Superponer, rotar y dar transparencia a imágenes

Colores transparentes

PHP permite crear colores con determinado color de transparencia. Para ello se utiliza la función:

ImageColorAllocateAlpha()
que debe contener dentro del paréntesis los siguientes parámetros (separados por comas):

$im que es el identificador de la imagen que ha sido creada previamente.

R,G,B que son valores numéricos (o variables) que contienen -en una escala de **0** a **255**- los la intensidad luminosa de cada uno de los tres colores primarios (rojo, verde y azul).

trans es un valor numérico (comprendido entre **0** y **127**) que indica el grado de transparencia de la tinta. El valor **0** indica *opacidad total*, mientras que 127 establece la transparencia total de ese color.

En el ejemplo que tienes a la derecha hemos incluido dos escalas de transparencias superpuestas a intervalos del 10% (desde 0 hasta 100%) transformados a la escala 0-127.

El orden de superposición –similar a las capas de otros programas gráficos– se corresponde con el orden de las instrucciones de creación. Los resultados de las últimas funciones se superponen siempre a los obtenidos como consecuencia de la ejecución de las anteriores.

Transparencia en imágenes externas

Mediante la utilización de la función *imagecopymerge* es posible ajustar el grado de transparencia de una imagen externa.

La función *imagecopymerge()* requiere que se incluyan (dentro del paréntesis y separados por comas y por el orden en el que los incluimos) los siguientes parámetros:

$destino que es el identificador de la imagen sobre la que se va a colocar la transparencia. Como es lógico, deberá haber sido creada antes de incluir la función.

$origen es el identificador de la imagen que pretendemos incluir con un determinado grado de transparencia.

Superposición de áreas transparentes

```
<?
/* Creamos una imagen en color verdadero, le aplicamos un color
   de fondo (para evitar el negro por defecto) y creamos un nuevo
   color que utilizaremos para los bordes de rectangulos posteriores*/
Header("Content-type:image/jpeg");
$im_base=imagecreatetruecolor(610,140);
$fondo=imagecolorAllocate($im_base,255,255,200);
$negro=imagecolorAllocate($im_base,0,0,0);
imagefill($im_base,0,0,$fondo);
# definimos las componentes de un nuevo color
$R=255; $G=00; $B=00;
/* vamos a construir una escala de transparencias
   de 0 a 10 que correspondería con valores de transparencia
   de 0% al 100%.
   Crearemos un bucle que dibuje rectangulos rellenos
   con el color definido en la variable trans que irá aplicando
   al color básico los diferentes grados de transparencia
   y le pondremos un contorno negro para encuadrarlos*/

for($i=0;$i<=10;$i++){
   $trans=ImageColorAllocateAlpha($im_base,$R,$G,$B,(int)($i*127/10));
   imagefilledrectangle($im_base, 10+55*$i, 20, 50+55*$i, 80, $trans);
   imagerectangle($im_base, 10+55*$i, 20, 50+55*$i, 80, $negro);
}
#creamos un nuevo color y repetimos el proceso con nuevos rectangulos
#superpuestos a los anteriores y con las transparencias en sentido opuesto
# es decir, de 100 a 0%
$R=0; $G=0; $B=255;
for($i=0;$i<=10;$i++){
  $trans=ImageColorAllocateAlpha($im_base,$R,$G,$B,127-(int)($i*127/10));
  imagefilledrectangle($im_base, 10+55*$i, 60, 50+55*$i, 120, $trans);
  imagerectangle($im_base, 10+55*$i, 60, 50+55*$i, 120, $negro);
}
# visualizamos el resultado
imagejpeg($im_base);
ImageDestroy();
?>
```

Ver ejemplo .jpg	Ver ejemplo .png	Ver ejemplo .gif

Transparencia de imágenes externas

```
<?
# obtener la imagen
$original=$_SERVER['DOCUMENT_ROOT']."/cursophp/images/aviones4.jpg";
# buscar el formato de la imagen mediante su extensión
for($i=strlen($original)-1;$i>0;$i--){
    if (substr($original,$i,1)=="."){
       $tipo=substr($original,$i+1);
       break;
    }
}
# tamaño del original extraido del array devuelto por getimagesize
$tamano=getimagesize($original);
$orig_Ancho = $tamano[0];
$orig_Alto =$tamano[1];
# estableceremos un margen en blanco alrededor de la imagen de 10 pixels
# igual por los cuatro lados
$borde=10;
$Ancho=$orig_Ancho+2*$borde;
$Alto=$orig_Alto+2*$borde;
# creamos la imagen segun el formato original
switch($tipo){
```

X_d e **Y_d** son las coordenadas de un punto situado en la *esquina superior izquierda de la imagen destino* a partir del que queremos que se *impresione la nueva imagen*. Si queremos una *imagen a sangre* pondremos **0,0** y, si quieres dejar **márgenes**, habrá que poner los anchos de esos márgenes (izquierdo y superior) respectivamente.

X_f e **Y_f** nos servirán para *reencuadrar* la foto original *recortando* por la izquiera y por arriba, respectivamente, los anchos que se indiquen aquí en pixels.

D_x e **D_y** indican el ancho y el alto (por este orden) que va a tener la mancha de imagen en el positivo. Ten en cuenta que *no puedes salirte del papel* así que esos valores sumados con los márgenes (izquierdo y superior) no podrán ser mayores que las dimensiones que has elegido para la *imagen destino*

opacidad es el último de los parámetros de la función al que puede asignársele un valor comprendido entre **0** y **100**.

Representa el porcentaje de opacidad de la imagen superpuesta. Con un valor 100 sería totalmente opaca y con 0 la transparencia sería total.

La función imagecopy

Mediante esta función se puede *copiar* sobre una imagen una parte de otra. Permite extraer porciones de imágenes con su tamaño original sin que permita ampliarlas ni reducirlas. Su sintaxis es la siguiente:

imagecopy($d,$o,$x,$y,$X,$Y,$A,$H) donde:

$d el identificador de la imagen destino, *$o* el identificador de la imagen original, *$x* y *$y* las coordenadas donde se posicionará –en la imagen destino– la esquina superior izquierda de la porción copiada.
$X y *$Y* son los anchos de los *recortes* izquierdo y superior de la imagen a copiar y *$A* y *$H* el ancho y el alto del área de imagen que pretendemos copiar.

Rotación de imágenes

Mediante la funcion: *imagerotate($im,ang,$fondo)* es posible presentar imágenes rotadas.

El parámetro *$im* es el identificador de la imagen a rotar, *ang* es el ángulo de rotación (expresado en **grados** y tomado en *sentido trigonométrico*) y *$fondo* es un color de fondo asociado a la imagen a rotar que puede ser definido mediante la función

```
        $importada=imagecreatefromjpeg($original);
      break;
    case "png":
        $importada=imagecreatefrompng($original);
      break;
    case "gif":
        $importada=imagecreatefromgif($original);
    break;
 }
Header("Content-type:image/jpeg");
# creamos una imagen nueva, un color de fondo y la rellenamos con él
$im_base=imagecreatetruecolor($Ancho,$Alto);
$fondo=imagecolorAllocate($im_base,255,255,200);
imagefill($im_base,0,0,$fondo);
# superponemos la imagen importada posicionandola y aplicandole
# una trasmparencia de 50
imagecopymerge( $im_base, $importada, $borde, $borde ,
                      0, 0, $orig_Ancho, $orig_Alto ,50 );

imagejpeg($im_base);
ImageDestroy();
?>
```

Ver ejemplo .jpg | Ver ejemplo .png | Ver ejemplo .gif

Si observas los resultados obtenidos en el ejemplo en el que intentamos dar transparencia a una imagen en formato **png** podrás observar que *deja bastante que desear* y produce un efecto indeseado por el recorte de las zonas *presuntamente* transparentes.

Esta situación nos obliga a replantear la situación para prever esta circunstancia y recurrir a un *truco* que parece solventar ese problema. La modificación del código fuente es la incluimos aquí debajo.

```
Header("Content-type:image/jpeg");
  $im_base=imagecreatetruecolor($Ancho,$Alto);
  $fondo=imagecolorAllocate($im_base,255,255,200);
  imagefill($im_base,0,0,$fondo);
  /* el truco consiste en crear una segunda imagen (im_truco)cuyas
  dimensiones coincidan con las de la imagen transparente
  que pretendemos colocar. Le asignamos como color el fondo el mismo
  de la imagen destino y hacemos transparente ese color en esa imagen.
         Después hacemos una copia de la imagen original sobre la imagen
         im_truco y sustituimos en la función imagecopymerge la
         imagen original por la obtenida mediante esta chapucilla */
 $im_truco=imagecreatetruecolor($orig_Ancho, $orig_Alto);
 $fondo1=imagecolorAllocate($im_truco,255,0,200);
 imagefill($im_truco,0,0,$fondo1);
 imagecolortransparent ($im_truco,$fondo1);
 imagecopy($im_truco, $importada, 0, 0, 0, 0, $orig_Ancho, $orig_Alto);
imagecopymerge( $im_base , $im_truco, $borde , $borde ,
                        0 , 0 ,$orig_Ancho, $orig_Alto ,60 );
imagejpeg($im_base);
ImageDestroy();
```

Ver el nuevo resultado

La función *imagecolortransparent ($imagen,$color)* permite hacer transparente –en la imagen indicada mediante la variable **$imagen**– el color señalado en la variable **$color**.

La variable **$color** deberá estar definida previamente mediante la función **imagecolorAllocate** u alguna otra que permita identificar un color determinado.

Rotación de imágenes

```
<?
# obtener la imagen
```

imagecolorallocate u otra función que permita asociar colores a imágenes.

Transparencia en la rotación de imágenes

Hemos intentado explorar la posibilidad de lograr imágenes rotadas con fondo transparente. Y la cosa resulta, cuando menos complicada. Cuando se trata de insertarlas sobre un *fondo plano* la situación puede *emularse* fácilmente sin más que asignar como color de fondo de la rotación uno idéntico al del *fondo plano* sobre el que ha de situarse la imagen.

Otra de las alternativas probadas fué tratar de usar *imagecolortransparent* en el color de fondo de rotación de la imagen. Es evidente que ese grado de transparencia solo lo lograríamos con un formato **png** ó **jpg**. Pero... no funciona.

La alternativa siguiente fué tratar de crear una imagen con fondo transparente e insertar en ella la imagen rotada asignándole transparencia a la capa que contiene la imagen rotada. Ahí nos encontramos con algunas dificultades.

La primera de ellas es que *fondo transparente* **sólo lo permiten** las imágenes que son creadas mediante la función **imagecreate**. Si se crean mediante la función **imagecreatetruecolor** esa opción no funciona.

Otra diferencia entre ambas funciones tiene también relación con los colores de fondo. Mientras que en el caso de **imagecreate** se asigna como color de fondo el que se haya definido inmediatamente después de crear la imagen, cuando se trata de **imagecreatetruecolor** se asignará siempre un fondo negro y para cambiar ese color será necesario recurrir a la función **imagefill**.

Pero la *felicidad completa* parece que no existe. Al intentar explorar la primera de estas opciones hemos podido observar que el precio a pagar por la dichosa transparencia es obtener una imagen final de no demasiado buena calidad.

Mejor lo compruebas tu mism@ en el ejemplo que tienes a la derecha.

Transparencia en capas rotadas

Cuando trabajamos con una imagen **truecolor** en la que vamos incluyendo en diferentes capas (mediante copy ó copymerge) otras imágenes –recortadas, rotadas, creadas a partir de otra imagen, etc.– la transparencia de los colores

```
$original=$_SERVER['DOCUMENT_ROOT']."/cursophp/images/aviones3.jpg";
for($i=strlen($original)-1;$i>0;$i--){
    if (substr($original,$i,1)=="."){
       $tipo=substr($original,$i+1);
       break;
    }
}
switch($tipo){
    case "jpg":
          $importada=imagecreatefromjpeg($original);
      break;
    case "png":
          $importada=imagecreatefrompng($original);
      break;
    case "gif":
          $importada=imagecreatefromgif($original);
    break;
 }
  Header("Content-type:image/jpeg");
    $fondo=imagecolorallocatealpha($importada,255,255,0,40);
      $im_base=imagerotate($importada,30,$fondo);
           imagejpeg($im_base);
ImageDestroy();
?>
```

Ver ejemplo

Diferencias entre *imagecreate* e *imagecreatetruecolor*

Este es el código fuente de un script que lee una imagen externa y la copia íntegra sobre otra imagen creada mediante PHP. Si visualizas el ejemplo podrás observar las diferencias entre usar la función *imagecreate* o utilizar *imagecreatetruecolor*.

```
<?
$original=$_SERVER['DOCUMENT_ROOT']."/cursophp/images/aviones3.jpg";

for($i=strlen($original)-1;$i>0;$i--){
    if (substr($original,$i,1)=="."){
       $tipo=substr($original,$i+1);
       break;
    }
}
switch($tipo){
    case "jpg":
          $importada=imagecreatefromjpeg($original);
      break;
    case "png":
          $importada=imagecreatefrompng($original);
      break;
    case "gif":
          $importada=imagecreatefromgif($original);
    break;
 }
$dimensiones=getimagesize($original);
$Ancho_original=$dimensiones[0];
$Alto_original=$dimensiones[1];
Header("Content-type:image/png");
$im_base=imagecreate($Ancho_original+20,$Alto_original+20);
$fondo=imagecolorallocate($im_base,255,0,0);
imagecolortransparent($im_base,$fondo);
imagecopy($im_base,$importada,10,10,0,0,$Ancho_original,$Alto_original);
imagepng($im_base);
ImageDestroy();
?>
```

Ver ejemplo

Ejemplo resumen

Aquí tienes un ejemplo en el que hemos utilizado superposiciones de imágenes, con giros, recortes y diferentes grados de transparencia.

Ver ejemplo resumen | Ver codigo fuente

de los fondos de rotación no plantea ningún problema.
Es suficiente usar la función *imagecolortransparent*, eso sí, aplicándola a la imagen correspondiente *antes* de insertarla mediante la opción *copy* en la imagen final.

¡Cuidado!

La función **imagerotate()** no funciona con la versión de la librería GD que se instala con Ubuntu o Debian que es diferente de la que utilizan otras distribuciones de PHP (la de Windows por ejemplo). La forma de resolverlo puedes encontrarla aquí

Ejercicio nº 34

Diseña un script que genere una imagen dinámica a partir de una fotografía. El script debe encuadrarla (recortándola por los cuatro bordes) y colocar en la parte inferior derecha un texto que diga: «Todos los derechos reservados».

Anterior Índice Siguiente