImage(img1);
   img2 := adoucie(img1);
   ecrireImage(img2);
end.
```

et son utilisation

```
./adoucir < lena.pgm > lena-adoucie.pgm
```