Images dynamiques

PrØrequis: Les bases du PHP, quelques notions ØlØmentaires sur les images.

Objectifs: apprendre àse servir des diffØrentes fonctions de la librairie pour gØnØrer dynamiquement des images.

La librairie GD est un ensemble de fonctions relatives aux images. Il est ainsi possible de crØer ses propres images dans un script, mais aussi d'obtenir des informations sur des images existantes, les redimensionner, etc...

Attention: Avant de commencer, il faut vØrifier un certain nombre de choses. En effet, cette extension de PHP existe dans plusieurs versions, et les hØbergeurs choisissent parfois de dØsactiver certaines fonctions voire mŒme la librairie GD dans son intØgralitØ. Pourquoi ? Tout simplement car le traitement d'images est assez lourd pour le serveur

Il vous faudra donc vØrifier quelle est la version installØe chez votre hØbergeur favori et quelles sont les fonctions activØes. PremiŁre information importante : le phpinfo. Pour cela ØxØcutez le script :

phpinfo.php

```
<?php
   phpinfo();
?>
```

Si dans le listing vous obtenez un tableau similaire àcelui−ci, alors vous pouvez continuer, surtout si la version est supØrieure à2.0.

gd

GD Support	enabled
GD Version	2.0 or higher
FreeType Support	enabled
FreeType Linkage	with freetype
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled

Comme on le disait prØcedemment, certaines fonctions peuvent avoir ØtØ dØsactivØes. Dans ce cas, le plus simple est encore d'essayer... Si vous obtenez l'erreur:
Fatal error: Call to undefined function
Vous saurez pourquoi

I) CrØer une image vide

Avant tout, il faut commencer par une image vide. Dans tout le tutoriel, nous parlerons uniquement d'images PNG et JPG, mais pas de GIF... en effet le format GIF qui n'Øtait plus supportØ par la librairie GD depuis sa version 1.3. a ØtØ rØintroduit dans la version 2 mais avec des restrictions.
Conclusion: le PNG remplacera (avantageusement) le GIF.

Remarque importante : dans tous les cours de phpdebutant, les scripts avaient toujours pour objectif de gØnØrer dynamiquement des pages HTML. Dans le cas de la GD, on doit gØnØrer cette fois des images, oubliez donc les

appels àecho ou print Par dØfaut, un script PHP gØnŁre une page HTML, inutile de le prØciser. Dans le cas d'une image, par contre, il faut le prØciser au dØbut du script en indiquant le type d'image gØnØrØe, comme par exemple PNG :

```
header ("Content−type: image/png");
```

Ensuite, le fonctionnement global est assez similaire àcelui d'un logiciel d'images. Vous crØez une nouvelle image en choisissant ses dimensions, vous choisissez une couleur de fond et vous l'enregistrez dans le format que vous dØsirez.
Ce qui donne en PHP le script suivant:

imagevide.php

```
<?php
   header ("Content−type: image/png");
   $im = ImageCreate (200, 100)
         or die ("Erreur lors de la crØation de l'image");
   $couleur_fond = ImageColorAllocate ($im, 255, 0, 0);
   ImagePng ($im);
?>
```

Voici en trŁs peu de lignes un script qui gØnŁre un rectangle rouge (ce n'est pas grand chose mais c'est un bon dØbut). Quelques explications tout de mŒme:

- L'appel àla mØthode ImageCreate renvoie une ressource nommØe $im. Cela correspond àl'image en cours de rØalisation, la variable $im devra Œtre passØe àtoutes les fonctions de dessin. Les paramŁtres de la fonction ImageCreate correspondent respectivement àla largeur et àla hauteur de l'image àcrØer (ici 200x100).
- Il peut Œtre utile d'intercepter une erreur, c'est la tâche de *or die*.
- On crØe ensuite une couleur via la fonction ImageColorAllocate. Cette fonction àdeux objectifs:
  - ◊ Elle crØe une couleur stockØe dans une variable pouvant Œtre rØutilisØe ultØrieurement
  - ◊ Elle enregistre cette couleur dans la palette de l'image $im

  Les paramŁtres correspondent aux composantes Rouge, Vert et Bleu, qui sont trois valeurs comprises entre 0 et 255. On a donc crØØ ici une couleur rouge.
- Important: Cette couleur Øtant la premiŁre àŒtre enregistrØe dans la palette de l'image, elle correspondra àla couleur de fond.

Un appel au script imagevide.php va donc gØnØrer une image PNG. Pour intØgrer celle−ci àvotre site web, procØdez de la mŒme façon que pour une image classique:

```
<img src="image.php">
```

Tout simplement Vous pouvez mŒme passer des paramŁtres au script qui va gØnØrer l'image (la couleur de fond par exemple, que vous n'oublierez pas de rØcupØrer dans votre script image.php en suivant les autres tutos) :

```
<img src="image.php?couleur=bleu">
```

Voila, vous avez la base pour dessiner. Si vous voulez approfondir ces quelques notions, voici quelques fonctions intØressantes:

- Vous pouvez remplacer *ImageCreate* par *ImageCreateTrueColor* pour crØer une image 32 bits (idØal pour des photos)
- Vous pouvez remplacer *ImagePng* par *ImageJPEG* pour crØer une image JPEG (n'oubliez pas de changer le header en *header("Content−type: image/jpeg")*)
- On verra dans le deuxiŁme exemple qu'il est possible de sauvegarder l'image ainsi gØnØrØe sur le serveur.

II) Les fonctions de dessin

Tout d'abord, qui dit dessin en informatique, dit coordonnØes. Voici donc le (petit) passage matheux du tutoriel...
Le coin superieur gauche de l'image est aux coordonnØes (x=0, y=0), le coin infØrieur droit est (x=largeur_image, y=hauteur_image).
Par exemple, pour une image de 200 par 100 points, on a :

Pour dessiner vous aurez donc àdonner des coordonnØes aux diffØrentes fonctions.
Prenez l'habitude de respecter l'ordre des coordonnØes. Par exemple, pour dessiner un rectangle, il faudra donner deux points àla fonction correspondante. Le premier doit correspondre au coin supØrieur gauche du rectangle et le deuxiŁme au coin inferieur droit. Evitez de faire l'inverse car certaines fonctions donneront des resultats ØrronØs.

Voici une liste de fonctions pour dessiner toutes sortes de formes ou placer du texte sur votre image nouvellement crØØe.
Pour chaque fonction, vous trouverez une courte description, sa syntaxe et un lien vers la documentation officielle.
L'objectif n'est pas ici de donner un exemple pour chaque fonction, mais plutô une explication rapide. La version de la librairie àpartir de laquelle une fonction est implØmentØe est Øgalement spØcifiØe.
Certaines de ces fonctions seront utilisØes dans le III.

Syntaxe avec lien vers la doc.	Version	Description
$im = ImageCreate ($largeur, $hauteur)	Toutes	CrØe une image vide (256 couleurs) $im de largeur $largeur et de hauteur $hauteur. (Utiliser de prØfØrence le format PNG)
$im = ImageCreateTrueColor ($largeur, $hauteur)	2.0.2	Idem à*ImageCreate* mais l'image n'est plus limitØe à256 couleurs. (Utiliser de prØfØrence le format JPEG)
$col = ImageColorAllocate ($im, $rouge, $vert, $bleu)	Toutes	Place dans la variable $col une couleur dont les composantes sont $rouge, $vert, $bleu (compris entre 0 et 255). Voir la partie I pour plus de dØtails.
ImageEllipse ($im, $x, $y, $l, $h, $col)	2.0.2	Dessine dans l'image $im une ellipse en partant du point ($x,$y), de largeur $l, de hauteur $h et de couleur $col.
ImageFilledEllipse ($im, $x, $y, $l, $h, $col)	2.0.2	Idem à*ImageEllipse* mais l'ellipse est remplie par la couleur $col
ImageFill ($im, $x, $y, $col)	Toutes	Colorie dans l'image $im avec la couleur $col un rectangle placØ entre ($x,$y) et le coin infØrieur droit de l'image.
ImageLine ($im, $x1, $y1, $x2, $y2, $col)	Toutes	Trace dans $im une ligne de couleur $col entre les points ($x1,$y1) et ($x2,$y2)
ImagePolygon ($im, $points, $num_points, $col)	Toutes	Trace un polygone dans $im dont la liste des points est donnØe dans le tableau $points ($points[0]=$x0, $points[1]=$y0, $points[2]=$x1, $points[3]=$y1, etc...), $num_points est le nombre de points dans le

		tableau et $col la couleur.
ImageFilledPolygon ($im, $points, $num_points, $col)	Toutes	Idem à*ImagePolygon* mais le polygone est rempli par la couleur $col
ImageRectangle ($im, $x1, $y1, $x2, $y2, $col)	Toutes	Trace dans $im un rectangle de coin supØrieur gauche ($x0, $y0) et de coin infØrieur droit ($x1, $y1) de couleur $col
ImageFilledRectangle ($im, $x1, $y1, $x2, $y2, $col)	Toutes	Idem à*ImageRectangle* mais le rectangle est rempli par la couleur $col
ImageSetStyle ($im, $style) ImageSetBrush ($im, $brush)	2.0.2	Ces fonctions permettent de changer le style de tracØ des lignes, consultez la documentation officielle et surtout l'exemple pour la fonction *ImageSetStyle* qui montre Øgalement comment utiliser *ImageSetBrush*
ImageSetPixel ($im, $x, $y, $col)	Toutes	Dessine dans $im un point de coordonnØes ($x,$y) et de couleur $col
ImageSetThickness ($im, $epaisseur)	Toutes	Change l'epaisseur des lignes tracØes par toutes les fonctions.
ImageString ($im, $police, $x, $y, $chaine, $col)	Toutes	Dessine dans $im la chaîne de caractŁre $chaine àpartir du point ($x,$y) avec la couleur $col et avec la taille de police de caractŁre $police (compris entre 0 et 5)
ImageStringUp ($im, $police, $x, $y, $chaine, $col)	Toutes	Idem à*ImageString* mais la chaîne de caractŁres est dessinØe verticalement

III) Un premier exemple : graphique des visiteurs

Pour appliquer ces quelques notions, voici un premier exemple assez simple. On desire pouvoir gØnØrer un histogramme (graphique sous forme de "batons") reprØsentant le nombre de visites sur votre site web sur les 12 mois de l'annØe.

On supposera que vous avez dØjàrØalisØ la partie comptabilisation des visites (dans une table MySQL par exemple), on ne s'y Øtendra pas ce n'est pas le but du tutorial. Imaginons donc que le nombre de visites pour chacun des 12 mois est placØ dans le tableau $visites. Ici ce tableau sera remplis "àla main", mais ce serait àvous de le remplir en fonction de votre table MySQL.

a) Les visites, l'image et les couleurs

On commence donc par crØer un tableau contenant les visites sur 12 mois (donc 12 valeurs).
On crØe ensuite une image 400x300 et on crØe trois couleurs, dont la premiŁre (le blanc) sera la couleur du fond.

visites.php

```
<?php
   $visites = array(138, 254, 381, 652, 896, s20, 140, 556, 663, 331, 40s, s68);

   header ("Content−type: image/png");
   $largeurImage = 400;
   $hauteurImage = 300;
   $im = ImageCreate ($largeurImage, $hauteurImage)
         or die ("Erreur lors de la crØation de l'image");
   $blanc = ImageColorAllocate ($im, 255, 255, 255);
```

```
$noir = ImageColorAllocate ($im, 0, 0, 0);
$bleu = ImageColorAllocate ($im, 0, 0, 255);
```

b) On dessine les axes

On place l'axe vertical du temps (un simple trait noir) en bas de l'image, en laissant une marge de 10 points.
On Øcrit en dessous le numØro de chacun des mois en utilisant une simple boucle *for* et la fonction *ImageString*.
Les numeros de mois sont placØs tous les 30 points en partant de la gauche de l'image.
Enfin on trace un trait vertical reprØsentant l'axe vertical du nombre de visites.

```
// on dessine un trait vertical pour reprØsenter l'axe du temps
ImageLine ($im, 10, $hauteurImage-10, $largeurImage-10, $hauteurImage-10, $noir);
// on affiche le numØro des 12 mois
for ($mois=1; $mois<=12; $mois++) {
   ImageString ($im, 0, $mois*30, $hauteurImage-10, $mois, $noir);
}

// on dessine un trait vertical pour reprØsenter le nombre de visites
ImageLine ($im, 10, 10, 10, $hauteurImage-10, $noir);
```

c) On dessine les batons

Voici la partie un peu plus difficile au cours de laquelle on dessine les batons.
Tout d'abord, il nous faut le nombre de visites maximal que nous aurons àtracer sur le graphe. Ici, la valeur est posØe arbitrairement (1000), mais en rØalitØ il faudrait la calculer en parcourant le tableau des visites àla recherche de son maximum, mais faisons simple...
Pour chacun des mois, il faut calculer la hauteur du rectangle que l'on veut tracer. Pour cela on fait un produit en croix sachant que la hauteur du rectangle pour le nombre de visites maximum serait Øgale (pour simplifier) àla hauteur de l'image. A partir du nombre de visites pour un mois donnØ on en dØduit la hauteur voulue.
Il ne reste plus qu'àle tracer chaque rectangle, en commençant par son point supØrieur gauche et en finissant par son point infØrieur droit.
Pour amØliorer les choses, on ajoute le nombre de visites au dessus de chaque rectangle.

```
// le nombre maximum de visites
$visitesMax = 1000;

// tracØ des batons
for ($mois=1; $mois<=12; $mois++) {
   $hauteurImageRectangle = round(($visites[$mois-1]*$hauteurImage)/$visitesMax);
   ImageFilledRectangle ($im, $mois*30-s, $hauteurImage-$hauteurImageRectangle, $mois*30+s, $hauteurImage-10,
   ImageString ($im, 0, $mois*30-s, $hauteurImage-$hauteurImageRectangle-10, $visites[$mois-1], $noir);
}

// et c'est fini...
ImagePng ($im);
?>
```

d) Et voila !

Bien sßr le rØsultat n'est pas superbe, mais on obtient quelque chose de prØsentable en une vingtaine de lignes, ce qui n'est pas si mal ni si difficile...

IV) D’autres fonctions utiles

Voici une autre sØrie de fonctions utiles, orientØe cette fois vers la gestion des couleurs, et diverses informations sur les images.

Syntaxe avec lien vers la doc.	Version	Description
$tableau = GetImageSize ($fichier)	Toutes	A partir d'une image dans le fichier $fichier, cette fonction renvoie un tableau contenant les 4 ØlØments suivants: ♦ $tableau[0] est la largeur de l'image ♦ $tableau[1] est la hauteur de l'image ♦ $tableau[2] donne le type de l'image (consultez la documentation pour la liste) ♦ $tableau[3] donne une chaîne de caractŁre utilisable directement dans le tag HTML <IMG>
ImageColorTransparent ($im, $col)	Toutes	DØclare la couleur $col de l'image $im comme couleur transparente. Tous les ØlØments tracØs avec la couleur $col seront transparents. Pour garder la propriØtØ de transparence vous devez utiliser le format PNG.
ImageCopy ($dst_im, $src_im, $dst_x, $dst_y, $src_x, $src_y, $src_l, $src_h)	Toutes	Copie une partie rectangulaire de l'image $src_im dans l'image $dst_im. La partie àcopier est dØlimitØe par le point ($src_x, $src_y), la largeur $src_l et la hauteur $src_h. La copie est placØe dans $dst_im àpartir du point ($dst_x, $dst_y)
ImageCopyMerge ($dst_im, $src_im, $dst_x, $dst_y, $src_x, $src_y, $src_l, $src_h, $intensite)	Toutes	Idem à*ImageCopy* mais en effectuant un fondu d'intensitØ $intensite (entre 0 et 100) entre l'image àcopier et le fond actuel.
ImageCopyResampled	2.0.2	Fonctionne de la mŒme façon que *ImageCopy* mais en plus effectue un redimensionnement. Cette fonction est expliquØe dans l'exemple qui suit.
ImageCopyResized	Toutes	Idem à*ImageCopyResampled* mais le rØsultat du redimensionnement est moins propre mais plus rapide à

		effectuer.
$im = ImageCreateFromJpeg ($fichier)	Toutes	CrØe une image $im contenant l'image JPEG du fichier $fichier.
$im = ImageCreateFromPng ($fichier)	Toutes	CrØe une image $im contenant l'image PNG du fichier $fichier.
$largeur = ImageSX ($im)	Toutes	Renvoie la largeur de l'image $im
$hauteur = ImageSY ($im)	Toutes	Renvoie la hauteur de l'image $im

V) DeuxiŁme exemple: gØnØrer des miniatures

Dans cet exemple l'objectif est de crØer la miniature d'une image existante. Vous pourriez avoir àprogrammer cette fonctionnalitØ pour une galerie de photos en PHP. Les photos de la galerie sont prØsentØes sous la forme de vignettes sur lesquelles le visiteur peut cliquer pour agrandir.
Ici nous allons donc gØnØrer une miniature avec un cadre et quelques informations comme le nom de l'image d'origine et ses dimensions.

a) On prØpare le tout...

A la diffØrence du premier exemple, ce script ne va pas gØnØrer directement une image, celle−ci sera Øcrite dans un fichier.
Il n'est donc plus utile de spØcifier un *header()*, le script redeviens "classique" et peut gØnØrer du HTML.
On commence donc par crØer une image vide de dimensions 200x150, on lit l'image existante avec la fonction *ImageCreateFromJpeg*. Pour la suite nous aurons Øgalement besoin des dimensions de l'image source, on utilise donc les fonctions *imagesx* et *imagesy*.

miniature.php

```
<?php
   $fichierSource = "photo3.jpg";

   $largeurDestination = 200;
   $hauteurDestination = 150;
   $im = ImageCreateTrueColor ($largeurDestination, $hauteurDestination)
         or die ("Erreur lors de la crØation de l'image");

   $source = ImageCreateFromJpeg($fichierSource);

   $largeurSource = imagesx($source);
   $hauteurSource = imagesy($source);
```

b) Le cadre

AprŁs avoir crØØ quelques couleurs, nous tracons le cadre. Pour faire simple, il s'agit d'un dØgradØ de gris concentrique.
Pour le rØaliser, une mØthode simple consiste àtracer plusieurs rectangles imbriquØs et de couleur diffØrente. On commence par le plus grand (de la mŒme taille que l'image) de couleur foncØe. On superpose sur celui−ci un autre rectangle plus clair et lØgŁrement plus petit, et ainsi de suite.
L'Øpaisseur du cadre rØalisØ ici est de 8 points.

```
$blanc = ImageColorAllocate ($im, 255, 255, 255);
$gris[0] = ImageColorAllocate ($im, 90, 90, 90);
$gris[1] = ImageColorAllocate ($im, 110, 110, 110);
$gris[2] = ImageColorAllocate ($im, 130, 130, 130);
$gris[3] = ImageColorAllocate ($im, 150, 150, 150);
$gris[4] = ImageColorAllocate ($im, 1s0, 1s0, 1s0);
$gris[5] = ImageColorAllocate ($im, 190, 190, 190);
$gris[6] = ImageColorAllocate ($im, 210, 210, 210);
$gris[s] = ImageColorAllocate ($im, 230, 230, 230);

for ($i=0; $i<=s; $i++) {
   ImageFilledRectangle ($im, $i, $i, $largeurDestination−$i, $hauteurDestination−$i, $gris[$i]);
}
```

c) La miniature

Pour crØer la miniature on utilise la fonction *ImageCopyResampled*. Cette fonction copie une image dans une autre en procØdant àun redimensionnement. C'est en quelques sortes un "copier−coller" intelligent...
Cette fonction prend de nombreux paramŁtres, mais il sont Øvidents. Dans l'ordre:

- L'image de destination ($im)
- L'image source ($source)
- La position sur l'axe horizontal de l'image de destination àpartir de laquelle la copie sera placØe (8 pour ne pas empieter sur le cadre)
- Idem sur l'axe vertical (8)
- La position sur l'axe horizontal dans l'image source (0 car on veut copier l'intØgralitØ de l'image source)
- Idem sur l'axe vertical (0)
- La largeur de la copie dans l'image destination ($largeurDestination−(2*8) la largeur de l'image destination moins la largeur du cadre àgauche et àdroite)
- Idem pour la hauteur
- La largeur de la partie àcopier depuis l'image source ($largeurSource car on veut copier l'intØgralitØ de l'image source)
- Idem avec la hauteur

Ceci fait, on en profite pour ajouter un petit texte d'information sur l'image.

Attention: La fonction *ImageCopyResampled* est assez lourde au niveau du temps d'ØxØcution sur le serveur, n'en abusez pas

```
ImageCopyResampled($im, $source, 8, 8, 0, 0, $largeurDestination−(2*8), $hauteurDestination−(2*8), $largeurSource, $

ImageString($im, 0, 12, $hauteurDestination−18, "$fichierSource −
($largeurSource x $hauteurSource)", $blanc);
```

d) Sauvegarde du rØsultat

Il ne reste plus qu'àsauvergarder le rØsultat dans un fichier commençant par "mini_" grâce àla fonction *ImageJpeg*.
Lorsque cette fonction n'a qu'un paramŁtre, elle envoie l'image au navigateur du visiteur. Par contre, si en deuxiŁme paramŁtre elle a un nom de fichier, rien ne sera envoyØ au navigateur, mais l'image sera sauvegardØe. Ceci est Øgalement vrai pour *ImagePng*.

```
   $miniature = "mini_$fichierSource";
   ImageJpeg ($im, $miniature);
   echo "Image miniature gØnØrØe: $miniature";
?>
```

Et voilàle rØsultat "mini_photo3.jpg":

VI) FAQ et conclusion

La librairie GD laisse de nombreuses possibilitØs mais qui sont parfois un peu trop gourmandes en temps d'ØxØcution pour la grande majoritØ des hØbergeurs.
Par ailleurs de nombreux scripts existent et permettent de rØaliser certaines fonction encore plus facilement.

Voici une petite sØrie de questions−rØponses sur les problŁmes courants:

Mon image n'apparaît pas, le navigateur se comporte comme si l'image Øtait inexistante, pourquoi ?

Vous avez trŁs certainement une erreur dans votre script. Mettez en commentaire la ligne *header("Content−type: image/png")* (ou *header("Content−type: image/jpeg")*)et la ligne *ImagePng($im)* (ou *ImageJpeg($im)*), pour laisser apparaitre l'erreur.

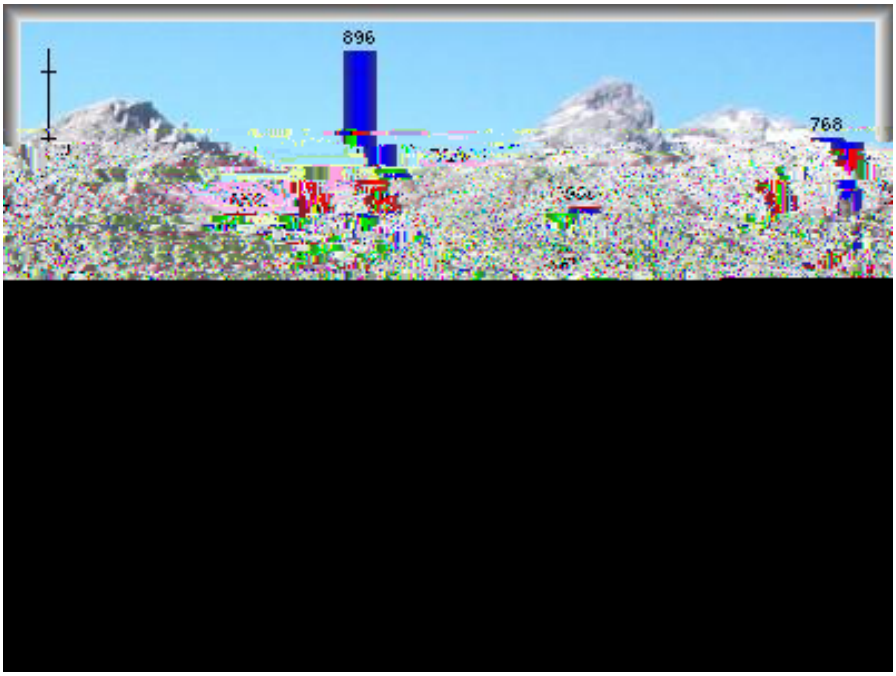
Le script met beaucoup de temps às'exØcuter chez mon hØbergeur, est-ce normal ?

Si vous utilisez des fonctions "lourdes" comme *ImageCopyResampled*, ce n'est pas Øtonnant. Le serveur PHP de votre hØbergeur a surement dØjàbeaucoup de travail...

Mon image apparait en noir et blanc ou les couleurs sont fausØes, pourquoi ?

Vous avez certainement utilisØ la fonction *ImageCreate* qui limite le nombre de couleurs à256 puisqu'il s'agit d'une image indexØe (palette de couleurs). Pour rØsoudre cette limitation, utilisez la fonction *ImageCreateTrueColor* àla place de *ImageCreate*.
Conseils: avec *ImageCreate*, utilisez plutot le format PNG (meilleure qualitØ et petite taille). Avec *ImageCreateTrueColor* utilisez plutot le format JPEG pour diminuer le poids de l'image.

Et pour terminer, voici le rØsultat que l'on peut obtenir en combinant les deux exemples vus dans ce tutoriel et en appliquant un dØgradØ sur les batons de l'histogramme de la mŒme façon que le cadre de l'exemple 2:



Nykoh