Créer des images en PHP

Vous savez quoi ? Il y a des gens qui croient que le PHP c'est fait que pour générer des pages web !
Si si je vous jure ! 

Quoi, vous aussi ? 
Bon remarquez, je peux pas vous en vouloir non plus : tout le long de ce cours, on n'a fait "que" générer des pages HTML avec PHP. Difficile de croire que l'on pourrait faire autre chose...

En fait, à la base, PHP a bien été créé pour réaliser des pages web. Mais, au fur et à mesure, on s'est rendu compte qu'il serait dommage de le limiter à ça. On a donc prévu de pouvoir lui rajouter des "extensions". Ainsi, en rajoutant certains fichiers (des DLL sous Windows), PHP peut alors se mettre à générer des images, ou même des PDF !
Dans ce chapitre, nous allons parler de l'extension spécialisée dans la génération d'images, il s'agit de la librairie GD.

Ne vous y trompez pas : ce que je vais vous apprendre c'est toujours du PHP ! Et vous allez pouvoir faire grâce à ce chapitre des choses vraiment passionnantes, vous pouvez me croire ! 

Activer la librairie GD

On a déjà un problème (ça commence fort)
En effet, la librairie GD (qui vous permet de créer des images) est livrée avec PHP, mais *elle n'est pas activée*. Ca veut dire quoi ? Qu'il va falloir aller modifier un fichier pour pouvoir utiliser GD.

Vous allez donc faire ceci :

1. Vous rendre dans le dossier où Apache (le serveur Web) est installé. Si vous utilisez EasyPHP, rendez-vous dans le dossier où EasyPHP est installé. Vous devriez voir un sous-dossier "apache". C'est là

2. Repérez un fichier appelé "php.ini" et ouvrez-le. C'est là-dedans que se trouvent toutes les options de configuration de PHP. Comme vous pouvez le voir, il y en a beaucoup.

3. Il n'y a qu'une ligne qui nous intéresse. Elle contient le texte :
   `;extension=php_gd2.dll`
   Faites une recherche pour repérer cette ligne, ça ira plus vite.

4. Enlevez le point-virgule qui se trouve devant cette ligne. Vous devriez voir ceci :

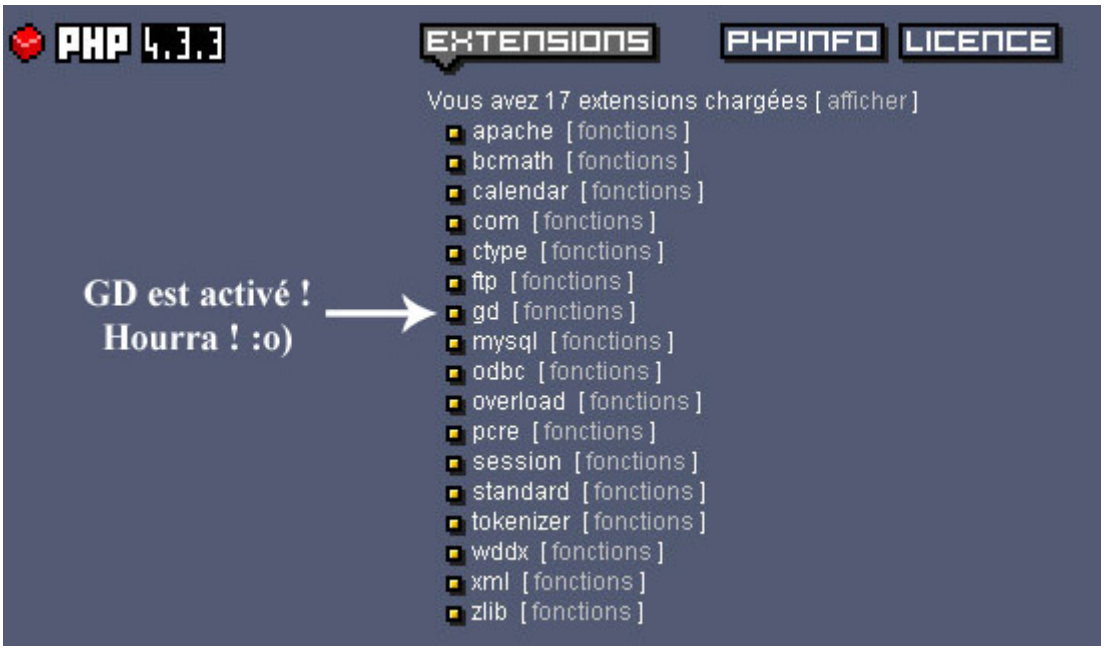
En fait, le point-virgule sert de commentaire. Tant qu'il y a le commentaire, la ligne n'est pas lue et PHP "oublie" d'utiliser GD. Si vous enlevez le commentaire, alors PHP va "charger" la librairie GD, et vous allez pouvoir travailler avec des images !

5. Enregistrez le fichier php.ini

6. Redémarrez EasyPHP.

OUF ! C'est fini ! 😄

Vous allez maintenant pouvoir utiliser GD sur votre ordinateur avec EasyPHP.
Normalement, si vous regardez les extensions chargées en cliquant sur le lien "Afficher", vous devriez avoir "gd" dans la liste :



Et sur Internet avec mon hébergeur ? Est-ce que je peux utiliser GD ?

Ca dépend des hébergeurs. Une grande partie des hébergeurs gratuits désactivent GD parce que ça consomme beaucoup de ressources du processeur.
Si des dizaines de sites se mettent à générer des images en même temps, ça risquerait de faire ramer toute la machine et donc de ralentir tous les autres sites

Ne désespérez pas pour autant, il existe certainement des hébergeurs gratuits qui acceptent la librairie GD... Sinon, il faudra peut-être trouver un hébergement payant (on peut en trouver des pas chers qui ont activé GD !).

Les bases de la création d'image

Vous avez réussi à surmonter le premier problème ?
Bravo, c'était le plus difficile

Voici le plan que nous allons suivre pour créer une image :

1. On va voir ce que c'est un header.
2. Ensuite, on va créer l'image de base.
3. Enfin, on verra comment on affiche l'image quand on a fini.

Y'a du boulot

Le header

Il y a 2 façons de générer une image en PHP :

- Soit on fait en sorte que notre script PHP renvoie une image (au lieu d'une page web comme on avait l'habitude). Dans ce cas, si on va sur la page http://www.monsite.com/testgd.php, ça affichera une image et non pas une page web !
- Soit on demande à PHP d'enregistrer l'image dans un fichier.

Dans les 2 cas, on utilisera exactement les mêmes fonctions.
On va commencer par la première façon de générer l'image, c'est-à-dire qu'on va faire en sorte que notre script "renvoie" une image au lieu d'une page web.

Mais comment faire pour que le navigateur sache que c'est une image et non pas une page HTML qu'il doit afficher ?

Il va falloir envoyer ce qu'on appelle un **header** (en-tête). Grâce à la fonction *header*, on va "dire" au navigateur que l'on est en train d'envoyer une image.
Je vous rappelle les types d'images les plus courants sur le web :

- JPEG : c'est un format très adapté pour les photos par exemple, car on peut utiliser beaucoup de couleurs.

- PNG : c'est le format le plus récent, très adapté dans la plupart des cas. En fait, à moins d'avoir affaire à une photo, le mieux est d'utiliser le PNG.
  Le PNG est en quelque sorte le "remplaçant" du format GIF.

Donc pour faire simple : si c'est une photo, vous faites un JPEG, sinon dans tous les autres cas vous faites un PNG 

Voici le code PHP qu'il faut mettre pour "annoncer" au navigateur que l'on va renvoyer une image PNG :

Source 4.2.1 : un header

```
<?
header ("Content-type: image/png");
?>
```

Voilà, c'est assez simple. Ce code signifiera pour le navigateur que l'on envoit une image PNG, et non pas une page HTML.
Si vous envoyez un JPEG, c'est presque pareil, mais vous remplacez le "png" par "jpeg".

La fonction header est particulière. Comme *setcookie*, elle doit être utilisée avant d'avoir écrit le moindre code HTML.
En clair, mettez cette ligne tout au début de votre code, et vous n'aurez pas de problèmes.

Créer l'image de base

Il faut savoir qu'il y a 2 façons de créer une image : soit vous créez une nouvelle image vide, soit vous chargez une image qui existe déjà et qui servira de fond à votre nouvelle image.

- On va commencer par créer une image vide.

  Pour créer une image vide en PHP, on utilise la fonction *imagecreate* (facile à retenir ça va ).
  Cette fonction est simple. Elle prend 2 paramètres : la largeur et la hauteur de l'image que vous voulez créer. Elle renvoit une information, que vous devez mettre dans une variable (par exemple $image).

  Ce qui nous donne :

Source 4.2.2 : création de l'image vide

```
<?
header ("Content-type: image/png");

$image = imagecreate (200,50);
?>
```

Ici, nous sommes en train de créer une image de **200 pixels de large** et **50 pixels de haut**.

$image ne contient ni un nombre, ni du texte. Cette variable contient une "image". C'est assez difficile à imaginer qu'une variable puisse "contenir" une image, mais c'est comme ça j'y peux rien

On dit que $image est une "ressource". Une ressource est une variable un peu spéciale qui contient toutes les informations sur un objet. Ici, il s'agit d'une image, mais il pourrait très bien s'agir d'un PDF ou même d'un fichier que vous avez ouvert avec *fopen*. Tiens tiens, ça vous rappelle quelque chose ?

- Maintenant l'autre possibilité : créer une image à partir d'une image déjà existante.

Cette fois, il y a 2 fonctions à connaître. Laquelle choisir ? Ca dépend du type de l'image que vous voulez charger :

  - JPEG : il faut utiliser la fonction *imagecreatefromjpeg*.

- PNG : il faut utiliser la fonction *imagecreatefrompng*.

Par exemple, j'ai une jolie photo de coucher de soleil qui s'appelle couchersoleil.jpg :

Pour créer une nouvelle image en se basant sur celle-là, je dois utiliser la fonction *imagecreatefromjpeg*. Ca nous donnerait le code suivant :

Source 4.2.3 : création de l'image à partir d'une autre image

```
<?
header ("Content-type:
image/jpeg");

$image = imagecreatefromjpeg
("couchersoleil.jpg");
?>
```

Voilà, vous savez créer une nouvelle image.
Nous allons maintenant voir comment afficher cette image que vous venez de créer.

Quand on a terminé : on affiche l'image

Une fois que vous avez chargé l'image, vous vous amusez à écire du texte dedans, à faire des cercles, des carrés etc... Ca, nous allons l'apprendre juste après.
Là, je vais vous montrer comment on fait pour dire à PHP qu'on a fini et qu'on veut afficher l'image.

La fonction à utiliser dépend du type de l'image que vous êtes en train de créer :

- JPEG : il faut utiliser la fonction *imagejpeg*.
- PNG : il faut utiliser la fonction *imagepng*.

Ces 2 fonctions marchent de la même manière : vous avez juste besoin d'indiquer quelle est l'image que vous voulez afficher.

Il faut savoir qu'il y a 2 façons d'utiliser les images en PHP : vous pouvez les afficher directement après les avoir créées, ou vous pouvez les enregistrer sur le disque pour pouvoir les réafficher plus tard sans avoir à refaire tous les calculs.

- Afficher directement l'image : c'est la méthode que l'on va utiliser dans la plupart de ce chapitre. Quand la page PHP est exécutée, elle vous affiche l'image que vous lui avez demandé de créer.

Vous avez toujours votre variable $image sous la main ? Parfait

Alors voici le code complet que j'utilise pour créer une nouvelle image PNG de taille 200x50 et l'afficher directement :

Source 4.2.4 : afficher l'image

```
<?
header ("Content-type: image/png"); // 1 : on indique qu'on va envoyer une image PNG

$image = imagecreate(200,50); // 2 : on crée une nouvelle image de taille 200x50

// 3 : on fait joujou avec notre image (on va apprendre à le faire)

imagepng($image); // 4 : on a terminé de faire joujou, on demande à afficher l'image
?>
```

C'est bien joli, mais là on n'a qu'une image sous les yeux. Et si je veux mettre du texte autour ? Les menus de mon site ?

En fait, on utilise une technique qui, j'en suis sûr, va pas mal vous surprendre. On va demander à afficher la page PHP comme une image.
Donc, si la page PHP s'appelle "image.php", vous mettrez ce code HTML pour l'afficher depuis une autre page :

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

Incredible, isn't it ?

Mais en fait, c'est logique quand on y pense ! La page PHP que l'on vient de créer EST une image (parce qu'on a modifié le header). On peut donc afficher l'image que l'on vient de créer

depuis n'importe quelle page de votre site en utilisant simplement la balise <img /> 

Le gros avantage de cette technique, c'est que l'image affichée pourra changer à chaque fois !

- Enregistrer l'image sur le disque : si, au lieu d'afficher directement l'image, vous préférez l'enregistrer sur le disque, alors il faut ajouter un paramètre à la fonction *imagepng* : le nom de l'image et éventuellement son dossier.
  Par contre, dans ce cas, votre script PHP ne va plus renvoyer une image (il va juste en enregistrer une sur le disque). Vous pouvez donc supprimer la fonction *header* qui ne sert plus à rien.
  Ce qui nous donne :

Source 4.2.5 : enregistrer l'image

```
<?
$image = imagecreate(200,50);

// on fait joujou avec notre image

imagepng($image, "images/monimage.png"); // on enregistre l'image dans le dossier "images"
?>
```

- Cette fois, l'image a été enregistrée sur le disque avec le nom "monimage.png". Pour l'afficher depuis une autre page web, vous ferez donc comme ceci :
  `<img src="images/monimage.png" />`

- Ca, vous avez un peu plus l'habitude j'imagine 
  Cette technique a l'avantage de ne pas nécessiter de recalculer

l'image à chaque fois (votre serveur aura moins de travail), mais le défaut c'est qu'une fois qu'elle est enregistrée, l'image ne change plus.
Vous allez comprendre l'intérêt de cette technique plus loin dans le chapitre.

Mais... Mais ??? Si je teste ces codes, ça crée une image toute blanche ! C'est nul, il s'est rien passé de bien !

Oui, je sais. Vous avez été patients et c'est bien parce que c'est maintenant que ça va devenir intéressant.

Allez donc chercher votre baguette magique, je vous attends 

Texte et couleur

C'est bon, vous avez votre baguette magique ?
Alors voici ce que nous allons apprendre à faire maintenant :

- Manipuler les couleurs
- Ecrire du texte

Vous allez commencer à voir un peu ce qu'il est possible de faire grâce à la librairie GD, mais vous verrez plus loin qu'on peut faire bien plus

Manipuler les couleurs

Un ordinateur, il faut le savoir, décompose chaque couleur en Rouge-Vert-Bleu. En mélangeant les quantités de rouge, vert et bleu, ça nous donne une couleur parmi les millions de possibilités !

On indique la "quantité" de rouge, vert et bleu par un nombre compris entre 0 et 255.

- Par exemple, si je dis que je mets 255 de bleu, ça veut dire qu'on met tout le bleu.
- Si je mets 100 de bleu, bah il y a un peu moins de bleu.

- Si je mets 0, alors là y'a plus du tout de bleu

On doit écrire les 3 quantités dans l'ordre RVB (Rouge Vert Bleu). Par exemple :

```
255 0 0
```

Ca, c'est une couleur qui contient plein de rouge, et pas du tout de vert ni de bleu. C'est donc la couleur... rouge ! Bravo !

Maintenant, si je mets plein de rouge et de vert :

```
255 255 0
```

Ca nous donne la couleur : jaune !

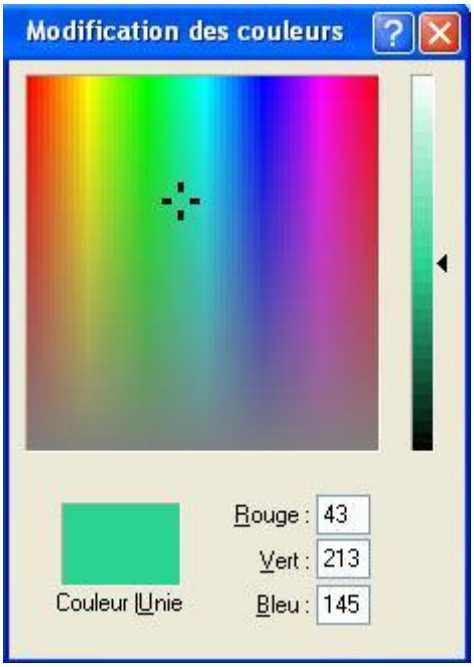
Allez un dernier essai pour la route et on arrête là :

```
255 128 0
```

Ca, c'est la couleur orange !

Pour info, la couleur blanche correspond à (255 255 255), et la couleur noire à (0 0 0).

Si vous avez un logiciel de dessin comme Paint et que vous allez dans le menu Couleur / Modifier les couleurs, vous pouvez choisir la couleur que vous voulez :



Comme vous pouvez le voir, en cliquant sur la couleur qui vous intéresse on vous donne les quantités de Rouge Vert Bleu.
Vous pouvez donc choisir la couleur que vous voulez. Allez-y, servez-vous

Mais revenons à ce qui nous intéresse : PHP (c'est bien pour ça que vous êtes là non ?)
Pour définir une couleur en PHP, on doit utiliser la fonction : *imagecolorallocate*.

On lui donne 4 paramètres : l'image sur laquelle on travaille, la quantité de rouge, la quantité de vert, et la quantité de bleu.
Cette fonction nous renvoit la couleur dans une variable. Grâce à cette fonction, on va pouvoir se créer plein de "variables-couleur" qui vont nous être utiles pour indiquer la couleur ensuite.

Voici quelques exemples de création de couleur :

Source 4.2.6 : préparer des couleurs

```
<?
header ("Content-type:
image/png");
$image = imagecreate(200,50);

$orange = imagecolorallocate
($image, 255, 128, 0);
$bleu = imagecolorallocate($image,
0, 0, 255);
$bleuclair = imagecolorallocate
($image, 156, 227, 254);
$noir = imagecolorallocate($image,
0, 0, 0);
$blanc = imagecolorallocate
($image, 255, 255, 255);


imagepng($image);
?>
```

Et voilà, on s'est préparés plein de couleurs qui vont beaucoup nous servir ensuite !

Une chose très importante à noter : la première fois que vous faites un *imagecolorallocate*, cette couleur devient la couleur de fond de votre image.
Donc, si vous avez bien compris, ce code doit créer une image... toute orange ! Essayez !

Essayer !

Si j'avais voulu que le fond soit blanc et pas orange, il aurait fallu mettre la ligne "$blanc..." en premier.

Voilà, vous savez maintenant créer toutes les couleurs de l'arc-en-ciel en PHP (et même plus)

Ecrire du texte

Nous voici enfin dans le vif du sujet (ouf !).
Nous avons une belle image avec un maaagnifique fond orange, et nous voulons écrire du texte dedans.
Avec la fonction *imagestring*, c'est facile !

Cette fonction prend pas mal de paramètres. Elle s'utilise comme suit :

```
imagestring($image, $police, $x, $y, $texte_a_ecrire, $couleur);
```

Il existe aussi la fonction *imagestringup* qui fonctionne exactement pareil, sauf qu'elle écrit le texte verticalement au lieu d'horizontalement !

Je vous détaille les paramètres dans l'ordre, c'est important que vous compreniez bien :

- $image : c'est notre fameuse variable qui contient l'image.

- $police : c'est la police de caractères que vous voulez utiliser. Vous devez mettre un nombre de 1 à 5 : 1 = petit, 5 = grand. Il est aussi possible d'utiliser une police de caractère personnalisée, mais il faut avoir des polices dans un format spécial qu'il serait trop long de détailler ici. On va donc se contenter des polices par défaut

- $x et $y : ce sont les coordonnées où vous voulez placer votre texte sur l'image.
  Et là vous vous dites : "Aïe, ça sent les maths " (comme quoi les maths ça sert)

Vous devez savoir que l'origine se trouve en haut à gauche de votre image. Le point de coordonnées 0, 0 représente donc le point tout en haut à gauche de l'image.
Voici le schéma de notre image orange de tout à l'heure, qui est de taille 200x50 :

Comme vous pouvez le voir, j'ai marqué en bleu les 4 points des côtés de l'image. 0, 0 se trouve tout en haut à gauche, et 200, 50 se trouve tout en bas à droite.
Si vous avez juste un peu l'habitude des maths, ça ne devrait pas vous poser de problème.
Sinon, vous me ferez le plaisir de réouvrir votre livre de géométrie page 125

- $texte_a_ecrire, c'est le... texte que vous voulez écrire. Non non, y'a pas de piège

- $couleur, c'est une couleur que vous avez créé tout à l'heure avec *imagecolorallocate*.

Voici un exemple concret de ce qu'on peut faire :

Source 4.2.7 : écrire du texte

```php
<?
header ("Content-type: image/png");
$image = imagecreate(200,50);

$orange = imagecolorallocate($image,
255, 128, 0);
$bleu = imagecolorallocate($image, 0,
0, 255);
$bleuclair = imagecolorallocate
($image, 156, 227, 254);
$noir = imagecolorallocate($image, 0,
0, 0);
$blanc = imagecolorallocate($image,
255, 255, 255);

imagestring($image, 4, 35, 15, "Salut
les Zér0s !", $blanc);

imagepng($image);
?>
```

Essayer !

La ligne avec imagestring peut se traduire par : *Mets dans l'image $image, avec la police de taille 4, aux coordonnées (35, 15), le texte "Salut les Zér0s !", de couleur blanche.*

Bien entendu, vous me direz qu'avec un bon Photoshop (ou même Paint), on peut faire pareil et c'est moins compliqué.
Oui c'est vrai, mais l'avantage c'est qu'on est en PHP là ! On peut donc faire varier le texte à afficher.

Un exemple ? Je souhaite afficher l'heure qu'il est, mais sur un fond différent selon qu'il fait jour ou qu'il fait nuit :

- S'il est entre 8h et 20h, j'affiche l'heure sur un fond bleu.

- S'il est entre 20h et 8h du matin, alors j'affiche l'heure sur un fond noir.

Source 4.2.8 : l'heure en couleurs

```php
<?
header ("Content-type: image/png");
$image = imagecreate(200,50);

if (date("H") > 8 AND date("H") < 20) // Il
fait jour
{
    $fond = imagecolorallocate($image, 143,
190, 241); // Fond bleu clair
    $couleur_texte = imagecolorallocate($image,
0, 255, 0); // Texte en vert
}
else // Il fait nuit
{
    $fond = imagecolorallocate($image, 0, 0,
0); // Fond noir
    $couleur_texte = imagecolorallocate($image,
255, 255, 255); // Texte en blanc
}

$heure = 'Il est ' . date('H\h i'); // On
stocke l'heure et les minutes dans une variable

imagestring($image, 5, 40, 15, $heure,
$couleur_texte); // On affiche l'heure dans la
bonne couleur

imagepng($image);
?>
```

Essayer !

C'est assez simple : je teste l'heure pour voir s'il fait jour ou pas, et en fonction de ça j'utilise des couleurs différentes. Je vous rappelle que le premier *imagecolorallocate* définit la couleur de fond de l'image. Je stocke la couleur du texte dans une variable pour m'en resservir

plus loin : ça me permettra d'afficher le texte dans la bonne couleur, selon qu'il fait jour ou qu'il fait nuit.

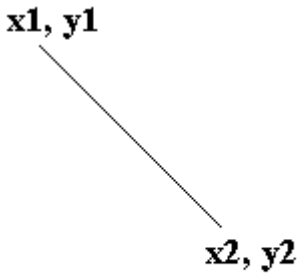
Pour vérifier si le changement de couleur fonctionne bien (et que je ne bluffe pas), revenez tester cet exemple à un autre moment de la journée

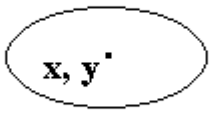
Vous pourriez améliorer ce script en chargeant une image de fond différente selon qu'il fait jour ou qu'il fait nuit. S'il fait jour vous chargez une photo de soleil en fond, et s'il fait nuit vous chargez une photo de Lune.
Vous allez voir que ça a de suite plus de classe !

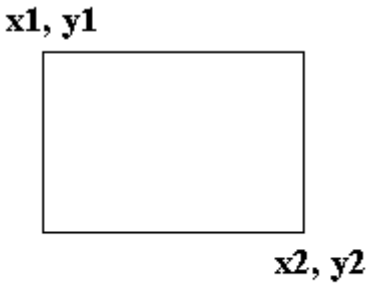
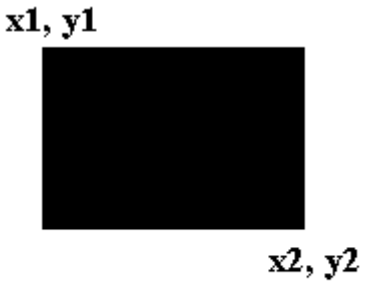
Dessiner une forme

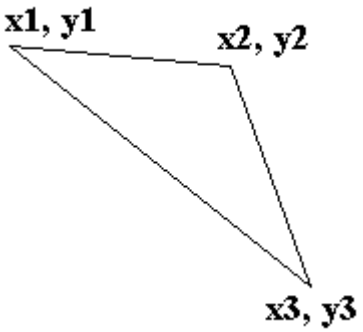
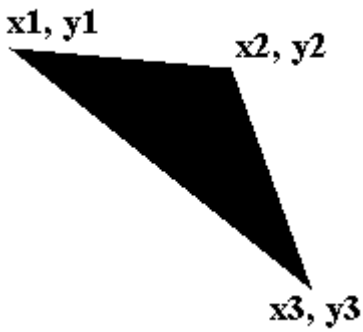
Dessiner du texte c'est bien, mais ça serait bête si on était limités à ça. Heureusement, PHP a pensé à tout !
Graphistes en herbe, vous allez certainement trouver votre bonheur dans toutes ces fonctions : vous pouvez créer des lignes, des rectangles, des cercles, des polygones...

Je vais vous présenter la plupart de ces fonctions ci-dessous, et je vous montrerai ce que ça donne dans une image de taille 200x200, histoire d'avoir un aperçu 

Fonction	Description	Aperçu
`ImageSetPixel ($image, $x, $y, $couleur);`	Dessine un pixel aux coordonnées (x, y)	 `ImageSetPixel ($image, 100, 100, $noir);`
`ImageLine ($image, $x1, $y1, $x2, $y2, $couleur);`	Dessine une ligne entre 2 points de coordonnées (x1, y1) et (x2, y2)	 `ImageLine ($image, 30, 30, 120, 120, $noir);`

`ImageEllipse ($image, $x, $y, $largeur, $hauteur, $couleur);`	Dessine une ellipse, dont le centre est aux coordonnées (x, y), de largeur $largeur et de hauteur $hauteur.	 `ImageEllipse ($image, 100, 100, 100, 50, $noir);`
`ImageFilledEllipse ($image, $x, $y, $largeur, $hauteur, $couleur);`	Pareil que ImageEllipse, sauf que l'ellipse est entièrement coloriée dans la couleur que vous avez demandée.	 `ImageFilledEllipse ($image, 100, 100, 100, 50, $noir);`

`ImageRectangle ($image, $x1, $y1, $x2, $y2, $couleur);`	Dessine un rectangle, dont le coin en haut à gauche est de coordonnées (x1, y1) et celui en bas à droite (x2, y2)	 `ImageRectangle ($image, 30, 30, 160, 120, $noir);`
`ImageFilledRectangle ($image, $x1, $y1, $x2, $y2, $couleur);`	Pareil que ImageRectangle, sauf que le rectangle est cette fois entièrement colorié.	 `ImageFilledRectangle ($image, 30, 30, 160, 120, $noir);`

`ImagePolygon ($image, $array_points, $nombre_de_points, $couleur);`	Dessine un polygône ayant un nombre de points égal à $nombre_de_points (s'il y a 3 points, c'est donc un triangle). L'array $array_points contient les coordonnées de tous les points du polygone dans l'ordre : x1, y1, x2, y2, x3, y3, x4, y4...	 `$points = array (10, 40, 120, 50, 160, 160); ImagePolygon ($image, $points, 3, $noir);`
`ImageFilledPolygon ($image, $array_points, $nombre_de_points, $couleur);`	Pareil que ImagePolygon, mais cette fois le polygône est colorié à l'intérieur.	 `$points = array (10, 40, 120, 50, 160, 160); ImageFilledPolygon ($image, $points, 3, $noir);`

On peut aussi dessiner des lignes plus épaisses. Pour cela, il faut utiliser la fonction *ImageSetThickness*. On doit préciser l'image concernée et l'épaisseur voulue (en pixels) :

```
ImageSetThickness ($image, $epaisseur);
```

Lorsque vous changez l'épaisseur, toutes les formes que vous dessinez après gardent cette épaisseur. Pour revenir à l'épaisseur initiale (1 pixel), il faut donc refaire appel à *ImageSetThickness* en demandant une épaisseur de 1.

Voilà, c'est pas bien compliqué pourvu qu'on sache bien manier les coordonnées des pixels 

Des fonctions encore plus puissantes

Des rectangles, des ellipses, des lignes... Ouais bof. C'est tout ce qu'on peut faire ?

Bien sûr que non ! Il y a d'autres fonctions que je veux absoluement vous montrer parce qu'elles permettent de faire de très belles choses facilement !
Nous allons apprendre à :

- Rendre une image transparente
- Mélanger deux images
- Redimensionner une image, pour créer une miniature par exemple.

J'espère que vous êtes encore en forme, ça serait dommage de s'endormir sur les fonctions les plus intéressantes 

Rendre une image transparente

Tout d'abord, il faut savoir que seul le PNG peut être rendu transparent. En effet, un des gros défauts du JPEG est qu'il ne supporte pas la transparence.
Nous allons donc ici travailler sur un PNG.

Rendre une image transparente est d'une facilité déconcertante
Il suffit d'utiliser la fonction *imagecolortransparent* et de lui indiquer quelle est la couleur que l'on veut rendre transparente. Cette fonction s'utilise comme ceci :

```
imagecolortransparent($image, $couleur);
```

Je vais reprendre l'exemple de l'image où j'ai écrit "Salut les Zér0s !" sur un vieux fond orange, et je vais y rajouter la fonction *imagecolortransparent* pour rendre ce fond transparent :

Source 4.2.9 : un fond transparent

```php
<?
header ("Content-type: image/png");
$image = imagecreate(200,50);

$orange = imagecolorallocate($image, 255, 128, 0); // Le fond est orange (car c'est la première couleur)
$bleu = imagecolorallocate($image, 0, 0, 255);
$bleuclair = imagecolorallocate($image, 156, 227, 254);
$noir = imagecolorallocate($image, 0, 0, 0);
$blanc = imagecolorallocate($image, 255, 255, 255);

imagestring($image, 4, 35, 15, "Salut les Zér0s !", $noir);
imagecolortransparent($image, $orange); // On rend le fond orange transparent

imagepng($image);
?>
```

Et voilà le PNG transparent que ça nous donne :

Sympa, non ?

Mélanger deux images

Ca, c'est un tout petit peu plus compliqué que de rendre une image transparente, mais bon je vous rassure c'est loin d'être insurmontable quand même et ça en vaut la peine 🙂
La fonction que je vais vous présenter permet de "fusionner" deux images en jouant sur un effet de transparence. Ca a l'air tordu comme ça, mais c'est en fait quelque chose de vraiment génial !

On peut s'en servir par exemple pour afficher le logo de son site sur une image.
Voici le logo :

Et voici l'image en question :

La fonction qui permet de réaliser la fusion entre 2 images est : *imagecopymerge*.
Ce script est un peu plus gros que les autres, alors je préfère vous le donner tout de suite. Je vous expliquerai juste après comment il fonctionne.

Source 4.2.10 : fusion d'images

```php
<?
header ("Content-type: image/jpeg"); // L'image que l'on va créer est un jpeg

// On charge d'abord les images
$source = imagecreatefrompng("logosdz.png"); // Le logo est la source
$destination = imagecreatefromjpeg ("couchersoleil.jpg"); // La photo est la destination

// Les fonctions imagesx et imagesy renvoient la largeur et la hauteur d'une image
$largeur_source = imagesx($source);
$hauteur_source = imagesy($source);
$largeur_destination = imagesx($destination);
$hauteur_destination = imagesy($destination);

// On veut placer le logo en bas à droite, on calcule les coordonnées où on doit placer le logo sur la photo
$destination_x = $largeur_destination - $largeur_source;
$destination_y =  $hauteur_destination - $hauteur_source;

// On met le logo (source) dans l'image de destination (la photo)
imagecopymerge($destination, $source, $destination_x, $destination_y, 0, 0, $largeur_source, $hauteur_source, 60);

// On affiche l'image de destination qui a été fusionnée avec le logo
imagejpeg($destination);
?>
```

Voici le zôôli résultat que donne ce script :

C'est là normalement qu'on dit : "Ouah trop puissant !"
En effet, *imagecopymerge* c'est une fonction vraiment sympa, parce que maintenant vous allez pouvoir "copyrighter" automatiquement toutes les images de votre site si vous le voulez

Cependant, le script utilisé ici est un petit peu plus complexe, et je crois que quelques explications ne seraient pas de refus
Voici donc les points à bien comprendre :

- Dans ce script, on manipule 2 images : $source (le logo) et $destination (la photo). Les deux sont créées à l'aide de la fonction *imagecreatefrompng* (et fromjpeg pour la photo).

- Il y a ensuite toute une série de calculs à partir des coordonnées et de la largeur et hauteur des images. J'imagine que ça a dû vous faire peur, mais c'est en fait très simple du temps qu'on sait faire une soustraction
  Notre but est de savoir à quelles coordonnées placer le logo sur la photo. Moi, je veux le mettre tout en bas à droite. Pour ça, j'ai besoin de connaître la dimension des images. J'utilise les

fonctions *imagesx* et *imagesy* pour récupérer les dimensions du logo et de la photo.
Ensuite, pour placer le logo tout en bas, il faut le mettre à la position $hauteur_de_la_photo - $hauteur_du_logo. On fait de même pour placer le logo à droite : $largeur_de_la_photo - $largeur_du_logo.
Si j'avais voulu mettre le logo tout en haut à gauche, là ça aurait été beaucoup plus simple : pas besoin de faire de calculs, vu qu'en haut à gauche c'est les coordonnées (0, 0) ! 

- Vient ensuite la fonction *imagecopymerge*, la plus importante. Elle prend tout plein de paramètres. Ce qu'il faut savoir, c'est qu'elle a besoin de 2 images : une source et une destination. Elle modifie l'image de destination (ici la photo) pour y intégrer l'image source. Cela explique pourquoi c'est $destination que l'on affiche à la fin, et non pas $source (le logo) qui n'a pas changé.
  Les paramètres à donner à la fonction sont, dans l'ordre :

  1. L'image de destination : ici $destination, la photo. C'est l'image qui va être modifiée et dans laquelle on va mettre notre logo.

  2. L'image source : ici $source, c'est notre logo. Cette image n'est pas modifiée.

  3. L'abscisse où vous désirez placer le logo sur la photo : il s'agit ici de l'abscisse du point située à la position largeur_de_la_photo - $largeur_du_logo

  4. L'ordonnée où vous désirez placer le logo sur la photo : de même, il s'agit de l'ordonnée du point sur la photo (ici $hauteur_de_la_photo - $hauteur_du_logo).

5. L'abscisse de la source : en fait, la fonction *imagecopymerge* permet aussi de ne prendre qu'une partie de l'image source. Ca peut devenir un peu compliqué, alors nous on va dire qu'on prend tout le logo. On part donc du point situé aux coordonnées (0, 0) de la source. Mettez donc 0 pour l'abscisse.

6. L'ordonnée de la source : de même pour l'ordonnée. Mettez 0.

7. La largeur de la source : c'est la largeur qui détermine quelle partie de l'image source vous allez prendre. Nous on prend toute l'image source, donc vous prenez pas la tête non plus et mettez $largeur_source.

8. La hauteur de la source : de même, mettez $hauteur_source.

9. Le pourcentage de transparence : c'est un nombre entre 0 et 100 qui indique la transparence de votre logo sur la photo. Si vous mettez 0, le logo sera invisible (totalement transparent) et si vous mettez 100 il sera totalement opaque (il n'y aura pas de joli effet de "fusion"). Mettez un nombre autour de 60-70, en général c'est pas mal 

Concrètement, on peut se servir de ce code pour faire une page "copyrighter.php". Cette page prendra un paramètre : le nom de l'image à copyrighter.
Par exemple, si vous voulez copyrighter automatiquement "tropiques.jpg", vous afficherez l'image comme ceci :

```
<img
src="copyrighter.php?image=tropiques.jpg" />
```

A vous maintenant d'écrire la page copyrighter.php
Si vous vous basez sur le script que je vous ai donné, ça ne devrait pas être bien long. Il faut juste récupérer le nom de l'image à charger (via la variable $_GET['image']). Arf, ça y est je vous ai tout dit

Redimensionner une image

C'est une des fonctionnalités les plus intéressantes de la librairie GD à mon goût. Ca permet de créer des miniatures de nos images.
Vous pouvez vous en servir par exemple pour faire une galerie de photos. Vous affichez les miniatures et si vous cliquez sur l'une d'elles, ça l'affiche dans sa taille originale.

Pour redimensionner une image, on va utiliser la fonction *imagecopyresampled*. C'est une des fonctions les plus poussées car elle fait beaucoup de calculs mathématiques pour créer une miniature de bonne qualité. Le résultat est très bon, mais cela donne énormément de travail au processeur.

> Cette fonction est donc puissante mais lente. Tellement lente que certains hébergeurs désactivent la fonction pour éviter que le serveur ne rame.
> Il serait suicidaire d'afficher directement l'image à chaque chargement d'une page. Nous allons donc créer la miniature une fois pour toutes et l'enregistrer dans un fichier.

Nous allons donc enregistrer notre miniature dans un fichier (par exemple "mini_couchersoleil.jpg"). Cela veut dire qu'on peut déjà virer la première ligne (le header) qui ne sert plus à rien.

Comme pour *imagecopymerge*, on va avoir besoin de 2 images : la source et la destination. Ici, la source c'est l'image originale, et la destination c'est l'image miniature que l'on va créer.

La première chose à faire sera donc de créer une nouvelle image vide... Avec quelle fonction ? *imagecreate* ? Oui, c'est presque la bonne réponse.
Le problème voyez-vous, c'est que *imagecreate* crée une nouvelle image dont le nombre de couleurs est limité (256 couleurs maximum en général). Or, notre miniature contiendra peut-être plus de couleurs que l'image originale à cause des calculs mathématiques.
On va donc devoir utiliser une autre fonction dont je ne vous ai pas encore parlé : *imagecreatetruecolor*. Elle fonctionne de la même manière que *imagecreate*, mais cette fois l'image pourra contenir beaucoup plus de couleurs 

Voici le code que je vais utiliser pour générer la miniature de ma photo "couchersoleil.jpg" :

Source 4.2.11 : créer une miniature

```
<?
$source = imagecreatefromjpeg
("couchersoleil.jpg"); // La photo est la
source
$destination = imagecreatetruecolor(200, 150);
// On crée la miniature vide

// Les fonctions imagesx et imagesy renvoient
la largeur et la hauteur d'une image
$largeur_source = imagesx($source);
$hauteur_source = imagesy($source);
$largeur_destination = imagesx($destination);
$hauteur_destination = imagesy($destination);

// On crée la miniature
imagecopyresampled($destination, $source, 0, 0,
0, 0, $largeur_destination,
$hauteur_destination, $largeur_source,
$hauteur_source);

// On enregistre la miniature sous le nom
"mini_couchersoleil.jpg"
imagejpeg($destination,
'mini_couchersoleil.jpg');
?>
```

Avant on avait ça :

Et grâce à *imagecopyresampled*, on a obtenu ça :

Je sais pas ce que vous en pensez, mais moi je trouve ça très efficace
Vous pouvez afficher ensuite l'image avec le code HTML :
```
<img src="mini_couchersoleil.jpg" alt="Coucher de soleil" />
```

Bon comment ça marche ?
On crée notre miniature vide avec *imagecreatetruecolor* en dimension réduite (200 x 150).
Je vous ai déjà expliqué les fonctions *imagesx* et *imagesy*, je ne reviens pas dessus. Voyons plutôt quels sont les paramètres de la fonction *imagecopyresampled* :

1. L'image de destination : c'est $destination, l'image qu'on a créé avec *imagecreatetruecolor*.

2. L'image source : l'image dont on veut créer la miniature, ici c'est notre couchersoleil.jpg qu'on a chargé avec *imagecreatefromjpeg*.

3. L'abscisse du point où vous placez la miniature sur l'image de destination : pour faire simple, on va dire que notre image de destination contiendra uniquement la miniature. Donc on placera la miniature aux coordonnées (0, 0), ce qui fait qu'il faut mettre 0 à cette valeur.

4. L'ordonnée du point où vous placez la miniature sur l'image de destination : pour les mêmes raisons, mettez 0.

5. L'abscisse du point de la source : ici, on prend toute l'image source et on en fait une miniature. Pour tout prendre, il faut partir du point (0, 0), ce qui fait que là encore on met 0 à cette valeur.

6. L'ordonnée du point de la source : encore 0.

7. La largeur de la miniature : un des paramètres les plus important, qui détermine la taille de la miniature à créer. Dans notre cas notre miniature fait 200 pixels de large. On a stocké ce nombre dans la variable $largeur_destination.

8. La hauteur de la miniature : de même pour la hauteur de la miniature à créer.

9. La largeur de la source : il suffit d'indiquer la taille de notre image source. On a stocké cette valeur dans $largeur_source, donc on la réutilise ici.

10. La hauteur de la source : de même pour la hauteur.

Comme vous pouvez le voir, *imagecopyresampled* permet de faire beaucoup de choses, et en général on ne se servira pas de tout.
Pas mal de paramètres sont à 0, et c'est pas vraiment la peine de chercher à comprendre pourquoi (même si c'est pas bien compliqué).
Basez-vous sur mon exemple pour créer vos miniatures, et le tour sera joué