PHP Graphics using GD Library
Lecture-13

By: Ashiqullah Alizai
Herat University
Computer Science Faculty

Alizai.csf@hotmail.com

Introduction

- PHP is not limited to create just HTML output.
- It can also be used to create and manipulate image files in a variety of different image formats, including GIF, PNG, JPEG, WBMP, and XPM.
- PHP can also output image streams directly to a browser.
- GD library of image functions is required for image streaming to a browser.
- GD and PHP may also require other libraries, depending on which image formats you want to work with.
- You can use the image functions in PHP to get the size of JPEG, GIF, PNG, SWF, TIFF and JPEG2000 images.

Introduction con..

- GD is an open source code library for the dynamic creation of images.
- GD is used for creating PNG, JPEG and GIF images and is commonly used to generate charts, graphics, thumbnails on the fly.
- While not restricted to use on the web, the most common applications of GD involve web site development.
- The GD library was originally developed by Thomas Boutell and is now maintained by Pierre-A. Joye under the umbrella of PHP.net

GD Image Formats Supports

- GD supports a varity of formats, below is a list of formats supported by GD and notes to their availability including read/write support.

Formats supported by GD

Format	Read support	Write support	Notes
JPEG	TRUE	TRUE	GD 1.8+
PNG	TRUE	TRUE	
GIF	TRUE	TRUE	Not available between GD 1.6 and GD 2.0.28
XBM	TRUE	TRUE	Read support as of PHP 4.0.1 and write support as of PHP 5.0.0
XPM	TRUE	FALSE	Read support as of PHP 4.0.1, not available on Windows. Requires bundled version of GD
WBMP	TRUE	TRUE	GD 1.8+
WebP	TRUE	TRUE	GD 2.1+, PHP 5.5+

Requirements

- Using gd requires an ANSI C compiler.
- All popular Windows 95 and NT C compilers are ANSI C compliant. Any full-ANSI-standard C compiler should do. Most Unix/Linux OS based systems has the gcc which is ANSI compliant.
- As of version 1.6, the zlib compression library and the libpng library are required.
- As of version 1.6.2, text using anti aliased TrueType fonts can be drawn if the libttf library is installed, but this is not mandatory.
- Zlib is available for almost every platform from the zlib web site.
- Libpng is available for a variety of platforms from the PNG web site.

Requirements

- A PNG viewer is also required and is supported through almost every web browser that currently exists.
- The GD library can be added to the php installation if the php services are compiled besides the bundle package offered within the apache installation and therefore any additional installation won't be required and will be available within the php itself.
- In order to find out if the installed PHP services provides the GD library support the php info page should be created with the code below.

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

Using GD libraries within the PHP

```
<?php
    header("Content-type: image/png");
    $string = $_GET['text'];
    $im = imagecreatefrompng("images/button1.png");
    $orange = imagecolorallocate($im, 220, 210, 60);
    $px = (imagesx($im) - 7.5 * strlen($string)) / 2;
    imagestring($im, 3, $px, 9, $string, $orange);
    imagepng($im);
     imagedestroy($im);
?>
```

Creating An Image With the PHP GD Library

- Three standard type of images that can be created from scratch with the PHP GD Library:
- JPG, GIF and PNG
- JPG Is Designed to Compress Full-Color Images, & Is Ideal For Digital Photos, Etc.
- GIF Is Designed to Support Only 256 Colors, Along With Transparency, Interlacing & Animation
- PNG Is Designed As An Alternative to GIF, With the Same Basic Features, But Does Not Support Animation

The Header And imagecreate Function

- The header() function is used to tell the browser which content type it will be sent.
- This must be specified before any other output is sent to the browser, whether it is blank lines, PHP code, HTML tags, etc.
- The header options for images are:
  - header('Content-type: image/jpeg');
  - header('Content-type: image/gif');
  - header('Content-type: image/png');
- Now we can use the imagecreate() function to create a blank image, assign it a width and height, and store it in a variable. The syntax is: imagecreate(width, height)

```
<?php
  header('Content-type: image/png');
  $png_image = imagecreate(150, 150);
 ?>
```

Imagecreatetruecolor and imagecolorallocate

- Using the imagecreatetruecolor() function will create a black image (instead of a blank image) with your specified width and height, which you can then see against a white background.
- Alternatively, the next step would be to specify the background color of the image, or fill it, using the imagecolorallocate() function.
  - The syntax is: imagecolorallocate(image, red, green, blue)
- The parameters allow integers between 0 and 255, or hexadecimals between 0x00 and 0xFF.

```
<?php
  header('Content-type: image/png');
  $png_image = imagecreate(150, 150);
  imagecolorallocate($png_image, 15, 142, 210);
 ?>
```

Sending Image to browser

- imagefilltoborder() function is used to flood fill the entire image with color.
- If you tried out either of the above examples, you received an error
- That is because the code was not complete.
- To complete the code, we can now send our image to the browser.
- Three functions are available for this:
  - imagejpeg()
  - imagegif()
  - imagepng()
- Afterward, we can clear up the memory that is being taken up by storing the image with imagedestroy() function.

Wrapping up

```
<?php
    header('Content-type: image/png');
    $png_image = imagecreate(150, 150);
    imagecolorallocate($png_image, 15, 142, 210);
    imagepng($png_image);
    imagedestroy($png_image);
  ?>
```

Summary

Function	Description
header()	Send a Raw HTTP Header
imagecreate()	Create a New (Blank) Palette-Based Image
imagecreatetruecolor()	Create a New True-Color (Black) Image
imagecolorallocate()	Allocate a Color For An Image
imagefilltoborder()	Flood Fill to Specific Color
imagejpeg()	Output a JPEG/JPG Image to Browser or File
imagegif()	Output a GIF Image to Browser or File
imagepng()	Output a PNG Image to Browser or File
imagedestroy()	Destroy An Image

GD and Image Functions

Image With Text

```
<?php
    header ("Content-type: image/png");
    $handle = ImageCreate (130, 50) or die ("Cannot
      Create image");
    $bg_color = ImageColorAllocate ($handle, 255, 0, 0);
    $txt_color = ImageColorAllocate ($handle, 0, 0, 0);
    ImageString ($handle, 5, 5, 18, "PHP.About.com",
      $txt_color);
    ImagePng ($handle);
 ?>
```

Playing with Fonts

```
<?php
 header ("Content-type: image/png");
 $handle = ImageCreate (130, 50) or die ("Cannot Create image");
 $bg_color = ImageColorAllocate ($handle, 255, 0, 0);
 $txt_color = ImageColorAllocate ($handle, 0, 0, 0);
 ImageTTFText ($handle, 20, 15, 30, 40, $txt_color,
    "/Fonts/Quel.ttf", "Quel");
  ImagePng ($handle);
 ?>
```

Note: we are now using *ImageTTFText ()* instead of *ImageString ()*. This allows us to choose our font, which must be in TTF format. The first parameter is our handle, then font size, rotation, starting X, starting Y, text color, font, and finally our text. For the font parameter, you need to include the path to font file.

Drawing Lines

```
<?php
    header ("Content-type: image/png");
     $handle = ImageCreate (130, 50) or die ("Cannot Create image");
     $bg_color = ImageColorAllocate ($handle, 255, 0, 0);
     $txt_color = ImageColorAllocate ($handle, 255, 255, 255);
     $line_color = ImageColorAllocate ($handle, 0, 0, 0);
     ImageLine($handle, 65, 0, 130, 50, $line_color);
     ImageString ($handle, 5, 5, 18, "PHP.About.com", $txt_color);
     ImagePng ($handle);
?>
```

Note: In this code, we use *ImageLine ()* to draw a line. The first parameter is our handle, followed by our starting X and Y, our ending X and Y, and finally our color.

Playing with line

Note: To make a cool volcano, we simply put this into a loop, keeping our starting coordinates the same, but moving along the x axis with our finishing coordinates.

```
<?php
    header ("Content-type: image/png");
    $handle = ImageCreate (130, 50) or die ("Cannot Create image");
    $bg_color = ImageColorAllocate ($handle, 255, 0, 0);
    $txt_color = ImageColorAllocate ($handle, 255, 255, 255);
    $line_color = ImageColorAllocate ($handle, 0, 0, 0);
   for($i=0;$i<=129;$i=$i+5)
    {
     ImageLine($handle, 65, 0, $i, 50, $line_color);
    }
    ImageString ($handle, 5, 5, 18, "PHP.About.com", $txt_color);
    ImagePng ($handle);
?>
```

Drawing An Ellipse

Imageellipse () function is used for drawing it. handle, the X and Y center coordinates, the width and height of the ellipse, and the color.

```
<?php
    header ("Content-type: image/png");
    $handle = ImageCreate (130, 50) or die ("Cannot Create image");
    $bg_color = ImageColorAllocate ($handle, 255, 0, 0);
    $txt_color = ImageColorAllocate ($handle, 255, 255, 255);
    $line_color = ImageColorAllocate ($handle, 0, 0, 0);
    imageellipse ($handle, 65, 25, 100, 40, $line_color);
    ImageString ($handle, 5, 5, 18, "PHP.About.com", $txt_color);
    ImagePng ($handle);
?>
```

Playing with elipse

```
<?php
header ("Content-type: image/png");
$handle = ImageCreate (130, 50) or die ("Cannot Create image");
$bg_color = ImageColorAllocate ($handle, 255, 0, 0);
$txt_color = ImageColorAllocate ($handle, 255, 255, 255);
$line_color = ImageColorAllocate ($handle, 0, 0, 0);
    for($i=0;$i<=130;$i=$i+10)
    {
    imageellipse ($handle, $i, 25, 40, 40, $line_color);
    }
ImageString ($handle, 5, 5, 18, "PHP.About.com", $txt_color);
ImagePng ($handle);
?>
```

If you need to create a solid ellipse, you should use I*magefilledellipse ()* instead.

Arcs & Pies

- Using *imagefilledarc* we can create a pie, or a slice.
- The parameters are: handle, center X & Y, width, height, start, end, color, and type.
  - The start and end points are in degrees, starting from the 3 o'clock position.
- The types are:
  - IMG_ARC_PIE- Filled arch
  - IMG_ARC_CHORD- filled with straight edge
  - IMG_ARC_NOFILL- when added as a parameter, makes it unfilled
  - IMG_ARC_EDGED- Connects to center. You will use this with nofill to make an unfilled pie.

Arcs & Pies

```
<?
    header('Content-type: image/png');
    $handle = imagecreate(100, 100);
    $background = imagecolorallocate($handle, 255, 255, 255);
    $red = imagecolorallocate($handle, 255, 0, 0);
    $green = imagecolorallocate($handle, 0, 255, 0);
    $blue = imagecolorallocate($handle, 0, 0, 255);
    imagefilledarc($handle, 50, 50, 100, 50, 0, 90, $red,
      IMG_ARC_PIE);
    imagefilledarc($handle, 50, 50, 100, 50, 90, 225 , $blue,
      IMG_ARC_PIE);
    imagefilledarc($handle, 50, 50, 100, 50, 225, 360 , $green,
      IMG_ARC_PIE);
    imagepng($handle);
?>
```

Before You Begin

- Before you begin, you should have a general knowledge of PHP and know a little about the GD Library. Also be sure you are actually running the GD Library on your version of PHP.

Wraping Up the Basics

```
<?php
header ("Content-type: image/gif");
$handle = ImageCreate (130, 50) or die ("Cannot Create
   image");
$bg_color = ImageColorAllocate ($handle, 255, 0, 0);
$txt_color = ImageColorAllocate ($handle, 0, 0, 0);
ImageString ($handle, 5, 5, 18, "PHP.About.com", $txt_color);
ImageGif ($handle);
?>
```

Note: So far all of the images we have created have been PNG format. Above, we are creating a GIF using the *ImageGif ()* function. We also change are headers accordingly. You can also use *ImageJpeg ()*

Using Imagecopyresized ()

- When using *imagecopyresized ()* we have a number of parameters that need to be entered. They are: destination_image, source_image, starting_X_ordinate, starting_Y_ordinate, initial_X_image_startpoint, initial_Y_image_startpoint, width, height, original_width, original_height.

Using Imagecopyresized ()

```
<?php
$file = 'your.jpg'; // The file you are resizing
$size = 0.45;  //This will set our output to 45% of the original size
header('Content-type: image/jpeg');
 // Setting the resize parameters
 list($width, $height) = getimagesize($file);
 $modwidth = $width * $size;
 $modheight = $height * $size;
$tn= imagecreatetruecolor($modwidth, $modheight); // Creating the
    Canvas
 $source = imagecreatefromjpeg($file);
 // Resizing our image to fit the canvas
 imagecopyresized($tn, $source, 0, 0, 0, 0, $modwidth,
    $modheight, $width, $height);
  imagejpeg($tn); ?>
```

This is the end for this lecture