Generated Documentation

Contents

Package SpriteGenerator Procedural Elements

SpriteGenerator.php

- **Package** SpriteGenerator
- **Author** Chris Morrell / cmorrell.com
- **Author** Saul Rosenbaum / visualchutzpah.com

- **Copyright** Copyright (c) 2008 Chris Morrell
- **Filesource** Source Code for this file
- **License** New

Package SpriteGenerator Classes

Class SpriteGenerator

[line 84]

CSS Sprite Generator Class

Generates CSS sprite pairs either based on an array of pairs or from a directory based on filename rules. Two examples:

```
<?php
require 'SpriteGenerator.php';
$sg = new SpriteGenerator(array(array('file1.gif'), array('file1b.gif'));
$sg->   batchSprites();
?>
```

The above code will overwrite file1.gif with a image containing file1.gif and file1b.gif vertically stacked.

```
<?php
require 'SpriteGenerator.php';
$sg = new SpriteGenerator('./images/');
$sg->   batchSprites();
?>
```

The above code will parse ./images/ and create sprites on using file.ext and file_over.ext (for example file1.gif and file1_over.gif will be converted to file1.gif).

If you would like to use a different matching method, pass a regular expression as the $match1 and $match2 variables. The first substring match is the filename base (in the above example "file1") and the second substring match is the extension (in the above example "gif"). If you'd rather use something like "myfile-a.gif" and "myfile-b.gif" you could use the following two regular expressions:

Match 1: /^([a-z0-9]+)-a\.(jpg|jpeg|jpe|png|gif)$/i Match 2: /^([a-z0-9]+)-b\.(jpg|jpeg|jpe|png|gif)$/i

- **Package** SpriteGenerator
- **Author** Saul Rosenbaum / visualchutzpah.com

- **Author** Chris Morrell / cmorrell.com
- **Copyright** Copyright (c) 2008 Chris Morrell
- **License** New

SpriteGenerator::$defaultOutput

string = 'png' *[line 99]*

Set this to the format you would like to output (gif, png, jpeg)

SpriteGenerator::$_pairs

array = array() *[line 92]*

Internal variable to hold each pair of images to be turned into a sprite

Constructor *SpriteGenerator* function SpriteGenerator::SpriteGenerator([$input = null], [$match1 = '/^([a-z0-9]+)\.(jpg|jpeg|jpe|png|gif)$/i'], [$match2 = '/^([a-z0-9]+)_over\.(jpg|jpeg|jpe|png|gif)$/i']) *[line 111]*

Function Parameters:

- *array|string* **$input**
- **$match1**
- **$match2**

Contructor

Pass the constructor an array and it'll use that array as its pair images. Pass it a directory and it'll parse that directory and create pairs based on the images found there.

void function SpriteGenerator::batchSprites() *[line 174]*

Batch creates sprites for all image pairs

resource function SpriteGenerator::generateSprite($file1, $file2) *[line 191]*

Function Parameters:

- *string* **$file1**
- *string* **$file2**

Generates a sprite image based on two files

boolean function SpriteGenerator::setDirectory($directory, [$match1 = '/^([a-z0-9]+)\.(jpg|jpeg|jpe|png|gif)$/i'], [$match2 = '/^([a-z0-9]+)_over\.(jpg|jpeg|jpe|png|gif)$/i']) *[line 141]*

Function Parameters:

- *string* **$directory**
- **$match1**
- **$match2**

Sets the directory to generate pairs from

boolean function SpriteGenerator::setPairs([$pairs = array()]) *[line 127]*

Function Parameters:

- *array* **$pairs**

Setter function for _pairs

resource function SpriteGenerator::_createImageFromFile($filename) *[line 222]*

Function Parameters:

- *string* **$filename**

Internal method that creates an image resource from a file name (choosing the correct GD function based on file extension)

void function SpriteGenerator::_parseDirectory($directory, $match1, $match2) *[line 152]*

Function Parameters:

- *string* **$directory**
- **$match1**

- **$match2**

Parses a passed directory

void function SpriteGenerator::_writeSprite($image, $filename) *[line 254]*
Function Parameters:

- *resource* **$image**
- *string* **$filename**

Internal method for writing a sprite to disk

Appendices

Appendix A - Class Trees

Package SpriteGenerator

SpriteGenerator

- SpriteGenerator

Appendix B - README/CHANGELOG/INSTALL

README

Sprite Generator v01
Quick Notes:

Authors:

Saul Rosenbaum / visualchutzpah.com
Chris Morrell / cmorrell.com

This is a quick little utility to take some of the pain out of
stitching together graphics for use as css-sprites it was conceived
to fit a specific workflow. The class is nicely commented,
read on for a quick overview.

To utilize it follow the simple steps below:

1. As you normally would create a folder of individual
   graphics that represent the states of your graphics.

2. Designate your over-state with '_over' so if your neutral state was 'home.gif'
   it's corresponding over state would be 'home_over.gif' [likewise if
   you want to change this default behavior you can pass your own
   pattern to the SpriteGenerator constructor - see line 37 of the
   class for the expected arguments]

   Example:
```
      working_directory
        spriteGen.php
        images
          a00.png
          a00_over.png
          a01.png
          a01_over.png
```

3. MAKE A COPY YOUR GRAPHICS - this class consumes the individual graphics,
   We fully reccomend you work on copies of your images

4. Pass the directory path containing your individual graphics to the class [line 190]
   $sg = new SpriteGenerator("./images/");
   If all has gone as expected you should now have:

   Results:
```
      working_directory
        spriteGen.php
        images
          a00.png
          a01.png
```

Where a00.png is a combined image of a00.png and a00_over.png stitched together vertically.

Notes on Color Space:
The class can deal with any 'web-friendly' format of graphic - in regards to stitching gifs - if both states don't share the same color palette you may get some artifacts where the palette of the overstate gets remapped to the palette of the neutral state, I'm sure it's addressable programatically but working with pngs and outputting as gifs is a simple enough solution.

Possible Enhancements:
- The most obvious one is support for a variable number of states.
- The addition of a vertical offset variable - we didn't need it but adding it to the generateSprite method would be simple.
- I suppose there are situations when you want to work horizontally rather than vertically - but I can't imagine what they may be.
- Some type of GUI front-end to make this process drag and drop from the desktop (perhaps via PHP-GTK).

Additional Info:
- info on css-sprites can be found here: http://www.alistapart.com/articles/sprites
- info on the benefits of reducing http requests canbe found here:
http://yuiblog.com/blog/2006/11/28/performance-research-part-1/

Appendix C - Source Code

Package SpriteGenerator

File Source for SpriteGenerator.php

Documentation for this file is available at SpriteGenerator.php

```php
<?php
/**
 * Copyright (c) 2008, Chris Morrell
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *
 *  - Redistributions of source code must retain the above copyright notice,
 *    this list of conditions and the following disclaimer.
 *
 *  - Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 *
 *  - Neither the name of Chris Morrell nor the names of its
 *    contributors may be used to endorse or promote products derived from this
 *    software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
 * ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * @category  Utilities
 * @package   SpriteGenerator
 * @copyright Copyright (c) 2008 Chris Morrell
 * @author    Saul Rosenbaum / visualchutzpah.com
 * @author    Chris Morrell / cmorrell.com
 * @license   New BSD (see above)
 */

/**
 * CSS Sprite Generator Class
 *
 * Generates CSS sprite pairs either based on an array of pairs or from a
 * directory based on filename rules.  Two examples:
 *
 * <code>
 * <?php
 * require 'SpriteGenerator.php';
 * $sg = new SpriteGenerator(array(array('file1.gif'), array('file1b.gif'));
 * $sg->batchSprites();
 * ?>
 * </code>
 *
 * The above code will overwrite file1.gif with a image containing file1.gif
 * and file1b.gif vertically stacked.
 *
 * <code>
 * <?php
 * require 'SpriteGenerator.php';
 * $sg = new SpriteGenerator('./images/');
 * $sg->batchSprites();
 * ?>
 * </code>
 *
 * The above code will parse ./images/ and create sprites on using file.ext and
 * file_over.ext (for example file1.gif and file1_over.gif will be converted to
 * file1.gif).
 *
 * If you would like to use a different matching method, pass a regular expression
 * as the $match1 and $match2 variables.  The first substring match is the filename
 * base (in the above example "file1") and the second substring match is the extension
 * (in the above example "gif").  If you'd rather use something like "myfile-a.gif" and
 * "myfile-b.gif" you could use the following two regular expressions:
 *
 * Match 1: /^([a-z0-9]+)-a\.(jpg|jpeg|jpe|png|gif)$/i
 * Match 2: /^([a-z0-9]+)-b\.(jpg|jpeg|jpe|png|gif)$/i
 *
 * @category  Utilities
 * @package   SpriteGenerator
 * @copyright Copyright (c) 2008 Chris Morrell
 * @author    Saul Rosenbaum / visualchutzpah.com
 * @author    Chris Morrell / cmorrell.com
 * @license   New BSD (see above)
 */
class SpriteGenerator
{
    /**
     * Internal variable to hold each pair of images to be
     * turned into a sprite
     *
     * @var array
     */
    var $_pairs = array();

    /**
     * Set this to the format you would like to output (gif, png, jpeg)
     *
     * @var string
     */
    var $defaultOutput = 'png';

    /**
     * Contructor
     *
     * Pass the constructor an array and it'll use that array as its
     * pair images.  Pass it a directory and it'll parse that directory
     * and create pairs based on the images found there.
     *
     * @param array|string$input
     * @return SpriteGenerator
     */
    function SpriteGenerator($input = null, $match1 = '/^([a-z0-9]+)\.(jpg|jpeg|jpe|png|gif)$/i', $match2 = '/^([a-z0-9]+)_over\.(jpg|jpeg|jpe|png|gif)$/i')
    {
        if (!is_null($input))
        {
            if (is_array($input)) $this->_pairs = $input;
            elseif (is_dir($input)) $this->_parseDirectory($input, $match1, $match2);
            else die('Unable to auto-detect ' . var_export($input, true));
        }
    }

    /**
     * Setter function for _pairs
     *
     * @param array $pairs
     * @return boolean
     */
    function setPairs($pairs = array())
    {
        if (!is_array($pairs)) return false;

        $this->_pairs = $pairs;
        return true;
    }

    /**
     * Sets the directory to generate pairs from
     *
     * @param string $directory
     * @return boolean
     */
    function setDirectory($directory, $match1 = '/^([a-z0-9]+)\.(jpg|jpeg|jpe|png|gif)$/i', $match2 = '/^([a-z0-9]+)_over\.(jpg|jpeg|jpe|png|gif)$/i')
    {
        if (is_dir($directory)) return $this->_parseDirectory($input, $match1, $match2);
        return false;
    }

    /**
     * Parses a passed directory
     *
     * @param string $directory
     */
    function _parseDirectory($directory, $match1, $match2)
    {
        if (!$handle = opendir($directory)) return false;

        $pairs = array();
        while (false !== ($file = readdir($handle)))
        {
            if ($file == "."           || $file == ".."          ) continue;

            if (preg_match($match1, $file, $matches)) $pairs[$matches[1]][1] = $directory . $file; // TODO: Check for separator
            elseif (preg_match($match2, $file, $matches)) $pairs[$matches[1]][2] = $directory . $file;
        }
        closedir($handle);

        foreach ($pairs as $pair)
            $this->   _pairs[] = array($pair[1], $pair[2]);
    }

    /**
     * Batch creates sprites for all image pairs
     *
     */
    function batchSprites()
    {
        foreach ($this->   _pairs as $pair)
        {
            $image = $this->   generateSprite($pair[0], $pair[1]);
            $this->   _writeSprite($image, $pair[0]);
            unlink($pair[1]);
        }
    }

    /**
     * Generates a sprite image based on two files
     *
     * @param string $file1
     * @param string $file2
     * @return resource
     */
    function generateSprite($file1, $file2)
    {
        $image1 = $this->   _createImageFromFile($file1) or die("     Cannot open {$file1}."     );
        $image2 = $this->   _createImageFromFile($file2) or die("     Cannot open {$file2}."     );

        $imageWidth1 = imagesx($image1);
        $imageWidth2 = imagesx($image2);

        $imageHeight1 = imagesy($image1);
        $imageHeight2 = imagesy($image2);

        $width = ($imageWidth1 >    $imageWidth2 ? $imageWidth1 : $imageWidth2);
        $height = $imageHeight1 + $imageHeight2;

        $image = @imagecreatetruecolor($width, $height) or die('Unable to create sprite.');
        imagecopymerge($image, $image1, 0, 0, 0, 0, $imageWidth1, $imageHeight1, 100) or die('Unable to create sprite.');
        imagecopymerge($image, $image2, 0, $imageHeight1, 0, 0, $imageWidth2, $imageHeight2, 100) or die('Unable to create sprite.');

        imagedestroy($image1);
        imagedestroy($image2);

        return $image;
    }

    /**
     * Internal method that creates an image resource from a file name (choosing
     * the correct GD function based on file extension)
     *
     * @param string $filename
     * @return resource
```

```
         */
        function _createImageFromFile($filename)
        {
            if (!is_readable($filename)) die("     Unable to read {$filename}"     );

            preg_match("|\.([a-z0-9]{2,4})$|i"          , $filename, $matches);
            $extension = $matches[1];
            switch ($extension)
            {
                case 'jpg':
                case 'jpeg':
                case 'jpe':
                    return @imagecreatefromjpeg($filename);

                case 'png':
                    return @imagecreatefrompng($filename);

                case 'gif':
                    return @imagecreatefromgif($filename);

                default:
                    die("     Unable to recognize {$filename}'s image type."     );
            }

            return false;
        }

        /**
         * Internal method for writing a sprite to disk
         *
         * @param resource $image
         * @param string $filename
         */
        function _writeSprite($image, $filename)
        {
            $function = 'image' . $this->   defaultOutput;

            if (file_exists($filename) && !        is_writable($filename)) die("     {$filename} is not 
writable!"     );
            $function($image, $filename) or die ("     Cannot write {$filename}"     );
            imagedestroy($image);
        }
    }

    /*

    // Example Usage:
    $sg = new SpriteGenerator("./images/");
    $sg->batchSprites();

    */
```

Index