Ming (flash)

Introduction

First of all: Ming is not an acronym. Ming is an open-source (LGPL) library which allows you to create SWF ("Flash") format movies. Ming supports almost all of Flash 4's features, including: shapes, gradients, bitmaps (pngs and jpegs), morphs ("shape tweens"), text, buttons, actions, sprites ("movie clips"), streaming mp3, and color transforms --the only thing that's missing is sound events.

Note that all values specifying length, distance, size, etc. are in "twips", twenty units per pixel. That's pretty much arbitrary, though, since the player scales the movie to whatever pixel size is specified in the embed/object tag, or the entire frame if not embedded.

Ming offers a number of advantages over the existing PHP/libswf module. You can use Ming anywhere you can compile the code, whereas libswf is closed-source and only available for a few platforms, Windows not one of them. Ming provides some insulation from the mundane details of the SWF file format, wrapping the movie elements in PHP objects. Also, Ming is still being maintained; if there's a feature that you want to see, just let us know at » http://ming.sourceforge.net/.

Ming was added in PHP 4.0.5.

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension?including the names of its functions and any other documentation surrounding this extension?may change without notice in a future release of PHP. This extension should be used at your own risk.

Installing/Configuring

Requirements

To use Ming with PHP, you first need to build and install the Ming library. Source code and installation instructions are available at the Ming home page: » http://ming.sourceforge.net/ along with examples, a small tutorial, and the latest news.

Download the ming archive. Unpack the archive. Go in the Ming directory. make. make install.

This will build *libming.so* and install it into */usr/lib/*, and copy *ming.h* into */usr/include/*. Edit the *PREFIX*= line in the *Makefile* to change the installation directory.

Installation

Example #1 - built into PHP (Unix)

```
mkdir <phpdir>/ext/ming
   cp php_ext/* <phpdir>/ext/ming
   cd <phpdir>
   ./buildconf
   ./configure --with-ming <other config options>
```

Build and install PHP as usual, restart web server if necessary.

Now either just add *extension=php_ming.so* to your *php.ini* file, or put *dl('php_ming.so');* at the head of all of your Ming scripts.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

MING_NEW (integer)

MING_ZLIB (integer)

SWFBUTTON_HIT (integer)

SWFBUTTON_DOWN (integer)

SWFBUTTON_OVER (integer)

SWFBUTTON_UP (integer)

SWFBUTTON_MOUSEUPOUTSIDE (integer)

SWFBUTTON_DRAGOVER (integer)

SWFBUTTON_DRAGOUT (integer)

SWFBUTTON_MOUSEUP (integer)

SWFBUTTON_MOUSEDOWN (integer)

SWFBUTTON_MOUSEOUT (integer)

SWFBUTTON_MOUSEOVER (integer)

SWFFILL_RADIAL_GRADIENT (integer)

SWFFILL_LINEAR_GRADIENT (integer)

SWFFILL_TILED_BITMAP (integer)

SWFFILL_CLIPPED_BITMAP (integer)

SWFTEXTFIELD_HASLENGTH (integer)

SWFTEXTFIELD_NOEDIT (integer)

SWFTEXTFIELD_PASSWORD (integer)

SWFTEXTFIELD_MULTILINE (integer)

SWFTEXTFIELD_WORDWRAP (integer)

SWFTEXTFIELD_DRAWBOX (integer)

SWFTEXTFIELD_NOSELECT (integer)

SWFTEXTFIELD_HTML (integer)

SWFTEXTFIELD_ALIGN_LEFT (integer)

SWFTEXTFIELD_ALIGN_RIGHT (integer)

SWFTEXTFIELD_ALIGN_CENTER (integer)

SWFTEXTFIELD_ALIGN_JUSTIFY (integer)

SWFACTION_ONLOAD (integer)

SWFACTION_ENTERFRAME (integer)

SWFACTION_UNLOAD (integer)

SWFACTION_MOUSEMOVE (integer)

SWFACTION_MOUSEDOWN (integer)

SWFACTION_MOUSEUP (integer)

SWFACTION_KEYDOWN (integer)

SWFACTION_KEYUP (integer)

SWFACTION_DATA (integer)

Examples

SWFAction Examples

This simple example will move the red square across the window.

Example #2 - swfaction() example

```
<?php
 $s = new SWFShape();
 $f = $s->addFill(0xff, 0, 0);
 $s->setRightFill($f);

 $s->movePenTo(-500, -500);
 $s->drawLineTo(500, -500);
 $s->drawLineTo(500, 500);
 $s->drawLineTo(-500, 500);
 $s->drawLineTo(-500, -500);

 $p = new SWFSprite();
 $i = $p->add($s);
 $i->setDepth(1);
 $p->nextFrame();

 for ($n=0; $n<5; ++$n) {
   $i->rotate(-15);
   $p->nextFrame();
 }

 $m = new SWFMovie();
 $m->setBackground(0xff, 0xff, 0xff);
 $m->setDimension(6000, 4000);

 $i = $m->add($p);
 $i->setDepth(1);
 $i->moveTo(-500,2000);
 $i->setName("box");

 $m->add(new SWFAction("/box.x += 3;"));
 $m->nextFrame();
 $m->add(new SWFAction("gotoFrame(0); play();"));
 $m->nextFrame();

 header('Content-type: application/x-shockwave-flash');
 $m->output();
?>
```

This simple example tracks down your mouse on the screen.

Example #3 - swfaction() example

```
<?php
```

```
$m = new SWFMovie();
$m->setRate(36.0);
$m->setDimension(1200, 800);
$m->setBackground(0, 0, 0);

/* mouse tracking sprite - empty, but follows mouse so we can
   get its x and y coordinates */

$i = $m->add(new SWFSprite());
$i->setName('mouse');

$m->add(new SWFAction("
  startDrag('/mouse', 1); /* '1' means lock sprite to the mouse */
"));

/* might as well turn off antialiasing, since these are just squares. */

$m->add(new SWFAction("
  this.quality = 0;
"));

/* morphing box */
$r = new SWFMorph();
$s = $r->getShape1();

/* Note this is backwards from normal shapes.  No idea why. */
$s->setLeftFill($s->addFill(0xff, 0xff, 0xff));
$s->movePenTo(-40, -40);
$s->drawLine(80, 0);
$s->drawLine(0, 80);
$s->drawLine(-80, 0);
$s->drawLine(0, -80);

$s = $r->getShape2();

$s->setLeftFill($s->addFill(0x00, 0x00, 0x00));
$s->movePenTo(-1, -1);
$s->drawLine(2, 0);
$s->drawLine(0, 2);
$s->drawLine(-2, 0);
$s->drawLine(0, -2);

/* sprite container for morphing box -
   this is just a timeline w/ the box morphing */

$box = new SWFSprite();
$box->add(new SWFAction("
  stop();
"));
$i = $box->add($r);

for ($n=0; $n<=20; ++$n) {
  $i->setRatio($n/20);
  $box->nextFrame();
}

/* this container sprite allows us to use the same action code many times
*/
```

```
 $cell = new SWFSprite();
 $i = $cell->add($box);
 $i->setName('box');

 $cell->add(new SWFAction("

   setTarget('box');

   /* ...x means the x coordinate of the parent, i.e. (..).x */
   dx = (/mouse.x + random(6)-3 - ...x)/5;
   dy = (/mouse.y + random(6)-3 - ...y)/5;
   gotoFrame(int(dx*dx + dy*dy));

 "));

 $cell->nextFrame();
 $cell->add(new SWFAction("

   gotoFrame(0);
   play();

 "));

 $cell->nextFrame();

 /* finally, add a bunch of the cells to the movie */

 for ($x=0; $x<12; ++$x) {
   for ($y=0; $y<8; ++$y) {
     $i = $m->add($cell);
     $i->moveTo(100*$x+50, 100*$y+50);
   }
 }

 $m->nextFrame();

 $m->add(new SWFAction("

   gotoFrame(1);
   play();

 "));

 header('Content-type: application/x-shockwave-flash');
 $m->output();
?>
```

Same as above, but with nice colored balls...

Example #4 - swfaction() example

```
<?php

 $m = new SWFMovie();
 $m->setDimension(11000, 8000);
 $m->setBackground(0x00, 0x00, 0x00);
```

```
 $m->add(new SWFAction("

this.quality = 0;
/frames.visible = 0;
startDrag('/mouse', 1);

 "));

 // mouse tracking sprite
 $t = new SWFSprite();
 $i = $m->add($t);
 $i->setName('mouse');

 $g = new SWFGradient();
 $g->addEntry(0, 0xff, 0xff, 0xff, 0xff);
 $g->addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
 $g->addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
 $g->addEntry(1.0, 0xff, 0xff, 0xff, 0);

 // gradient shape thing
 $s = new SWFShape();
 $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
 $f->scaleTo(0.03);
 $s->setRightFill($f);
 $s->movePenTo(-600, -600);
 $s->drawLine(1200, 0);
 $s->drawLine(0, 1200);
 $s->drawLine(-1200, 0);
 $s->drawLine(0, -1200);

 // need to make this a sprite so we can multColor it
 $p = new SWFSprite();
 $p->add($s);
 $p->nextFrame();

 // put the shape in here, each frame a different color
 $q = new SWFSprite();
 $q->add(new SWFAction("gotoFrame(random(7)+1); stop();"));
 $i = $q->add($p);

 $i->multColor(1.0, 1.0, 1.0);
 $q->nextFrame();
 $i->multColor(1.0, 0.5, 0.5);
 $q->nextFrame();
 $i->multColor(1.0, 0.75, 0.5);
 $q->nextFrame();
 $i->multColor(1.0, 1.0, 0.5);
 $q->nextFrame();
 $i->multColor(0.5, 1.0, 0.5);
 $q->nextFrame();
 $i->multColor(0.5, 0.5, 1.0);
 $q->nextFrame();
 $i->multColor(1.0, 0.5, 1.0);
 $q->nextFrame();

 // finally, this one contains the action code
 $p = new SWFSprite();
 $i = $p->add($q);
 $i->setName('frames');
 $p->add(new SWFAction("
```

```
dx = (/:mousex-/:lastx)/3 + random(10)-5;
dy = (/:mousey-/:lasty)/3;
x = /:mousex;
y = /:mousey;
alpha = 100;

 "));
 $p->nextFrame();

 $p->add(new SWFAction("

this.x = x;
this.y = y;
this.alpha = alpha;
x += dx;
y += dy;
dy += 3;
alpha -= 8;

 "));
 $p->nextFrame();

 $p->add(new SWFAction("prevFrame(); play();"));
 $p->nextFrame();

 $i = $m->add($p);
 $i->setName('frames');
 $m->nextFrame();

 $m->add(new SWFAction("

lastx = mousex;
lasty = mousey;
mousex = /mouse.x;
mousey = /mouse.y;

++num;

if (num == 11)
 num = 1;

removeClip('char' & num);
duplicateClip(/frames, 'char' & num, num);

 "));

 $m->nextFrame();
 $m->add(new SWFAction("prevFrame(); play();"));

 header('Content-type: application/x-shockwave-flash');
 $m->output();
?>
```

SWFSPrite basic examples

This simple example will spin gracefully a big red square.

Example #5 - swfsprite() example

```
<?php
$s = new SWFShape();
$s->setRightFill($s->addFill(0xff, 0, 0));
$s->movePenTo(-500, -500);
$s->drawLineTo(500, -500);
$s->drawLineTo(500, 500);
$s->drawLineTo(-500, 500);
$s->drawLineTo(-500, -500);

$p = new SWFSprite();
$i = $p->add($s);
$p->nextFrame();
$i->rotate(15);
$p->nextFrame();
$i->rotate(15);
$p->nextFrame();
$i->rotate(15);
$p->nextFrame();
$i->rotate(15);
$p->nextFrame();
$i->rotate(15);
$p->nextFrame();

$m = new SWFMovie();
$i = $m->add($p);
$i->moveTo(1500, 1000);
$i->setName("blah");

$m->setBackground(0xff, 0xff, 0xff);
$m->setDimension(3000, 2000);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

Ming Functions

ming_keypress

ming_keypress -- Returns the action flag for keyPress(char)

Description

int **ming_keypress** (string `$char`)

Warning
This function is currently not documented; only its argument list is available.

ming_setcubicthreshold

ming_setcubicthreshold -- Set cubic threshold

Description

void **ming_setcubicthreshold** (int `$threshold`)

Sets the threshold error for drawing cubic beziers.

Parameters

`threshold`
: The Threshold. Lower is more accurate, hence larger file size.

Return Values

No value is returned.

ming_setscale

ming_setscale -- Set the global scaling factor.

Description

void **ming_setscale** (int `$scale`)

Sets the scale of the output SWF. Inside the SWF file, coordinates are measured in TWIPS, rather than PIXELS. There are 20 TWIPS in 1 pixel.

Parameters

`scale`
: The scale to be set.

Return Values

No value is returned.

ming_setswfcompression

ming_setswfcompression -- Sets the SWF output compression

Description

void **ming_setswfcompression** (int `$level`)

Sets the SWF output compression level.

Parameters

`level`
The new compression level. Should be a value between 1 and 9 inclusive.

Return Values

No value is returned.

ming_useconstants

ming_useconstants -- Use constant pool

Description

void **ming_useconstants** (int `$use`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

ming_useswfversion

ming_useswfversion -- Sets the SWF version

Description

void **ming_useswfversion** (int `$version`)

Sets the SWF version to be used in the movie. This affect the bahaviour of Action Script.

Parameters

`version`
SWF version to use.

Return Values

No value is returned.

Examples

Example #6 - ming_useswfversion() example

```
<?php

$movie = new SWFMovie();
ming_useswfversion(4); // Flash 4

?>
```

The SWFAction class

Introduction

SWFAction.

Class synopsis

SWFAction

SWFAction {

    /* Methods */

    SWFAction __**construct** (string `$script`)
}

Description

The script syntax is based on the C language, but with a lot taken out- the SWF bytecode machine is just too simpleminded to do a lot of things we might like. For instance, we can't implement function calls without a tremendous amount of hackery because the jump bytecode has a hardcoded offset value. No pushing your calling address to the stack and returning- every function would have to know exactly where to return to.

So what's left? The compiler recognises the following tokens:

- break
- for
- continue
- if
- else
- do
- while

There is no typed data; all values in the SWF action machine are stored as strings. The following functions can be used in expressions:

time()
Returns the number of milliseconds (?) elapsed since the movie started.

random(seed)
Returns a pseudo-random number in the range 0-seed.

length(expr)
Returns the length of the given expression.

int(number)
Returns the given number rounded down to the nearest integer.

concat(expr, expr)
Returns the concatenation of the given expressions.

ord(expr)
Returns the ASCII code for the given character

chr(num)
Returns the character for the given ASCII code

substr(string, location, length)
Returns the substring of length *`length`* at location *`location`* of the given string *`string`*.

Additionally, the following commands may be used:
duplicateClip(clip, name, depth)
Duplicate the named movie *`clip`* (aka sprite). The new movie clip has name *`name`* and is at depth *`depth`*.

removeClip(expr)
Removes the named movie clip.

trace(expr)
Write the given expression to the trace log. Doubtful that the browser plugin does anything with this.

startDrag(target, lock, [left, top, right, bottom])
Start dragging the movie clip *`target`*. The *`lock`* argument indicates whether to lock the mouse (?)- use 0 (**FALSE**) or 1 (**TRUE**). Optional parameters define a bounding area for the dragging.

stopDrag()
Stop dragging my heart around. And this movie clip, too.

callFrame(expr)
Call the named frame as a function.

getURL(url, target, [method])
Load the given URL into the named target. The *`target`* argument corresponds to HTML document targets (such as "_top" or "_blank"). The optional *`method`* argument can be POST or GET if you want to submit variables back to the server.

loadMovie(url, target)
Load the given URL into the named target. The *target* argument can be a frame name (I think), or one of the magical values "_level0" (replaces current movie) or "_level1" (loads new movie on top of current movie).

nextFrame()
Go to the next frame.

prevFrame()
Go to the last (or, rather, previous) frame.

play()
Start playing the movie.

stop()
Stop playing the movie.

toggleQuality()
Toggle between high and low quality.

stopSounds()
Stop playing all sounds.

gotoFrame(num)
Go to frame number *num*. Frame numbers start at 0.

gotoFrame(name)
Go to the frame named *name*. Which does a lot of good, since I haven't added frame labels yet.

setTarget(expr)
Sets the context for action. Or so they say- I really have no idea what this does.
And there's one weird extra thing. The expression frameLoaded(num) can be used in if statements and while loops to check if the given frame number has been loaded yet. Well, it's supposed to, anyway, but I've never tested it and I seriously doubt it actually works. You can just use /:framesLoaded instead.

Movie clips (all together now- aka sprites) have properties. You can read all of them (or can you?), you can set some of them, and here they are:

- x
- y
- xScale
- yScale
- currentFrame - (read-only)
- totalFrames - (read-only)
- alpha - transparency level
- visible - 1=on, 0=off (?)

- width - (read-only)
- height - (read-only)
- rotation
- target - (read-only) (???)
- framesLoaded - (read-only)
- name
- dropTarget - (read-only) (???)
- url - (read-only) (???)
- highQuality - 1=high, 0=low (?)
- focusRect - (???)
- soundBufTime - (???)

So, setting a sprite's x position is as simple as */box.x = 100;*. Why the slash in front of the box, though? That's how flash keeps track of the sprites in the movie, just like a Unix filesystem- here it shows that box is at the top level. If the sprite named box had another sprite named biff inside of it, you'd set its x position with /box/biff.x = 100;. At least, I think so; correct me if I'm wrong here.

SWFAction->__construct()

SWFAction->__construct() -- Creates a new SWFAction

Description

SWFAction

SWFAction __**construct** (string `$script`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Creates a new SWFAction and compiles the given *`script`* in it.

Parameters

`script`
An ActionScript snippet to associate with the SWFAction. See SWFAction for more details.

The SWFBitmap class

Introduction

SWFBitmap.

Class synopsis

SWFBitmap

```
SWFBitmap {

    /*  Methods */

    SWFBitmap __construct ( mixed $file [, mixed $alphafile ] )

    float getHeight ( void )

    float getWidth ( void )
}
```

SWFBitmap->__construct()

SWFBitmap->__construct() -- Loads Bitmap object

Description

SWFBitmap

SWFBitmap **__construct** (mixed `$file` [, mixed `$alphafile`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Creates the new SWFBitmap object from the given *`file`*.

Parameters

For both parameters you can provide a file pointer returned by fopen() or the image data, as a binary string.

`file`

Note

We can only deal with baseline (frame 0) jpegs, no baseline optimized or progressive scan jpegs!

You can't import png images directly, though- have to use the png2dbl utility to make a dbl ("define bits lossless") file from the png. The reason for this is that I don't want a dependency on the png library in ming- autoconf should solve this, but that's not set up yet.

`alphafile`

An MSK file to be used as an alpha mask for a JPEG image.

Examples

Example #7 - Importing a DBL file

```
<?php
$s = new SWFShape();
$f = $s->addFill(new SWFBitmap(file_get_contents("image.dbl")));
$s->setRightFill($f);

$s->drawLine(32, 0);
$s->drawLine(0, 32);
$s->drawLine(-32, 0);
$s->drawLine(0, -32);

$m = new SWFMovie();
$m->setDimension(32, 32);
$m->add($s);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

Example #8 - Using an alpha mask

```
<?php

$s = new SWFShape();

// .msk file generated with "gif2mask" utility
$f = $s->addFill(new SWFBitmap(file_get_contents("alphafill.jpg"),
file_get_contents("alphafill.msk")));
$s->setRightFill($f);

$s->drawLine(640, 0);
$s->drawLine(0, 480);
$s->drawLine(-640, 0);
$s->drawLine(0, -480);

$c = new SWFShape();
$c->setRightFill($c->addFill(0x99, 0x99, 0x99));
$c->drawLine(40, 0);
$c->drawLine(0, 40);
$c->drawLine(-40, 0);
$c->drawLine(0, -40);

$m = new SWFMovie();
$m->setDimension(640, 480);
$m->setBackground(0xcc, 0xcc, 0xcc);

// draw checkerboard background
for ($y=0; $y<480; $y+=40) {
 for ($x=0; $x<640; $x+=80) {
   $i = $m->add($c);
   $i->moveTo($x, $y);
 }
```

```
 $y+=40;

 for ($x=40; $x<640; $x+=80) {
   $i = $m->add($c);
   $i->moveTo($x, $y);
 }
}

$m->add($s);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFBitmap->getHeight()

SWFBitmap->getHeight() -- Returns the bitmap's height

Description

SWFBitmap

float **getHeight** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Returns the bitmap's height.

Return Values

Returns the bitmap height in pixels.

See Also

- SWFBitmap->getWidth()

SWFBitmap->getWidth()

SWFBitmap->getWidth() -- Returns the bitmap's width

Description

SWFBitmap

float **getWidth** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Returns the bitmap's width.

Return Values

Returns the bitmap width in pixels.

See Also

- SWFBitmap->getHeight()

The SWFButton class

Introduction

SWFButton.

Class synopsis

SWFButton

SWFButton {

/* Methods */

void **addAction** (SWFAction `$action`, int `$flags`)

SWFSoundInstance **addASound** (SWFSound `$sound`, int `$flags`)

void **addShape** (SWFShape `$shape`, int `$flags`)

SWFButton __**construct** (void)

void **setAction** (SWFAction `$action`)

void **setDown** (SWFShape `$shape`)

void **setHit** (SWFShape `$shape`)

void **setMenu** (int `$flag`)

void **setOver** (SWFShape `$shape`)

void **setUp** (SWFShape `$shape`)

}

SWFButton->addAction()

SWFButton->addAction() -- Adds an action

Description

SWFButton

void **addAction** (SWFAction `$action`, int `$flags`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Adds the given *`action`* to the button for the given conditions.

Parameters

`action`
An SWFAction, returned by SWFAction->__construct().

`flags`
The following *`flags`* are valid: **SWFBUTTON_MOUSEOVER**, **SWFBUTTON_MOUSEOUT**, **SWFBUTTON_MOUSEUP**, **SWFBUTTON_MOUSEUPOUTSIDE**, **SWFBUTTON_MOUSEDOWN**, **SWFBUTTON_DRAGOUT** and **SWFBUTTON_DRAGOVER**.

Return Values

No value is returned.

See Also

- SWFButton->addShape()
- SWFAction

SWFButton->addASound()

SWFButton->addASound() -- Associates a sound with a button transition

Description

SWFButton

SWFSoundInstance **addASound** (SWFSound `$sound`, int `$flags`)

Warning

This function is currently not documented; only its argument list is available.

SWFButton->addShape()

SWFButton->addShape() -- Adds a shape to a button

Description

SWFButton

void **addShape** (SWFShape `$shape`, int `$flags`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Adds the given *`shape`* to the button.

Parameters

`shape`
: An SWFShape instance

`flags`
: The following *`flags`* are valid: **SWFBUTTON_UP**, **SWFBUTTON_OVER**, **SWFBUTTON_DOWN** and **SWFBUTTON_HIT**. **SWFBUTTON_HIT** isn't ever displayed, it defines the hit region for the button. That is, everywhere the hit shape would be drawn is considered a "touchable" part of the button.

Return Values

No value is returned.

SWFButton->__construct()

SWFButton->__construct() -- Creates a new Button

Description

SWFButton

SWFButton __**construct** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Creates a new Button.

Examples

This simple example will show your usual interactions with buttons : rollover, rollon, mouseup, mousedown, noaction.

Example #9 - Usual interactions with buttons

```
<?php

$f = new SWFFont("_serif");

$p = new SWFSprite();

function label($string)
{
 global $f;

 $t = new SWFTextField();
 $t->setFont($f);
 $t->addString($string);
 $t->setHeight(200);
 $t->setBounds(3200, 200);
 return $t;
}

function addLabel($string)
{
 global $p;
```

```
 $i = $p->add(label($string));
 $p->nextFrame();
 $p->remove($i);
}

$p->add(new SWFAction("stop();"));
addLabel("NO ACTION");
addLabel("SWFBUTTON_MOUSEUP");
addLabel("SWFBUTTON_MOUSEDOWN");
addLabel("SWFBUTTON_MOUSEOVER");
addLabel("SWFBUTTON_MOUSEOUT");
addLabel("SWFBUTTON_MOUSEUPOUTSIDE");
addLabel("SWFBUTTON_DRAGOVER");
addLabel("SWFBUTTON_DRAGOUT");

function rect($r, $g, $b)
{
 $s = new SWFShape();
 $s->setRightFill($s->addFill($r, $g, $b));
 $s->drawLine(600, 0);
 $s->drawLine(0, 600);
 $s->drawLine(-600, 0);
 $s->drawLine(0, -600);

 return $s;
}

$b = new SWFButton();
$b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP | SWFBUTTON_HIT);
$b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
$b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(1);"),
         SWFBUTTON_MOUSEUP);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(2);"),
     SWFBUTTON_MOUSEDOWN);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(3);"),
     SWFBUTTON_MOUSEOVER);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(4);"),
     SWFBUTTON_MOUSEOUT);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(5);"),
     SWFBUTTON_MOUSEUPOUTSIDE);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(6);"),
     SWFBUTTON_DRAGOVER);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(7);"),
     SWFBUTTON_DRAGOUT);

$m = new SWFMovie();
$m->setDimension(4000, 3000);

$i = $m->add($p);
$i->setName("label");
$i->moveTo(400, 1900);
```

```
$i = $m->add($b);
$i->moveTo(400, 900);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

This simple example will enables you to drag draw a big red button on the windows. No drag-and-drop, just moving around.

Example #10 - Drag example

```
<?php

$s = new SWFShape();
$s->setRightFill($s->addFill(0xff, 0, 0));
$s->drawLine(1000,0);
$s->drawLine(0,1000);
$s->drawLine(-1000,0);
$s->drawLine(0,-1000);

$b = new SWFButton();
$b->addShape($s, SWFBUTTON_HIT | SWFBUTTON_UP | SWFBUTTON_DOWN |
SWFBUTTON_OVER);

$b->addAction(new SWFAction("startDrag('/test', 0);"), // '0' means don't
lock to mouse
     SWFBUTTON_MOUSEDOWN);

$b->addAction(new SWFAction("stopDrag();"),
     SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);

$p = new SWFSprite();
$p->add($b);
$p->nextFrame();

$m = new SWFMovie();
$i = $m->add($p);
$i->setName('test');
$i->moveTo(1000,1000);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFButton->setAction()

SWFButton->setAction() -- Sets the action

Description

SWFButton

void **setAction** (SWFAction `$action`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Sets the action to be performed when the button is clicked.

This is a shortcut for SWFButton->addAction() called with the **SWFBUTTON_MOUSEUP** flag.

Parameters

`action`
An SWFAction, returned by SWFAction->__construct().

Return Values

No value is returned.

See Also

- SWFButton->addAction()
- SWFAction

SWFButton->setDown()

SWFButton->setDown() -- Alias for addShape(shape, SWFBUTTON_DOWN)

Description

SWFButton

void **setDown** (SWFShape `$shape`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfbutton->setdown() alias for addShape(shape, SWFBUTTON_DOWN).

Return Values

No value is returned.

See Also

- SWFButton->addShape()
- SWFAction

SWFButton->setHit()

SWFButton->setHit() -- Alias for addShape(shape, SWFBUTTON_HIT)

Description

SWFButton

void **setHit** (SWFShape `$shape`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfbutton->sethit() alias for addShape(shape, SWFBUTTON_HIT).

Return Values

No value is returned.

See Also

- SWFButton->addShape()
- SWFAction

SWFButton->setMenu()

SWFButton->setMenu() -- enable track as menu button behaviour

Description

SWFButton

void **setMenu** (int `$flag`)

Warning

This function is currently not documented; only its argument list is available.

Parameters

`flag`
: This parameter can be used for a slight different behavior of buttons. You can set it to 0 (off) or 1 (on).

Return Values

No value is returned.

SWFButton->setOver()

SWFButton->setOver() -- Alias for addShape(shape, SWFBUTTON_OVER)

Description

SWFButton

void **setOver** (SWFShape `$shape`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfbutton->setover() alias for addShape(shape, SWFBUTTON_OVER).

Return Values

No value is returned.

See Also

- SWFButton->addShape()
- SWFAction

SWFButton->setUp()

SWFButton->setUp() -- Alias for addShape(shape, SWFBUTTON_UP)

Description

SWFButton

void **setUp** (SWFShape `$shape`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfbutton->setup() alias for addShape(shape, SWFBUTTON_UP).

Return Values

No value is returned.

See Also

- SWFButton->addShape()
- SWFAction

The SWFDisplayItem class

Introduction

SWFDisplayItem.

Class synopsis

SWFDisplayItem

SWFDisplayItem {

/* Methods */

void **addAction** (SWFAction `$action`, int `$flags`)

void **addColor** (int `$red`, int `$green`, int `$blue` [, int `$a`])

void **endMask** (void)

float **getRot** (void)

float **getX** (void)

float **getXScale** (void)

float **getXSkew** (void)

float **getY** (void)

float **getYScale** (void)

float **getYSkew** (void)

void **move** (int `$dx`, int `$dy`)

void **moveTo** (int `$x`, int `$y`)

void **multColor** (int `$red`, int `$green`, int `$blue` [, int `$a`])

void **remove** (void)

void **rotate** (float `$angle`)

void **rotateTo** (float `$angle`)

void **scale** (int `$dx`, int `$dy`)

void **scaleTo** (int `$x` [, int `$y`])

void **setDepth** (float `$depth`)

void **setMaskLevel** (int `$level`)

void **setMatrix** (float `$a`, float `$b`, float `$c`, float `$d`, float `$x`, float `$y`)

void **setName** (string `$name`)

void **setRatio** (float `$ratio`)

void **skewX** (float `$ddegrees`)

void **skewXTo** (float `$degrees`)

void **skewY** (float `$ddegrees`)

void **skewYTo** (float `$degrees`)

}

SWFDisplayItem->addAction()

SWFDisplayItem->addAction() -- Adds this SWFAction to the given SWFSprite instance

Description

SWFDisplayItem

void **addAction** (SWFAction `$action`, int `$flags`)

Warning

This function is currently not documented; only its argument list is available.

Parameters

`action`
: An SWFAction, returned by SWFAction->__construct().

`flags`

Return Values

No value is returned.

See Also

- SWFAction

SWFDisplayItem->addColor()

SWFDisplayItem->addColor() -- Adds the given color to this item's color transform

Description

SWFDisplayItem

void **addColor** (int `$red`, int `$green`, int `$blue` [, int `$a`])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->addcolor() adds the color to this item's color transform. The color is given in its RGB form.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

Return Values

No value is returned.

SWFDisplayItem->endMask()

SWFDisplayItem->endMask() -- Another way of defining a MASK layer

Description

SWFDisplayItem

void **endMask** (void)

Warning
This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

SWFDisplayItem->getRot()

SWFDisplayItem->getRot() --

Description

SWFDisplayItem

float **getRot** (void)

Warning
This function is currently not documented; only its argument list is available.

SWFDisplayItem->getX()

SWFDisplayItem->getX() --

Description

SWFDisplayItem

float **getX** (void)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFDisplayItem->getY()

SWFDisplayItem->getXScale()

SWFDisplayItem->getXScale() --

Description

SWFDisplayItem

float **getXScale** (void)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFDisplayItem->getYScale()

SWFDisplayItem->getXSkew()

SWFDisplayItem->getXSkew() --

Description

SWFDisplayItem

float **getXSkew** (void)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFDisplayItem->getYSkew()

SWFDisplayItem->getY()

SWFDisplayItem->getY() --

Description

SWFDisplayItem

float **getY** (void)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFDisplayItem->getX()

SWFDisplayItem->getYScale()

SWFDisplayItem->getYScale() --

Description

SWFDisplayItem

float **getYScale** (void)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFDisplayItem->getXScale()

SWFDisplayItem->getYSkew()

SWFDisplayItem->getYSkew() --

Description

SWFDisplayItem

float **getYSkew** (void)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFDisplayItem->getXSkew()

SWFDisplayItem->move()

SWFDisplayItem->move() -- Moves object in relative coordinates

Description

SWFDisplayItem

void **move** (int `$dx`, int `$dy`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->move() moves the current object by (*dx*, *dy*) from its current position.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

See Also

- SWFDisplayItem->moveTo()

SWFDisplayItem->moveTo()

SWFDisplayItem->moveTo() -- Moves object in global coordinates

Description

SWFDisplayItem

void **moveTo** (int $x, int $y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->moveto() moves the current object to (*x*, *y*) in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

See Also

- SWFDisplayItem->move()

SWFDisplayItem->multColor()

SWFDisplayItem->multColor() -- Multiplies the item's color transform

Description

SWFDisplayItem

void **multColor** (int `$red`, int `$green`, int `$blue` [, int `$a`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->multcolor() multiplies the item's color transform by the given values.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Parameters

These parameters are integers between 0 and 255 or hexadecimals between *0x00* and *0xFF*:

`red`
Value of red component

`green`
Value of green component

`blue`
Value of blue component

`a`
Value of alpha component

Return Values

No value is returned.

Examples

This simple example will modify your picture's atmosphere to Halloween (use a landscape or bright picture).

Example #11 - swfdisplayitem->multcolor() example

```
<?php

$b = new SWFBitmap(file_get_contents("backyard.jpg"));
// note use your own picture :-)
$s = new SWFShape();
$s->setRightFill($s->addFill($b));
$s->drawLine($b->getWidth(), 0);
$s->drawLine(0, $b->getHeight());
$s->drawLine(-$b->getWidth(), 0);
$s->drawLine(0, -$b->getHeight());

$m = new SWFMovie();
$m->setDimension($b->getWidth(), $b->getHeight());

$i = $m->add($s);

for ($n=0; $n<=20; ++$n) {
 $i->multColor(1.0-$n/10, 1.0, 1.0);
 $i->addColor(0xff*$n/20, 0, 0);
 $m->nextFrame();
}

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFDisplayItem->remove()

SWFDisplayItem->remove() -- Removes the object from the movie

Description

SWFDisplayItem

void **remove** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->remove() removes this object from the movie's display list.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

See Also

- SWFMovie->add()

SWFDisplayItem->rotate()

SWFDisplayItem->rotate() -- Rotates in relative coordinates

Description

SWFDisplayItem

void **rotate** (float `$angle`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->rotate() rotates the current object by *`angle`* degrees from its current rotation.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

See Also

- SWFDisplayItem->rotateTo()

SWFDisplayItem->rotateTo()

SWFDisplayItem->rotateTo() -- Rotates the object in global coordinates

Description

SWFDisplayItem

void **rotateTo** (float `$angle`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->rotateto() set the current object rotation to *`angle`* degrees in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

Examples

This example bring three rotating string from the background to the foreground. Pretty nice.

Example #12 - swfdisplayitem->rotateto() example

```
<?php
$thetext =  "ming!";

$f = new SWFFont("Bauhaus 93.fdb");

$m = new SWFMovie();
$m->setRate(24.0);
$m->setDimension(2400, 1600);
$m->setBackground(0xff, 0xff, 0xff);

// functions with huge numbers of arbitrary
```

```
// arguments are always a good idea!  Really!

function text($r, $g, $b, $a, $rot, $x, $y, $scale, $string)
{
 global $f, $m;

 $t = new SWFText();
 $t->setFont($f);
 $t->setColor($r, $g, $b, $a);
 $t->setHeight(960);
 $t->moveTo(-($f->getWidth($string))/2, $f->getAscent()/2);
 $t->addString($string);

 // we can add properties just like a normal PHP var,
 // as long as the names aren't already used.
 // e.g., we can't set $i->scale, because that's a function

 $i = $m->add($t);
 $i->x = $x;
 $i->y = $y;
 $i->rot = $rot;
 $i->s = $scale;
 $i->rotateTo($rot);
 $i->scale($scale, $scale);

 // but the changes are local to the function, so we have to
 // return the changed object.  kinda weird..

 return $i;
}

function step($i)
{
 $oldrot = $i->rot;
 $i->rot = 19*$i->rot/20;
 $i->x = (19*$i->x + 1200)/20;
 $i->y = (19*$i->y + 800)/20;
 $i->s = (19*$i->s + 1.0)/20;

 $i->rotateTo($i->rot);
 $i->scaleTo($i->s, $i->s);
 $i->moveTo($i->x, $i->y);

 return $i;
}

// see?  it sure paid off in legibility:

$i1 = text(0xff, 0x33, 0x33, 0xff, 900, 1200, 800, 0.03, $thetext);
$i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 1200, 800, 0.04, $thetext);
$i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 1200, 800, 0.001, $thetext);

for ($i=1; $i<=100; ++$i) {
 $i1 = step($i1);
 $i2 = step($i2);
 $i3 = step($i3);

 $m->nextFrame();
}
```

```
header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

See Also

- SWFDisplayItem->rotate()

SWFDisplayItem->scale()

SWFDisplayItem->scale() -- Scales the object in relative coordinates

Description

SWFDisplayItem

void **scale** (int `$dx`, int `$dy`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->scale() scales the current object by (*dx*, *dy*) from its current size.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

See Also

- SWFDisplayItem->scaleTo()

SWFDisplayItem->scaleTo()

SWFDisplayItem->scaleTo() -- Scales the object in global coordinates

Description

SWFDisplayItem

void **scaleTo** (int $x [, int $y])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->scaleto() scales the current object to (*x*, *y*) in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

See Also

- SWFDisplayItem->scale()

SWFDisplayItem->setDepth()

SWFDisplayItem->setDepth() -- Sets z-order

Description

SWFDisplayItem

void **setDepth** (float `$depth`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->setdepth() sets the object's z-order to `depth`. Depth defaults to the order in which instances are created (by adding a shape/text to a movie)- newer ones are on top of older ones. If two objects are given the same depth, only the later-defined one can be moved.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

SWFDisplayItem->setMaskLevel()

SWFDisplayItem->setMaskLevel() -- Defines a MASK layer at level

Description

SWFDisplayItem

void **setMaskLevel** (int `$level`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

SWFDisplayItem->setMatrix()

SWFDisplayItem->setMatrix() -- Sets the item's transform matrix

Description

SWFDisplayItem

void **setMatrix** (float `$a`, float `$b`, float `$c`, float `$d`, float `$x`, float `$y`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

SWFDisplayItem->setName()

SWFDisplayItem->setName() -- Sets the object's name

Description

SWFDisplayItem

void **setName** (string `$name`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->setname() sets the object's name to *`name`*, for targetting with action script. Only useful on sprites.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

SWFDisplayItem->setRatio()

SWFDisplayItem->setRatio() -- Sets the object's ratio

Description

SWFDisplayItem

void **setRatio** (float `$ratio`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->setratio() sets the object's ratio to *`ratio`*. Obviously only useful for morphs.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

Examples

This simple example will morph nicely three concentric circles.

Example #13 - swfdisplayitem->setname() example

```
<?php

$p = new SWFMorph();

$g = new SWFGradient();
$g->addEntry(0.0, 0, 0, 0);
$g->addEntry(0.16, 0xff, 0xff, 0xff);
$g->addEntry(0.32, 0, 0, 0);
$g->addEntry(0.48, 0xff, 0xff, 0xff);
$g->addEntry(0.64, 0, 0, 0);
$g->addEntry(0.80, 0xff, 0xff, 0xff);
$g->addEntry(1.00, 0, 0, 0);
```

```
$s = $p->getShape1();
$f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
$f->scaleTo(0.05);
$s->setLeftFill($f);
$s->movePenTo(-160, -120);
$s->drawLine(320, 0);
$s->drawLine(0, 240);
$s->drawLine(-320, 0);
$s->drawLine(0, -240);

$g = new SWFGradient();
$g->addEntry(0.0, 0, 0, 0);
$g->addEntry(0.16, 0xff, 0, 0);
$g->addEntry(0.32, 0, 0, 0);
$g->addEntry(0.48, 0, 0xff, 0);
$g->addEntry(0.64, 0, 0, 0);
$g->addEntry(0.80, 0, 0, 0xff);
$g->addEntry(1.00, 0, 0, 0);

$s = $p->getShape2();
$f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
$f->scaleTo(0.05);
$f->skewXTo(1.0);
$s->setLeftFill($f);
$s->movePenTo(-160, -120);
$s->drawLine(320, 0);
$s->drawLine(0, 240);
$s->drawLine(-320, 0);
$s->drawLine(0, -240);

$m = new SWFMovie();
$m->setDimension(320, 240);
$i = $m->add($p);
$i->moveTo(160, 120);

for ($n=0; $n<=1.001; $n+=0.01) {
   $i->setRatio($n);
   $m->nextFrame();
}

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFDisplayItem->skewX()

SWFDisplayItem->skewX() -- Sets the X-skew

Description

SWFDisplayItem

void **skewX** (float `$ddegrees`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->skewx() adds *`ddegrees`* to current x-skew.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

See Also

- SWFDisplayItem->skewXTo()
- SWFDisplayItem->skewY()
- SWFDisplayItem->skewYTo()

SWFDisplayItem->skewXTo()

SWFDisplayItem->skewXTo() -- Sets the X-skew

Description

SWFDisplayItem

void **skewXTo** (float `$degrees`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->skewxto() sets the x-skew to `degrees`. For `degrees` is 1.0, it means a 45-degree forward slant. More is more forward, less is more backward.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

See Also

- SWFDisplayItem->skewX()
- SWFDisplayItem->skewY()
- SWFDisplayItem->skewYTo()

SWFDisplayItem->skewY()

SWFDisplayItem->skewY() -- Sets the Y-skew

Description

SWFDisplayItem

void **skewY** (float `$ddegrees`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->skewy() adds *`ddegrees`* to current y-skew.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

See Also

- SWFDisplayItem->skewYTo()
- SWFDisplayItem->skewX()
- SWFDisplayItem->skewXTo()

SWFDisplayItem->skewYTo()

SWFDisplayItem->skewYTo() -- Sets the Y-skew

Description

SWFDisplayItem

void **skewYTo** (float `$degrees`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfdisplayitem->skewyto() sets the y-skew to *`degrees`*. For *`degrees`* is 1.0, it means a 45-degree forward slant. More is more upward, less is more downward.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

Return Values

No value is returned.

See Also

- SWFDisplayItem->skewY()
- SWFDisplayItem->skewX()
- SWFDisplayItem->skewXTo()

The SWFFill class

Introduction

The SWFFill object allows you to transform (scale, skew, rotate) bitmap and gradient fills.

swffill objects are created by the SWFShape->addFill() method.

Class synopsis

SWFFill

```
SWFFill {
    /*  Methods */
    void moveTo ( int  $x, int  $y )
    void rotateTo ( float  $angle )
    void scaleTo ( int  $x [, int  $y ] )
    void skewXTo ( float  $x )
    void skewYTo ( float  $y )
}
```

SWFFill->moveTo()

SWFFill->moveTo() -- Moves fill origin

Description

SWFFill

void **moveTo** (int $x, int $y)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Moves the fill origin to the given global coordinates.

Parameters

x
X-coordinate

y
Y-coordinate

Return Values

No value is returned.

SWFFill->rotateTo()

SWFFill->rotateTo() -- Sets fill's rotation

Description

SWFFill

void **rotateTo** (float `$angle`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Sets the fill rotation to the given *`angle`*.

Parameters

`angle`
The rotation angle, in degreess.

Return Values

No value is returned.

SWFFill->scaleTo()

SWFFill->scaleTo() -- Sets fill's scale

Description

SWFFill

void **scaleTo** (int $x [, int $y])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Sets the fill scale to the given coordinates.

Parameters

x
X-coordinate

y
Y-coordinate

Return Values

No value is returned.

SWFFill->skewXTo()

SWFFill->skewXTo() -- Sets fill x-skew

Description

SWFFill

void **skewXTo** (float $x)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Sets the fill x-skew to *x*.

Parameters

x

When *x* is 1.0, it is a 45-degree forward slant. More is more forward, less is more backward.

Return Values

No value is returned.

See Also

- SWFFill->skewYTo()

SWFFill->skewYTo()

SWFFill->skewYTo() -- Sets fill y-skew

Description

SWFFill

void **skewYTo** (float $y)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Sets the fill y-skew to *y*.

Parameters

y
When *y* is 1.0, it is a 45-degree upward slant. More is more upward, less is more downward.

Return Values

No value is returned.

See Also

- SWFFill->skewXTo()

The SWFFont class

Introduction

The SWFFont object represent a reference to the font definition, for us with SWFText->setFont() and SWFTextField->setFont().

Class synopsis

SWFFont

```
SWFFont {

    /*  Methods */

    SWFFont __construct ( string $filename )

    float getAscent ( void )

    float getDescent ( void )

    float getLeading ( void )

    string getShape ( int $code )

    float getUTF8Width ( string $string )

    float getWidth ( string $string )
}
```

SWFFont->__construct()

SWFFont->__construct() -- Loads a font definition

Description

SWFFont

SWFFont __**construct** (string `$filename`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

If *`filename`* is the name of an FDB file (i.e., it ends in ".fdb"), load the font definition found in said file. Otherwise, create a browser-defined font reference.

FDB ("font definition block") is a very simple wrapper for the SWF DefineFont2 block which contains a full description of a font. One may create FDB files from SWT Generator template files with the included makefdb utility- look in the util directory off the main ming distribution directory.

Browser-defined fonts don't contain any information about the font other than its name. It is assumed that the font definition will be provided by the movie player. The fonts _serif, _sans, and _typewriter should always be available. For example:

```
<?php
$f = newSWFFont("_sans");
?>
```

will give you the standard sans-serif font, probably the same as what you'd get with *<font name="sans-serif">* in HTML.

SWFFont->getAscent()

SWFFont->getAscent() -- Returns the ascent of the font, or 0 if not available

Description

SWFFont

float **getAscent** (void)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFFont->getDescent()

SWFFont->getDescent()

SWFFont->getDescent() -- Returns the descent of the font, or 0 if not available

Description

SWFFont

float **getDescent** (void)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFFont->getAscent()

SWFFont->getLeading()

SWFFont->getLeading() -- Returns the leading of the font, or 0 if not available

Description

SWFFont

float **getLeading** (void)

Warning
This function is currently not documented; only its argument list is available.

SWFFont->getShape()

SWFFont->getShape() -- Returns the glyph shape of a char as a text string

Description

SWFFont

string **getShape** (int `$code`)

Warning

This function is currently not documented; only its argument list is available.

SWFFont->getUTF8Width()

SWFFont->getUTF8Width() -- Calculates the width of the given string in this font at full height

Description

SWFFont

float **getUTF8Width** (string `$string`)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFFont->getWidth()

SWFFont->getWidth()

SWFFont->getWidth() -- Returns the string's width

Description

SWFFont

float **getWidth** (string `$string`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swffont->getwidth() returns the string *`string`* 's width, using font's default scaling. You'll probably want to use the **swftext()** version of this method which uses the text object's scale.

See Also

- SWFFont->getUTF8Width()

The SWFFontChar class

Introduction

SWFFontChar.

Class synopsis

SWFFontChar

SWFFontChar {

    /* Methods */

    void **addChars** (string `$char`)

    void **addUTF8Chars** (string `$char`)
}

SWFFontChar->addChars()

SWFFontChar->addChars() -- Adds characters to a font for exporting font

Description

SWFFontChar

void **addChars** (string `$char`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

See Also

- SWFFontChar->addUTF8Chars()

SWFFontChar->addUTF8Chars()

SWFFontChar->addUTF8Chars() -- Adds characters to a font for exporting font

Description

SWFFontChar

void **addUTF8Chars** (string `$char`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

See Also

- SWFFontChar->addChars()

The SWFGradient class

Introduction

SWFGradient.

Class synopsis

SWFGradient

SWFGradient {

    /* Methods */

    void **addEntry** (float `$ratio`, int `$red`, int `$green`, int `$blue` [, int `$a`])

    SWFGradient __**construct** (void)
}

SWFGradient->addEntry()

SWFGradient->addEntry() -- Adds an entry to the gradient list

Description

SWFGradient

void **addEntry** (float `$ratio`, int `$red`, int `$green`, int `$blue` [, int `$a`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfgradient->addentry() adds an entry to the gradient list. *`ratio`* is a number between 0 and 1 indicating where in the gradient this color appears. Thou shalt add entries in order of increasing ratio.

`red`, *`green`*, *`blue`* is a color (RGB mode). Last parameter *`a`* is optional.

Return Values

No value is returned.

SWFGradient->__construct()

SWFGradient->__construct() -- Creates a gradient object

Description

SWFGradient

SWFGradient __**construct** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfgradient() creates a new SWFGradient object.

After you've added the entries to your gradient, you can use the gradient in a shape fill with the **swfshape->addfill()** method.

SWFGradient has the following methods : **swfgradient->addentry()**.

This simple example will draw a big black-to-white gradient as background, and a reddish disc in its center.

Example #14 - swfgradient() example

```
<?php

 $m = new SWFMovie();
 $m->setDimension(320, 240);

 $s = new SWFShape();

 // first gradient- black to white
 $g = new SWFGradient();
 $g->addEntry(0.0, 0, 0, 0);
 $g->addEntry(1.0, 0xff, 0xff, 0xff);

 $f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
 $f->scaleTo(0.01);
 $f->moveTo(160, 120);
 $s->setRightFill($f);
 $s->drawLine(320, 0);
 $s->drawLine(0, 240);
```

```
 $s->drawLine(-320, 0);
 $s->drawLine(0, -240);

 $m->add($s);

 $s = new SWFShape();

 // second gradient- radial gradient from red to transparent
 $g = new SWFGradient();
 $g->addEntry(0.0, 0xff, 0, 0, 0xff);
 $g->addEntry(1.0, 0xff, 0, 0, 0);

 $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
 $f->scaleTo(0.005);
 $f->moveTo(160, 120);
 $s->setRightFill($f);
 $s->drawLine(320, 0);
 $s->drawLine(0, 240);
 $s->drawLine(-320, 0);
 $s->drawLine(0, -240);

 $m->add($s);

 header('Content-type: application/x-shockwave-flash');
 $m->output();
?>
```

The SWFMorph class

Introduction

The methods here are sort of weird. It would make more sense to just have newSWFMorph(shape1, shape2);, but as things are now, shape2 needs to know that it's the second part of a morph. (This, because it starts writing its output as soon as it gets drawing commands- if it kept its own description of its shapes and wrote on completion this and some other things would be much easier.)

Class synopsis

SWFMorph

```
SWFMorph {

    /*  Methods */

    SWFMorph __construct ( void )

    SWFShape getShape1 ( void )

    SWFShape getShape2 ( void )
}
```

SWFMorph->__construct()

SWFMorph->__construct() -- Creates a new SWFMorph object

Description

SWFMorph

SWFMorph __**construct** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Creates a new SWFMorph object.

Also called a "shape tween". This thing lets you make those tacky twisting things that make your computer choke. Oh, joy!

Examples

This simple example will morph a big red square into a smaller blue black-bordered square.

Example #15 - swfmorph() example

```
<?php
 $p = new SWFMorph();

 $s = $p->getShape1();
 $s->setLine(0, 0, 0, 0);

 /* Note that this is backwards from normal shapes (left instead of right).
    I have no idea why, but this seems to work.. */

 $s->setLeftFill($s->addFill(0xff, 0, 0));
 $s->movePenTo(-1000,-1000);
 $s->drawLine(2000,0);
 $s->drawLine(0,2000);
 $s->drawLine(-2000,0);
 $s->drawLine(0,-2000);

 $s = $p->getShape2();
 $s->setLine(60,0,0,0);
```

```
 $s->setLeftFill($s->addFill(0, 0, 0xff));
 $s->movePenTo(0,-1000);
 $s->drawLine(1000,1000);
 $s->drawLine(-1000,1000);
 $s->drawLine(-1000,-1000);
 $s->drawLine(1000,-1000);

 $m = new SWFMovie();
 $m->setDimension(3000,2000);
 $m->setBackground(0xff, 0xff, 0xff);

 $i = $m->add($p);
 $i->moveTo(1500,1000);

 for ($r=0.0; $r<=1.0; $r+=0.1) {
   $i->setRatio($r);
   $m->nextFrame();
 }

 header('Content-type: application/x-shockwave-flash');
 $m->output();
?>
```

SWFMorph->getShape1()

SWFMorph->getShape1() -- Gets a handle to the starting shape

Description

SWFMorph

SWFShape **getShape1** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Gets the morph's starting shape.

Return Values

Returns a SWFShape object.

See Also

- SWFMorph->getShape2()

SWFMorph->getShape2()

SWFMorph->getShape2() -- Gets a handle to the ending shape

Description

SWFMorph

SWFShape **getShape2** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Gets the morph's ending shape.

Return Values

Returns a SWFShape object.

See Also

- SWFMorph->getShape1()

The SWFMovie class

Introduction

SWFMovie is a movie object representing an SWF movie.

Class synopsis

SWFMovie

SWFMovie {

/* Methods */

mixed **add** (object `$instance`)

void **addExport** (SWFCharacter `$char`, string `$name`)

mixed **addFont** (SWFFont `$font`)

SWFMovie **__construct** (int `$version`)

SWFSprite **importChar** (string `$libswf`, string `$name`)

SWFFontChar **importFont** (string `$libswf`, string `$name`)

void **labelFrame** (string `$label`)

void **nextFrame** (void)

int **output** ([int `$compression`])

void **remove** (object `$instance`)

int **save** (string `$filename` [, int `$compression`])

int **saveToFile** (stream `$x` [, int `$compression`])

void **setbackground** (int `$red`, int `$green`, int `$blue`)

void **setDimension** (int `$width`, int `$height`)

void **setFrames** (int `$number`)

```
    void setRate ( int $rate )

    SWFSoundInstance startSound ( SWFSound $sound )

    void stopSound ( SWFSound $sound )

    int streamMP3 ( mixed $mp3file [, float $skip ] )

    void writeExports ( void )
}
```

SWFMovie->add()

SWFMovie->add() -- Adds any type of data to a movie

Description

SWFMovie

mixed **add** (object `$instance`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Adds an SWF object *`instance`* to the current movie.

Parameters

`instance`
Any type of object instance, like SWFShape, SWFText, SWFFont.

Return Values

For displayable types (shape, text, button, sprite), this returns an SWFDisplayItem, a handle to the object in a display list. Thus, you can add the same shape to a movie multiple times and get separate handles back for each separate instance.

See Also

- SWFMovie->remove()

SWFMovie->addExport()

SWFMovie->addExport() --

Description

SWFMovie

void **addExport** (SWFCharacter `$char`, string `$name`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

SWFMovie->addFont()

SWFMovie->addFont() --

Description

SWFMovie

mixed **addFont** (SWFFont `$font`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

SWFMovie->__construct()

SWFMovie->__construct() -- Creates a new movie object, representing an SWF version 4 movie

Description

SWFMovie

SWFMovie **__construct** (int `$version`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Creates a new movie object, representing an SWF movie.

Parameters

`version`
The desired SWF version. Default is 4.

SWFMovie->importChar()

SWFMovie->importChar() --

Description

SWFMovie

SWFSprite **importChar** (string `$libswf`, string `$name`)

Warning

This function is currently not documented; only its argument list is available.

See Also

- SWFMovie->importFont()

SWFMovie->importFont()

SWFMovie->importFont() --

Description

SWFMovie

SWFFontChar **importFont** (string `$libswf`, string `$name`)

Warning

This function is currently not documented; only its argument list is available.

See Also

- SWFMovie->importChar()

SWFMovie->labelFrame()

SWFMovie->labelFrame() -- Labels a frame

Description

SWFMovie

void **labelFrame** (string `$label`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

SWFMovie->nextFrame()

SWFMovie->nextFrame() -- Moves to the next frame of the animation

Description

SWFMovie

void **nextFrame** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Moves to the next frame of the animation.

Return Values

No value is returned.

SWFMovie->output()

SWFMovie->output() -- Dumps your lovingly prepared movie out

Description

SWFMovie

int **output** ([int `$compression`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Dumps the SWFMovie.

Don't forget to send the *Content-Type* HTTP header file before using this function, in order to display the movie in a browser.

Parameters

`compression`
: The compression level can be a value between 0 and 9, defining the SWF compression similar to gzip compression. This parameter is only available as of Flash MX (6).

Return Values

Return the number of bytes written or **FALSE** on error.

Examples

Example #16 - Displaying your $movie in a browser

```
<?php
header('Content-type: application/x-shockwave-flash');
$movie->output();
```

```
?>
```

See Also

- SWFMovie->save()
- SWFMovie->saveToFile()

SWFMovie->remove()

SWFMovie->remove() -- Removes the object instance from the display list

Description

SWFMovie

void **remove** (object `$instance`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Removes the given object *`instance`* from the display list.

Return Values

No value is returned.

See Also

- SWFMovie->add()

SWFMovie->save()

SWFMovie->save() -- Saves the SWF movie in a file

Description

SWFMovie

int **save** (string `$filename` [, int `$compression`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Saves the SWF movie to the specified *`filename`*.

Parameters

`filename`
: The path to the saved SWF document.

`compression`
: The compression level can be a value between 0 and 9, defining the SWF compression similar to gzip compression. This parameter is only available as of Flash MX (6).

Return Values

Return the number of bytes written or **FALSE** on error.

See Also

- SWFMovie->output()
- SWFMovie->saveToFile()

SWFMovie->saveToFile()

SWFMovie->saveToFile() --

Description

SWFMovie

int **saveToFile** (stream `$x` [, int `$compression`])

Warning

This function is currently not documented; only its argument list is available.

Parameters

x

compression
: The compression level can be a value between 0 and 9, defining the SWF compression similar to gzip compression. This parameter is only available as of Flash MX (6).

Return Values

Return the number of bytes written or **FALSE** on error.

See Also

- SWFMovie->output()
- SWFMovie->save()

SWFMovie->setbackground()

SWFMovie->setbackground() -- Sets the background color

Description

SWFMovie

void **setbackground** (int `$red`, int `$green`, int `$blue`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Sets the background color.

Why is there no rgba version? Think about it, you might want to let the HTML background show through. There's a way to do that, but it only works on IE4. Search the » http://www.macromedia.com/ site for details.

Parameters

These parameters are integers between 0 and 255 or hexadecimals between *0x00* and *0xFF*:

`red`
: Value of red component

`green`
: Value of green component

`blue`
: Value of blue component

Return Values

No value is returned.

SWFMovie->setDimension()

SWFMovie->setDimension() -- Sets the movie's width and height

Description

SWFMovie

void **setDimension** (int `$width`, int `$height`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Sets the movie's dimension to the specified *`width`* and *`height`*.

Parameters

`width`
The movie width.

`height`
The movie height.

Return Values

No value is returned.

SWFMovie->setFrames()

SWFMovie->setFrames() -- Sets the total number of frames in the animation

Description

SWFMovie

void **setFrames** (int `$number`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Sets the total number of frames in the animation to the given *`number`*.

Parameters

`number`
The number of frames.

Return Values

No value is returned.

SWFMovie->setRate()

SWFMovie->setRate() -- Sets the animation's frame rate

Description

SWFMovie

void **setRate** (int `$rate`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Sets the frame rate to the specified *`rate`*.

Animation will slow down if the player can't render frames fast enough- unless there's a streaming sound, in which case display frames are sacrificed to keep sound from skipping.

Parameters

`rate`
The frame rate, in frame per seconds.

Return Values

No value is returned.

SWFMovie->startSound()

SWFMovie->startSound() --

Description

SWFMovie

SWFSoundInstance **startSound** (SWFSound `$sound`)

Warning

This function is currently not documented; only its argument list is available.

See Also

- SWFMovie->stopSound()

SWFMovie->stopSound()

SWFMovie->stopSound() --

Description

SWFMovie

void **stopSound** (SWFSound `$sound`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

See Also

- SWFMovie->startSound()

SWFMovie->streamMP3()

SWFMovie->streamMP3() -- Streams a MP3 file

Description

SWFMovie

int **streamMP3** (mixed `$mp3file` [, float `$skip`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Streams the given MP3 file *`mp3file`*.

This method is not very robust in dealing with oddities (can skip over an initial ID3 tag, but that's about it).

Note that the movie isn't smart enough to put enough frames in to contain the entire mp3 stream- you'll have to add (length of song * frames per second) frames to get the entire stream in.

Parameters

`mp3file`
Can be a file pointer returned by fopen() or the MP3 data, as a binary string.

`skip`
Number of seconds to skip.

Return Values

Return number of frames.

ChangeLog

Version	Description
5.2.0	*skip* added

Examples

Example #17 - Streaming example

```
<?php
$m = new SWFMovie();
$m->setRate(12.0);
$m->streamMp3(file_get_contents("distortobass.mp3"));
// use your own MP3

// The file is 11.85 seconds at 12.0 fps = 142 frames
$m->setFrames(142);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFMovie->writeExports()

SWFMovie->writeExports() --

Description

SWFMovie

void **writeExports** (void)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

The SWFPrebuiltClip class

Introduction

SWFPrebuiltClip.

Class synopsis

SWFPrebuiltClip

SWFPrebuiltClip {

    /* Methods */

    SWFPrebuiltClip __**construct** ([string `$file`])
}

SWFPrebuiltClip->__construct()

SWFPrebuiltClip->__construct() -- Returns a SWFPrebuiltClip object

Description

SWFPrebuiltClip

SWFPrebuiltClip __**construct** ([string `$file`])

Warning

This function is currently not documented; only its argument list is available.

The SWFShape class

Introduction

SWFShape.

Class synopsis

SWFShape

SWFShape {

/* Methods */

SWFFill **addFill** (int `$red`, int `$green`, int `$blue` [, int `$a`])

SWFShape __**construct** (void)

void **drawArc** (float `$r`, float `$startAngle`, float `$endAngle`)

void **drawCircle** (float `$r`)

int **drawCubic** (float `$bx`, float `$by`, float `$cx`, float `$cy`, float `$dx`, float `$dy`)

int **drawCubicTo** (float `$bx`, float `$by`, float `$cx`, float `$cy`, float `$dx`, float `$dy`)

int **drawCurve** (int `$controldx`, int `$controldy`, int `$anchordx`, int `$anchordy` [, int `$targetdx`], int `$targetdy`)

int **drawCurveTo** (int `$controlx`, int `$controly`, int `$anchorx`, int `$anchory` [, int `$targetx`], int `$targety`)

void **drawGlyph** (SWFFont `$font`, string `$character` [, int `$size`])

void **drawLine** (int `$dx`, int `$dy`)

void **drawLineTo** (int `$x`, int `$y`)

void **movePen** (int `$dx`, int `$dy`)

void **movePenTo** (int `$x`, int `$y`)

void **setLeftFill** (SWFGradient `$fill`)

```
    void setLine ( SWFShape $shape )

    void setRightFill ( SWFGradient $fill )
}
```

SWFShape->addFill()

SWFShape->addFill() -- Adds a solid fill to the shape

Description

SWFShape

SWFFill **addFill** (int `$red`, int `$green`, int `$blue` [, int `$a`])

SWFFill **addFill** (SWFBitmap `$bitmap` [, int `$flags`])

SWFFill **addFill** (SWFGradient `$gradient` [, int `$flags`])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

SWFShape->addFill() adds a solid fill to the shape's list of fill styles.
SWFShape->addFill() accepts three different types of arguments.

`red`, *`green`*, *`blue`* is a color (RGB mode). Last parameter *`a`* is optional.

The *`bitmap`* argument is an **SWFBitmap()** object. The *`flags`* argument can be one of the following values: SWFFILL_CLIPPED_BITMAP, SWFFILL_TILED_BITMAP, SWFFILL_LINEAR_GRADIENT or SWFFILL_RADIAL_GRADIENT. Default is SWFFILL_TILED_BITMAP for SWFBitmap and SWFFILL_LINEAR_GRADIENT for SWFGradient.

The *`gradient`* argument is an **SWFGradient()** object. The flags argument can be one of the following values : SWFFILL_RADIAL_GRADIENT or SWFFILL_LINEAR_GRADIENT. Default is SWFFILL_LINEAR_GRADIENT. I'm sure about this one. Really.

SWFShape->addFill() returns an **SWFFill()** object for use with the **SWFShape->setLeftFill()** and **SWFShape->setRightFill()** functions described below.

Examples

This simple example will draw a frame on a bitmap. Ah, here's another buglet in the flash player- it doesn't seem to care about the second shape's bitmap's transformation in a morph. According to spec, the bitmap should stretch along with the shape in this example..

Example #18 - SWFShape->addFill() example

```
<?php

$p = new SWFMorph();

$b = new SWFBitmap(file_get_contents("alphafill.jpg"));
// use your own bitmap
$width = $b->getWidth();
$height = $b->getHeight();

$s = $p->getShape1();
$f = $s->addFill($b, SWFFILL_TILED_BITMAP);
$f->moveTo(-$width/2, -$height/4);
$f->scaleTo(1.0, 0.5);
$s->setLeftFill($f);
$s->movePenTo(-$width/2, -$height/4);
$s->drawLine($width, 0);
$s->drawLine(0, $height/2);
$s->drawLine(-$width, 0);
$s->drawLine(0, -$height/2);

$s = $p->getShape2();
$f = $s->addFill($b, SWFFILL_TILED_BITMAP);

// these two have no effect!
$f->moveTo(-$width/4, -$height/2);
$f->scaleTo(0.5, 1.0);

$s->setLeftFill($f);
$s->movePenTo(-$width/4, -$height/2);
$s->drawLine($width/2, 0);
$s->drawLine(0, $height);
$s->drawLine(-$width/2, 0);
$s->drawLine(0, -$height);

$m = new SWFMovie();
$m->setDimension($width, $height);
$i = $m->add($p);
$i->moveTo($width/2, $height/2);

for ($n=0; $n<1.001; $n+=0.03) {
   $i->setRatio($n);
   $m->nextFrame();
}

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

See Also

- SWFShape->setLeftFill()

- SWFShape->setRightFill()

SWFShape->__construct()

SWFShape->__construct() -- Creates a new shape object

Description

SWFShape

SWFShape __**construct** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Created a new SWFShape object.

Examples

This simple example will draw a big red elliptic quadrant.

Example #19 - swfshape() example

```
<?php
$s = new SWFShape();
$s->setLine(40, 0x7f, 0, 0);
$s->setRightFill($s->addFill(0xff, 0, 0));
$s->movePenTo(200, 200);
$s->drawLineTo(6200, 200);
$s->drawLineTo(6200, 4600);
$s->drawCurveTo(200, 4600, 200, 200);

$m = new SWFMovie();
$m->setDimension(6400, 4800);
$m->setRate(12.0);
$m->add($s);
$m->nextFrame();

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFShape->drawArc()

SWFShape->drawArc() -- Draws an arc of radius r centered at the current location, from angle startAngle to angle endAngle measured clockwise from 12 o'clock

Description

SWFShape

void **drawArc** (float `$r`, float `$startAngle`, float `$endAngle`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

See Also

- SWFShape->drawCircle()

SWFShape->drawCircle()

SWFShape->drawCircle() -- Draws a circle of radius r centered at the current location, in a counter-clockwise fashion

Description

SWFShape

void **drawCircle** (float `$r`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

SWFShape->drawCubic()

SWFShape->drawCubic() -- Draws a cubic bezier curve using the current position and the three given points as control points

Description

SWFShape

int **drawCubic** (float `$bx`, float `$by`, float `$cx`, float `$cy`, float `$dx`, float `$dy`)

Warning

This function is currently not documented; only its argument list is available.

See Also

- SWFShape->drawCubicTo()

SWFShape->drawCubicTo()

SWFShape->drawCubicTo() -- Draws a cubic bezier curve using the current position and the three given points as control points

Description

SWFShape

int **drawCubicTo** (float `$bx`, float `$by`, float `$cx`, float `$cy`, float `$dx`, float `$dy`)

Warning

This function is currently not documented; only its argument list is available.

See Also

- SWFShape->drawCubic()

SWFShape->drawCurve()

SWFShape->drawCurve() -- Draws a curve (relative)

Description

SWFShape

int **drawCurve** (int `$controldx`, int `$controldy`, int `$anchordx`, int `$anchordy` [, int `$targetdx`], int `$targetdy`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfshape->drawcurve() draws a quadratic curve (using the current line style,set by **swfshape->setline()**) from the current pen position to the relative position (*`anchorx`*, *`anchory`*) using relative control point (*`controlx`*, *`controly`*). That is, head towards the control point, then smoothly turn to the anchor point.

With 6 parameters, it draws a cubic bezier to point (x+ *`targetdx`*, x+ *`targetdy`*) with control points (x+ *`controldx`*, y+ *`controldy`*) and (x+ *`anchordx`*, y+ *`anchordy`*).

See Also

- SWFShape->drawCurve()

SWFShape->drawCurveTo()

SWFShape->drawCurveTo() -- Draws a curve

Description

SWFShape

int **drawCurveTo** (int `$controlx`, int `$controly`, int `$anchorx`, int `$anchory` [, int `$targetx`], int `$targety`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfshape->drawcurveto() draws a quadratic curve (using the current line style, set by **swfshape->setline()**) from the current pen position to (*`anchorx`*, *`anchory`*) using (*`controlx`*, *`controly`*) as a control point. That is, head towards the control point, then smoothly turn to the anchor point.

With 6 parameters, it draws a cubic bezier to point (*`targetx`*, *`targety`*) with control points (*`controlx`*, *`controly`*) and (*`anchorx`*, *`anchory`*).

See Also

- SWFShape->drawCurveTo()

SWFShape->drawGlyph()

SWFShape->drawGlyph() -- Draws the first character in the given string into the shape using the glyph definition from the given font

Description

SWFShape

void **drawGlyph** (SWFFont `$font`, string `$character` [, int `$size`])

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

SWFShape->drawLine()

SWFShape->drawLine() -- Draws a line (relative)

Description

SWFShape

void **drawLine** (int `$dx`, int `$dy`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfshape->drawline() draws a line (using the current line style set by **swfshape->setline()**) from the current pen position to displacement (*dx*, *dy*).

Return Values

No value is returned.

See Also

- SWFShape->drawLineTo()

SWFShape->drawLineTo()

SWFShape->drawLineTo() -- Draws a line

Description

SWFShape

void **drawLineTo** (int $x, int $y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfshape->setrightfill() draws a line (using the current line style, set by **swfshape->setline()**) from the current pen position to point (*x*, *y*) in the shape's coordinate space.

Return Values

No value is returned.

See Also

- SWFShape->drawLine()

SWFShape->movePen()

SWFShape->movePen() -- Moves the shape's pen (relative)

Description

SWFShape

void **movePen** (int `$dx`, int `$dy`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfshape->setrightfill() move the shape's pen from coordinates (current x,current y) to (current x + *dx*, current y + *dy*) in the shape's coordinate space.

Return Values

No value is returned.

See Also

- SWFShape->movePenTo()

SWFShape->movePenTo()

SWFShape->movePenTo() -- Moves the shape's pen

Description

SWFShape

void **movePenTo** (int $x, int $y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfshape->setrightfill() move the shape's pen to (*x*, *y*) in the shape's coordinate space.

Return Values

No value is returned.

See Also

- SWFShape->movePen()

SWFShape->setLeftFill()

SWFShape->setLeftFill() -- Sets left rasterizing color

Description

SWFShape

void **setLeftFill** (SWFGradient `$fill`)

void **setLeftFill** (int `$red`, int `$green`, int `$blue` [, int `$a`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

What this nonsense is about is, every edge segment borders at most two fills. When rasterizing the object, it's pretty handy to know what those fills are ahead of time, so the swf format requires these to be specified.

swfshape->setleftfill() sets the fill on the left side of the edge- that is, on the interior if you're defining the outline of the shape in a counter-clockwise fashion. The fill object is an SWFFill object returned from one of the addFill functions above.

This seems to be reversed when you're defining a shape in a morph, though. If your browser crashes, just try setting the fill on the other side.

Shortcut for *swfshape->setleftfill($s->addfill($r, $g, $b [, $a]));*.

Return Values

No value is returned.

See Also

- SWFShape->setRightFill()

SWFShape->setLine()

SWFShape->setLine() -- Sets the shape's line style

Description

SWFShape

void **setLine** (SWFShape `$shape`)

void **setLine** (int `$width`, int `$red`, int `$green`, int `$blue` [, int `$a`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfshape->setline() sets the shape's line style. *`width`* is the line's width. If *`width`* is 0, the line's style is removed (then, all other arguments are ignored). If *`width`* > 0, then line's color is set to *`red`*, *`green`*, *`blue`*. Last parameter *`a`* is optional.

You must declare all line styles before you use them (see example).

Return Values

No value is returned.

Examples

This simple example will draw a big "!#%*@", in funny colors and gracious style.

Example #20 - swfshape->setline() example

```
<?php
$s = new SWFShape();
$f1 = $s->addFill(0xff, 0, 0);
$f2 = $s->addFill(0xff, 0x7f, 0);
$f3 = $s->addFill(0xff, 0xff, 0);
$f4 = $s->addFill(0, 0xff, 0);
$f5 = $s->addFill(0, 0, 0xff);

// bug: have to declare all line styles before you use them
$s->setLine(40, 0x7f, 0, 0);
```

```
$s->setLine(40, 0x7f, 0x3f, 0);
$s->setLine(40, 0x7f, 0x7f, 0);
$s->setLine(40, 0, 0x7f, 0);
$s->setLine(40, 0, 0, 0x7f);

$f = new SWFFont('Techno.fdb');

$s->setRightFill($f1);
$s->setLine(40, 0x7f, 0, 0);
$s->drawGlyph($f, '!');
$s->movePen($f->getWidth('!'), 0);

$s->setRightFill($f2);
$s->setLine(40, 0x7f, 0x3f, 0);
$s->drawGlyph($f, '#');
$s->movePen($f->getWidth('#'), 0);

$s->setRightFill($f3);
$s->setLine(40, 0x7f, 0x7f, 0);
$s->drawGlyph($f, '%');
$s->movePen($f->getWidth('%'), 0);

$s->setRightFill($f4);
$s->setLine(40, 0, 0x7f, 0);
$s->drawGlyph($f, '*');
$s->movePen($f->getWidth('*'), 0);

$s->setRightFill($f5);
$s->setLine(40, 0, 0, 0x7f);
$s->drawGlyph($f, '@');

$m = new SWFMovie();
$m->setDimension(3000,2000);
$m->setRate(12.0);
$i = $m->add($s);
$i->moveTo(1500-$f->getWidth("!#%*@")/2, 1000+$f->getAscent()/2);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

Return Values

No value is returned.

SWFShape->setRightFill()

SWFShape->setRightFill() -- Sets right rasterizing color

Description

SWFShape

void **setRightFill** (SWFGradient `$fill`)

void **setRightFill** (int `$red`, int `$green`, int `$blue` [, int `$a`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Shortcut for *swfshape->setrightfill($s->addfill($r, $g, $b [, $a]));*.

Return Values

No value is returned.

See Also

- SWFShape->setLeftFill()

The SWFSound class

Introduction

SWFSound.

Class synopsis

SWFSound

SWFSound {

/* Methods */

SWFSound __**construct** (string `$filename`, int `$flags`)
}

SWFSound

SWFSound -- Returns a new SWFSound object from given file

Description

SWFSound

SWFSound **__construct** (string `$filename`, int `$flags`)

Warning

This function is currently not documented; only its argument list is available.

The SWFSoundInstance class

Introduction

SWFSoundInstance objects are returned by the SWFSprite->startSound() and SWFMovie->startSound() methods.

Class synopsis

SWFSoundInstance

```
SWFSoundInstance {

    /*  Methods */

    void loopCount ( int $point )

    void loopInPoint ( int $point )

    void loopOutPoint ( int $point )

    void noMultiple ( void )
}
```

SWFSoundInstance->loopCount()

SWFSoundInstance->loopCount() --

Description

SWFSoundInstance

void **loopCount** (int `$point`)

Warning

This function is currently not documented; only its argument list is available.

Parameters

`point`

Return Values

No value is returned.

See Also

- SWFSoundInstance->loopOutPoint()

SWFSoundInstance->loopInPoint()

SWFSoundInstance->loopInPoint() --

Description

SWFSoundInstance

void **loopInPoint** (int `$point`)

Warning

This function is currently not documented; only its argument list is available.

Parameters

`point`

Return Values

No value is returned.

See Also

- SWFSoundInstance->loopOutPoint()

SWFSoundInstance->loopOutPoint()

SWFSoundInstance->loopOutPoint() --

Description

SWFSoundInstance

void **loopOutPoint** (int `$point`)

Warning

This function is currently not documented; only its argument list is available.

Parameters

`point`

Return Values

No value is returned.

See Also

- SWFSoundInstance->loopInPoint()

SWFSoundInstance->noMultiple()

SWFSoundInstance->noMultiple() --

Description

SWFSoundInstance

void **noMultiple** (void)

Warning
This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

The SWFSprite class

Introduction

An SWFSprite is also known as a "movie clip", this allows one to create objects which are animated in their own timelines. Hence, the sprite has most of the same methods as the movie.

Class synopsis

SWFSprite

SWFSprite {

/* Methods */

void **add** (object `$object`)

SWFSprite __**construct** (void)

void **labelFrame** (string `$label`)

void **nextFrame** (void)

void **remove** (object `$object`)

void **setFrames** (int `$number`)

SWFSoundInstance **startSound** (SWFSound `$sount`)

void **stopSound** (SWFSound `$sount`)
}

SWFSprite->add()

SWFSprite->add() -- Adds an object to a sprite

Description

SWFSprite

void **add** (object `$object`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfsprite->add() adds a **swfshape()**, a **swfbutton()**, a **swftext()**, a **swfaction()** or a **swfsprite()** object.

For displayable types (**swfshape()**, **swfbutton()**, **swftext()**, **swfaction()** or **swfsprite()**), this returns a handle to the object in a display list.

Return Values

No value is returned.

SWFSprite->__construct()

SWFSprite->__construct() -- Creates a movie clip (a sprite)

Description

SWFSprite

SWFSprite __**construct** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Creates a new SWFSprite object.

SWFSprite->labelFrame()

SWFSprite->labelFrame() -- Labels frame

Description

SWFSprite

void **labelFrame** (string `$label`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

SWFSprite->nextFrame()

SWFSprite->nextFrame() -- Moves to the next frame of the animation

Description

SWFSprite

void **nextFrame** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfsprite->setframes() moves to the next frame of the animation.

Return Values

No value is returned.

SWFSprite->remove()

SWFSprite->remove() -- Removes an object to a sprite

Description

SWFSprite

void **remove** (object `$object`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfsprite->remove() remove a **swfshape()**, a **swfbutton()**, a **swftext()**, a **swfaction()** or a **swfsprite()** object from the sprite.

Return Values

No value is returned.

SWFSprite->setFrames()

SWFSprite->setFrames() -- Sets the total number of frames in the animation

Description

SWFSprite

void **setFrames** (int `$number`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swfsprite->setframes() sets the total number of frames in the animation to *`numberofframes`*.

Return Values

No value is returned.

SWFSprite->startSound()

SWFSprite->startSound() --

Description

SWFSprite

SWFSoundInstance **startSound** (SWFSound `$sount`)

Warning

This function is currently not documented; only its argument list is available.

See Also

- SWFSprite->stopSound()

SWFSprite->stopSound()

SWFSprite->stopSound() --

Description

SWFSprite

void **stopSound** (SWFSound `$sount`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

See Also

- SWFSprite->startSound()

The SWFText class

Introduction

SWFText.

Class synopsis

SWFText

SWFText {

/* Methods */

void **addString** (string `$string`)

void **addUTF8String** (string `$text`)

void **__construct** (void)

float **getAscent** (void)

float **getDescent** (void)

float **getLeading** (void)

float **getUTF8Width** (string `$string`)

float **getWidth** (string `$string`)

void **moveTo** (int `$x`, int `$y`)

void **setColor** (int `$red`, int `$green`, int `$blue` [, int `$a`])

void **setFont** (string `$font`)

void **setHeight** (int `$height`)

void **setSpacing** (float `$spacing`)

}

SWFText->addString()

SWFText->addString() -- Draws a string

Description

SWFText

void **addString** (string `$string`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftext->addstring() draws the string *`string`* at the current pen (cursor) location. Pen is at the baseline of the text; i.e., ascending text is in the -y direction.

Return Values

No value is returned.

See Also

- SWFText->addUTF8String()

SWFText->addUTF8String()

SWFText->addUTF8String() -- Writes the given text into this SWFText object at the current pen position, using the current font, height, spacing, and color

Description

SWFText

void **addUTF8String** (string `$text`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

See Also

- SWFText->addString()

SWFText->__construct()

SWFText->__construct() -- Creates a new SWFText object

Description

SWFText

void __**construct** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Creates a new SWFText object, fresh for manipulating.

Examples

This simple example will draw a big yellow "PHP generates Flash with Ming" text, on white background.

Example #21 - swftext() example

```
<?php
$f = new SWFFont("Techno.fdb");
$t = new SWFText();
$t->setFont($f);
$t->moveTo(200, 2400);
$t->setColor(0xff, 0xff, 0);
$t->setHeight(1200);
$t->addString("PHP generates Flash with Ming!!");

$m = new SWFMovie();
$m->setDimension(5400, 3600);

$m->add($t);

header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

SWFText->getAscent()

SWFText->getAscent() -- Returns the ascent of the current font at its current size, or 0 if not available

Description

SWFText

float **getAscent** (void)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFText->getDescent()

SWFText->getDescent()

SWFText->getDescent() -- Returns the descent of the current font at its current size, or 0 if not available

Description

SWFText

float **getDescent** (void)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFText->getAscent()

SWFText->getLeading()

SWFText->getLeading() -- Returns the leading of the current font at its current size, or 0 if not available

Description

SWFText

float **getLeading** (void)

Warning
This function is currently not documented; only its argument list is available.

SWFText->getUTF8Width()

SWFText->getUTF8Width() -- calculates the width of the given string in this text objects current font and size

Description

SWFText

float **getUTF8Width** (string `$string`)

Warning
This function is currently not documented; only its argument list is available.

See Also

- SWFText->getWidth()

SWFText->getWidth()

SWFText->getWidth() -- Computes string's width

Description

SWFText

float **getWidth** (string `$string`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Returns the rendered width of the *`string`* at the text object's current font, scale, and spacing settings.

See Also

- SWFText->getUTF8Width()

SWFText->moveTo()

SWFText->moveTo() -- Moves the pen

Description

SWFText

void **moveTo** (int $x, int $y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftext->moveto() moves the pen (or cursor, if that makes more sense) to (*x*, *y*) in text object's coordinate space. If either is zero, though, value in that dimension stays the same. Annoying, should be fixed.

Return Values

No value is returned.

SWFText->setColor()

SWFText->setColor() -- Sets the current text color

Description

SWFText

void **setColor** (int `$red`, int `$green`, int `$blue` [, int `$a`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Changes the current text color.

Parameters

These parameters are integers between 0 and 255 or hexadecimals between *0x00* and *0xFF*:

`red`
: Value of red component

`green`
: Value of green component

`blue`
: Value of blue component

`a`
: Value of alpha component

Return Values

No value is returned.

SWFText->setFont()

SWFText->setFont() -- Sets the current font

Description

SWFText

void **setFont** (string `$font`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftext->setfont() sets the current font to `font`.

Return Values

No value is returned.

SWFText->setHeight()

SWFText->setHeight() -- Sets the current font height

Description

SWFText

void **setHeight** (int `$height`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftext->setheight() sets the current font height to *`height`*. Default is 240.

Return Values

No value is returned.

SWFText->setSpacing()

SWFText->setSpacing() -- Sets the current font spacing

Description

SWFText

void **setSpacing** (float `$spacing`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftext->setspacing() sets the current font spacing to *`spacing`*. Default is 1.0. 0 is all of the letters written at the same point. This doesn't really work that well because it inflates the advance across the letter, doesn't add the same amount of spacing between the letters. I should try and explain that better, prolly. Or just fix the damn thing to do constant spacing. This was really just a way to figure out how letter advances work, anyway.. So nyah.

Return Values

No value is returned.

The SWFTextField class

Introduction

SWFTextField.

Class synopsis

SWFTextField

SWFTextField {

/* Methods */

void **addChars** (string `$chars`)

void **addString** (string `$string`)

void **align** (int `$alignement`)

SWFTextField __**construct** ([int `$flags`])

void **setBounds** (int `$width`, int `$height`)

void **setColor** (int `$red`, int `$green`, int `$blue` [, int `$a`])

void **setFont** (string `$font`)

void **setHeight** (int `$height`)

void **setIndentation** (int `$width`)

void **setLeftMargin** (int `$width`)

void **setLineSpacing** (int `$height`)

void **setMargins** (int `$left`, int `$right`)

void **setName** (string `$name`)

void **setPadding** (float `$padding`)

void **setRightMargin** (int `$width`)

```
}
```

SWFTextField->addChars()

SWFTextField->addChars() -- adds characters to a font that will be available within a textfield

Description

SWFTextField

void **addChars** (string `$chars`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

See Also

- SWFTextField->addString()

SWFTextField->addString()

SWFTextField->addString() -- Concatenates the given string to the text field

Description

SWFTextField

void **addString** (string `$string`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->setname() concatenates the string *`string`* to the text field.

Return Values

No value is returned.

SWFTextField->align()

SWFTextField->align() -- Sets the text field alignment

Description

SWFTextField

void **align** (int `$alignement`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->align() sets the text field alignment to *`alignement`*. Valid values for *`alignement`* are : SWFTEXTFIELD_ALIGN_LEFT, SWFTEXTFIELD_ALIGN_RIGHT, SWFTEXTFIELD_ALIGN_CENTER and SWFTEXTFIELD_ALIGN_JUSTIFY.

Return Values

No value is returned.

SWFTextField->__construct()

SWFTextField->__construct() -- Creates a text field object

Description

SWFTextField

SWFTextField **__construct** ([int `$flags`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield() creates a new text field object. Text Fields are less flexible than **swftext()** objects- they can't be rotated, scaled non-proportionally, or skewed, but they can be used as form entries, and they can use browser-defined fonts.

The optional flags change the text field's behavior. It has the following possibles values :

- SWFTEXTFIELD_DRAWBOX draws the outline of the textfield
- SWFTEXTFIELD_HASLENGTH
- SWFTEXTFIELD_HTML allows text markup using HTML-tags
- SWFTEXTFIELD_MULTILINE allows multiple lines
- SWFTEXTFIELD_NOEDIT indicates that the field shouldn't be user-editable
- SWFTEXTFIELD_NOSELECT makes the field non-selectable
- SWFTEXTFIELD_PASSWORD obscures the data entry
- SWFTEXTFIELD_WORDWRAP allows text to wrap

Flags are combined with the bitwise OR operation. For example,

```
<?php
$t = newSWFTextField(SWFTEXTFIELD_PASSWORD | SWFTEXTFIELD_NOEDIT);
?>
```

creates a totally useless non-editable password field.

SWFTextField has the following methods : **swftextfield->setfont()**, **swftextfield->setbounds()**, **swftextfield->align()**, **swftextfield->setheight()**, **swftextfield->setleftmargin()**, **swftextfield->setrightmargin()**, **swftextfield->setmargins()**, **swftextfield->setindentation()**,

swftextfield->setlinespacing(), **swftextfield->setcolor()**, **swftextfield->setname()** and **swftextfield->addstring()**.

SWFTextField->setBounds()

SWFTextField->setBounds() -- Sets the text field width and height

Description

SWFTextField

void **setBounds** (int `$width`, int `$height`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->setbounds() sets the text field width to *`width`* and height to *`height`*. If you don't set the bounds yourself, Ming makes a poor guess at what the bounds are.

Return Values

No value is returned.

SWFTextField->setColor()

SWFTextField->setColor() -- Sets the color of the text field

Description

SWFTextField

void **setColor** (int `$red`, int `$green`, int `$blue` [, int `$a`])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->setcolor() sets the color of the text field. Default is fully opaque black. Color is represented using RGB system.

Parameters

These parameters are integers between 0 and 255 or hexadecimals between *0x00* and *0xFF*:

`red`
: Value of red component

`green`
: Value of green component

`blue`
: Value of blue component

`a`
: Value of alpha component

Return Values

No value is returned.

SWFTextField->setFont()

SWFTextField->setFont() -- Sets the text field font

Description

SWFTextField

void **setFont** (string `$font`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->setfont() sets the text field font to the [browser-defined?] *`font`* font.

Return Values

No value is returned.

SWFTextField->setHeight()

SWFTextField->setHeight() -- Sets the font height of this text field font

Description

SWFTextField

void **setHeight** (int `$height`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->setheight() sets the font height of this text field font to the given height *`height`*. Default is 240.

Return Values

No value is returned.

SWFTextField->setIndentation()

SWFTextField->setIndentation() -- Sets the indentation of the first line

Description

SWFTextField

void **setIndentation** (int `$width`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->setindentation() sets the indentation of the first line in the text field, to *`width`*.

Return Values

No value is returned.

SWFTextField->setLeftMargin()

SWFTextField->setLeftMargin() -- Sets the left margin width of the text field

Description

SWFTextField

void **setLeftMargin** (int `$width`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->setleftmargin() sets the left margin width of the text field to *`width`*. Default is 0.

Return Values

No value is returned.

SWFTextField->setLineSpacing()

SWFTextField->setLineSpacing() -- Sets the line spacing of the text field

Description

SWFTextField

void **setLineSpacing** (int `$height`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->setlinespacing() sets the line spacing of the text field to the height of *`height`*. Default is 40.

Return Values

No value is returned.

SWFTextField->setMargins()

SWFTextField->setMargins() -- Sets the margins width of the text field

Description

SWFTextField

void **setMargins** (int `$left`, int `$right`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->setmargins() set both margins at once, for the man on the go.

Return Values

No value is returned.

SWFTextField->setName()

SWFTextField->setName() -- Sets the variable name

Description

SWFTextField

void **setName** (string `$name`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->setname() sets the variable name of this text field to *`name`*, for form posting and action scripting purposes.

Return Values

No value is returned.

SWFTextField->setPadding()

SWFTextField->setPadding() -- Sets the padding of this textfield

Description

SWFTextField

void **setPadding** (float `$padding`)

Warning

This function is currently not documented; only its argument list is available.

Return Values

No value is returned.

SWFTextField->setRightMargin()

SWFTextField->setRightMargin() -- Sets the right margin width of the text field

Description

SWFTextField

void **setRightMargin** (int `$width`)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

swftextfield->setrightmargin() sets the right margin width of the text field to *`width`*. Default is 0.

Return Values

No value is returned.

The SWFVideoStream class

Introduction

SWFVideoStream.

Class synopsis

SWFVideoStream

SWFVideoStream {

/* Methods */

SWFVideoStream __**construct** ([string `$file`])

int **getNumFrames** (void)

void **setDimension** (int `$x`, int `$y`)
}

SWFVideoStream->__construct()

SWFVideoStream->__construct() -- Returns a SWFVideoStream object

Description

SWFVideoStream

SWFVideoStream __**construct** ([string `$file`])

Warning

This function is currently not documented; only its argument list is available.

SWFVideoStream->getNumFrames()

SWFVideoStream->getNumFrames() -- Returns the number of frames in the video

Description

SWFVideoStream

int **getNumFrames** (void)

This function returns the number of video-frames of a SWFVideoStream.

Return Values

Returns the number of frames, as an integer

SWFVideoStream->setDimension()

SWFVideoStream->setDimension() -- Sets video dimension

Description

SWFVideoStream

void **setDimension** (int $x, int $y)

Sets the width and height for streamed videos.

Parameters

x
: Width in pixels.

y
: Height in pixels.

Return Values

No value is returned.