# pChart

Official documentation

# pChart

Official documentation

All persons, companies, email addresses, and URLs listed in the examples are purely fictitious and any resemblance to existing persons, companies, email addresses, or URLs is purely coincidental Mention of third-party products and URLs is for informational purposes only and constitutes neither an endorsement nor a recommendation.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS 'AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Document generated automatically on 26/01/2011 from http://wiki.pchart.net

# Basic Syntax

This page summarize some of the basic concepts of pCharts graphics.

## Include the library

pChart is composed of classes, to add the objects definition in you codes, you must add some includes in your projects headers. In PHP, adding external files is pretty easy using the **include()** function. Sometimes, you may prefer the **require_once()** function.

All your scripts will start by the following lines :

Code sample

```
/* Include all the classes */
include("class/pDraw.class.php");
include("class/pImage.class&.phpquot;);
include("class/pData.class.php");
```

Note that if you need to use sessions, they must be initiated before this line to avoid messages like :

```
Cannot send session cookie - headers already sent by [..]
```

## Prepare your dataset

There are many ways to populate your dataset. The easiest way is to use the **addPoints** method of the pData class. With this method you can add values one by one or directly provide an array containing the data that you want to chart.

Code sample

```
/* Create your dataset object */
$myData = new pData();

/* Add data in your dataset */
$myData->addPoints(array(VOID,3,4,3,5));
```

In this example we are providing 5 values, the 4[SUP]th[/SUP] one is missing and replaced by the VOID keyword.

## Create your pChart object

Next step is to create your drawing object. The following will create a 700x230px picture :

Code sample

```
/* Create a pChart object and associate your dataset */
$myPicture = new pImage(700,230,$myData);
```

## Define your chart area

Before calling any scaling or charting function, you must define the chart area boundaries.

Code sample

```
/* Define the boundaries of the graph area */
$myPicture->setGraphArea(60,40,670,190);
```

## Choose a nice font

The default font may look a bit "agressive", hopefully the web is full of free nice-looking TTF files. You can set a different font, all the function that are writting text will use it.

Code sample

```
/* Choose a nice font */
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>11));
```

### Compute the scale

Scale computing is mandatory before calling the charting functions. You can define which scaling mode to use (classical, add values) and the way you want to display and rotate the axis.

Code sample

```
/* Draw the scale, keep everything automatic */
$myPicture->drawScale();
```

### Draw your chart

It‘s now time to choose which kind of chart you‘ll draw. We‘ll use a spline here :

Code sample

```
/* Draw the scale, keep everything automatic */
$myPicture->drawSplineChart();
```

### All done! Let‘s render to the browser

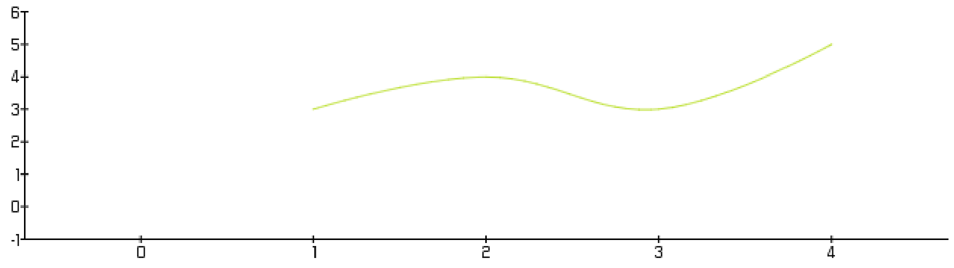
You can now finalize your graph and send it to the user browser :

Code sample

```
/* Build the PNG file and send it to the web browser */
$myPicture->Stroke();
```

## Put it all together

As you‘ll see, this is a very basic example of what pChart can do for you. When you‘ll make your own scripts, you‘ll see that the most difficult thing is to figure out how you want to represent your data. pChart can do a lot of things for you, we invite you to walk in this documentation to use the full potential of this library.



Code sample

```
/* Include all the classes */
include("class/pDraw.class.php");
include("class/pImage.class.php");
include("class/pData.class.php");

/* Create your dataset object */
$myData = new pData();


/* Add data in your dataset */
$myData->addPoints(array(VOID,3,4,3,5));

/* Create a pChart object and associate your dataset */
$myPicture = new pImage(700,230,$myData);

/* Choose a nice font */
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>11));

/* Define the boundaries of the graph area */
```

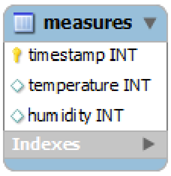
```
$myPicture->setGraphArea(60,40,670,190);

/* Draw the scale, keep everything automatic */
$myPicture->drawScale();

/* Draw the scale, keep everything automatic */
$myPicture->drawSplineChart();

/* Build the PNG file and send it to the web browser */
$myPicture->Render("basic.png");
```

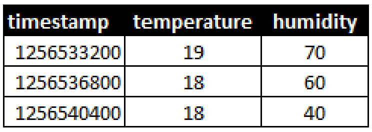
# MySQL Integration

Data can be retrieved from various sources. In almost all cases data will come from an online database but you can imagine a lot of alternative sources like flat files, XML, scripts,.. You'll find here some example on how to interact with the pData class.

## Retrieving data from MySQL

MySQL is a powerfull open source database engine. This is the perfect companion for the pChart library but of course you can use any other engine! A lot of databases connectors are shipped with PHP, the syntax to use will depends of the one you'll choose. The following example shows how to retrieve data from a MySQL database. In this example we'll use a table called **measures** containing history data for **temperature** and **humidity**.





| timestamp | temperature | humidity |
|---|---|---|
| 1256533200 | 19 | 70 |
| 1256536800 | 18 | 60 |
| 1256540400 | 18 | 40 |

We want here to extract the measured values and the associated timestamp to build an history chart. First step will be to create your pData object and connect to your database :

Code sample

```
/* Include the pData class */
include("class/pData.class.php");

/* Create the pData object */
$MyData = new pData();

/* Connect to the MySQL database */
$db = mysql_connect("localhost", "dbuser", "dbpwd");
mysql_select_db("pchart",$db);
```

Now we want to extract each column and bind it to one data serie. This can be done in many ways, the simplest (and not optimised) would be :

Code sample

```
/* Build the query that will returns the data to graph */
$Requete = "SELECT * FROM `measures`";
$Result  = mysql_query($Requete,$db);
while($row = mysql_fetch_array($Result))
 {
  /* Get the data from the query result */
  $timestamp   = $row["timestamp"];
  $temperature = $row["temperature"];
  $humidity    = $row["humidity"];

  /* Save the data in the pData array */
  $myData->addPoints($timestamp,"Timestamp");
  $myData->addPoints($temperature,"Temperature");
  $myData->addPoints($humidity,"Humidity");
 }
```

But you must prefer to reduce the number of calls to the pData class in order to greatly improve your script speed. As the **addPoints** method can handle arrays, it would be a nice way to push all the data in a array and pass it to the pData class in only one call :

Code sample

```
/* Build the query that will returns the data to graph */
$Requete = "SELECT * FROM `measures`";
$Result  = mysql_query($Requete,$db);
$timestamp=""; $temperature=""; $humidity="";
while($row = mysql_fetch_array($Result))
 {
  /* Push the results of the query in an array */
  $timestamp[]   = $row["timestamp"];
  $temperature[] = $row["temperature"];
  $humidity[]    = $row["humidity"];
 }

/* Save the data in the pData array */
$myData->addPoints($timestamp,"Timestamp");
$myData->addPoints($temperature,"Temperature");
$myData->addPoints($humidity,"Humidity");
```

Now we want to use the data of the timestamp column as the abscissa labels. We also want to display it in a readable way :

Code sample

```
/* Put the timestamp column on the abscissa axis */
$myData->setAbscissa("Timestamp");
```

As we want to chart here to values of different units, we must create a second axis and associate it the humidity :

Code sample

```
/* Associate the "Humidity" data serie to the second axis */
$myData->setSerieOnAxis("Humidity", 1);

/* Name this axis "Time" */
$myData->setXAxisName("Time");

/* Specify that this axis will display time values */
$myData->setXAxisDisplay(AXIS_FORMAT_TIME,"H:i");
```

Now we can make some makeup by specifying the units and axis names :

Code sample

```
/* First Y axis will be dedicated to the temperatures */
$myData->setAxisName(0,"Temperature");
$myData->setAxisUnit(0,"°C");

/* Second Y axis will be dedicated to humidity */
$myData->setAxisName(1,"Humidity");
$myData->setAxisUnit(0,"%");
```

Information

Your dataset is now ready to be charted!

## Format Array

Almost all drawing fonctions can be tunned by manually feeding rendering parameters like color, transparency,.. This parameters are given to the function within a 2 dimensionnal array.

## Feeding the array

Arrays are easy to use in PHP. In this descriptive array, we have to provide a couple of values : the name of the parameters will be used as a key and the value will be provided rawly :

Code sample

```
$myArray = array("Alpha"=>50);
```

..this will associate the value 50 with the parameter Alpha. Basically, drawing function will at least require a color that will be composed of 3 values R,G,B :

Code sample

```
$myArray = array("R"=>191,"G"=>215,"B"=>59);
```

You can also use a less compact method to fill the array :

Code sample

```
$myArray["R"]=191;
$myArray["G"]=215;
$myArray["B"]=59;
```

## Options may change

When calling a drawing function refer to its documentation. The drawFilledCircle for example have many parameters that can be tuned :

- The fill color can be set with R, G, B.
- The border color can be set with BorderR, BorderG, BorderB.
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.
- The alpha transparency factor can be set with Alpha

This may lead in a long configuration array :

Code sample

```
$myOptions=array("R"=>191,"G"=>215,"B"=>59,"BorderR"=>211,"BorderG"=>235,"BorderB"=>79,"Alpha"=>80);
```

# Settings functions

setShadow
setGraphArea
setPalette
setFontProperties

# setShadow - Enable / Disable shadow support

Shadow support is one key feature of the 2.0 trunk. This rendering option can globally be applied to all drawing functions. All the shadow parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

[ERR]Only enable shadows when visually required. Using shadows will slow down the rendering speed[/ERR]

## Calling this function

```
setShadow($Enabled=TRUE,$Format="");
```

Where :

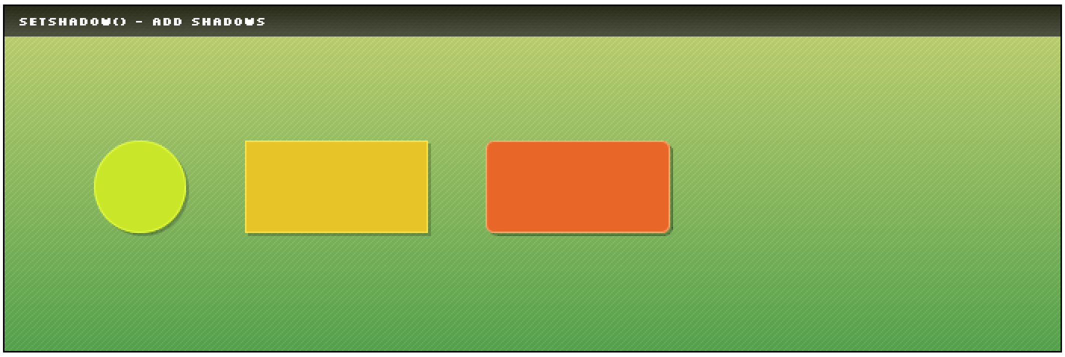
- Enabled is either TRUE if you want to activate shadows of FALSE.
- Format is an array containing the drawing parameters of the arrow.

## Customisation array - Tune up your shadows!

It is possible to customize the shadow rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the shadow will be drawn black with an offset of (+2,+2) and a transparency of 20%.

- The shadow offset can be set using X, Y.
- The shadow color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.

## Sample script



Code sample

```
/* Enable shadow support */
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20);

/* Draw a filled circle */
$formSettings = array("R"=>201,"G"=>230,"B"=>40,"Alpha"=>100,"Surrounding"=>30);
$myPicture->drawFilledCircle(90,120,30,$formSettings);

/* Draw a filled rectangle */
$formSettings = array("R"=>231,"G"=>197,"B"=>40,"Alpha"=>100,"Surrounding"=>30);
$myPicture->drawFilledRectangle(160,90,280,150,$formSettings);

/* Draw a filled rectangle with rounded corners */
$formSettings = array("R"=>231,"G"=>102,"B"=>40,"Alpha"=>100,"Surrounding"=>30);
$myPicture->drawRoundedFilledRectangle(320,90,440,150,5,$formSettings);
```

This will draw various basic shapes with a shadow.

# setGraphArea - Define the bounding box of your charts

This function allows you to define the bounding box that will be used for all charting function that will be called. It is mandatory to define the coordinates of this rectangular area before calling any chart drawing component.

Be careful

This function must be called before drawing any charts.

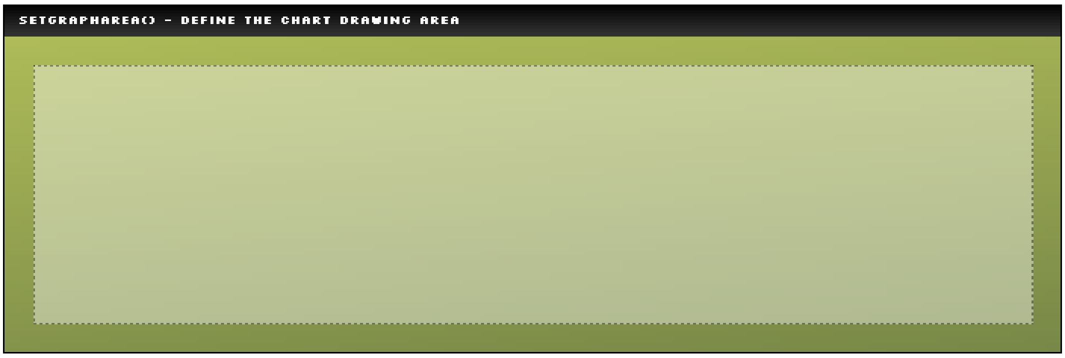
## Calling this function

```
setGraphArea($X1,$Y1,$X2,$Y2);
```

Where :

- X1,Y1 is the top left corner of the bounding box.
- X2,Y2 is the borrom right corner of the bounding box.

## Sample script



Code sample

```
/* Define the chart area*/
$myPicture->setGraphArea(20,40,680,210);
```

This will define the charting area at position (20,40)-(680,210). Note that this command will not display anything, the transparent box in the sample picture is just here to help you figuring out where will be set the graph area.

# setPalette - Associate color to your Data series

This function allows you to modify the series color schemes by associating a color to one serie. Parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
setPalette($Serie,$Format="");
```

Where :

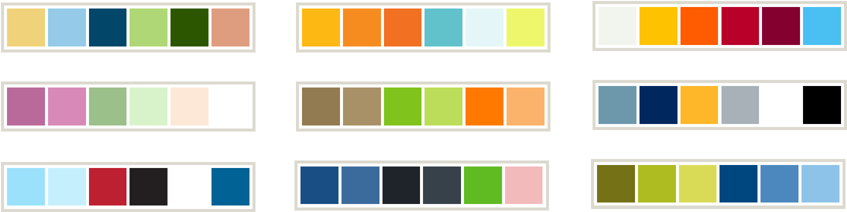
- Serie is the name of the serie that you want to customize.
- Format is an array containing the description of what you want to achieve.

## Customisation array - Choose your color!

It is possible to specify the colors and alpha transparency of the given serie with this array. If nothing is specified, the serie will be drawn black with no transparency.

- The font color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.

A lot of sites are providing nice looking color schemes on internet. It is important to keep in mind that the color must be visible on the background you‘ve choosen and that all series must be visually different.

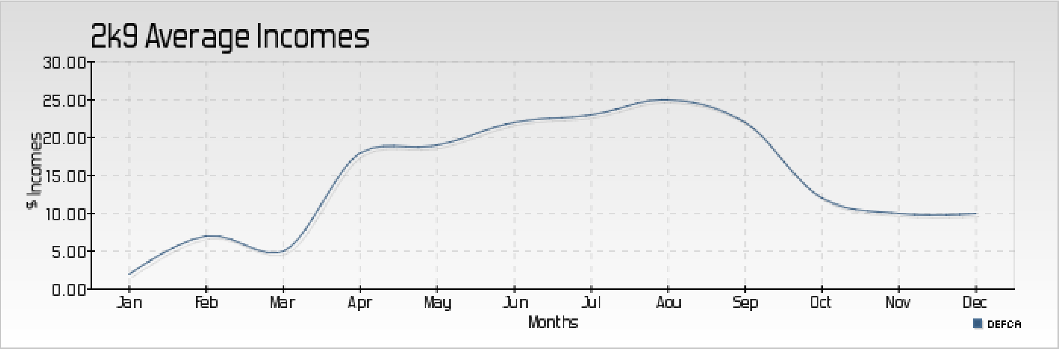


## Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(24,25,26,25,25),"My Serie 1");

/* Draw serie 1 in red with a 80% opacity */
$serieSettings = array("R"=>229,"G"=>11,"B"=>11,"Alpha"=>80);
$MyData->setPalette("My Serie 1",$serieSettings);
```

## Sample script #2



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(2,7,5,18,19,22,23,25,22,12,10,10),"DEFCA");
$MyData->setAxisName(0,"$ Incomes");
$MyData->setAxisDisplay(0,AXIS_FORMAT_CURRENCY);
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun","Jul","Aou","Sep","Oct","Nov","Dec"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");
$MyData->setPalette("DEFCA",array("R"=>55,"G"=>91,"B"=>127));

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,array("StartR"=>220,"StartG"=>220,"StartB"=>220,"EndR"=>255,"EndG"=>255,"En
dB"=>255,"Alpha"=>100));
```

```
$myPicture->drawRectangle(0,0,699,229,array("R"=>200,"G"=>200,"B"=>200));


/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));
$myPicture->drawText(60,35,"2k9 Average Incomes",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMLEFT));

/* Do some cosmetic and draw the chart */
$myPicture->setGraphArea(60,40,670,190);
$myPicture->drawFilledRectangle(60,40,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("GridR"=>180,"GridG"=>180,"GridB"=>180));
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawSplineChart();
$myPicture->setShadow(FALSE);

/* Write the chart legend */
$myPicture->drawLegend(643,210,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawSimple.png");
```

# setFontProperties - Set default fonts properties

This function allows you to specify the default settings for all text outputs on yout pictures. As you'll notice, some functions can override this default settings but if not explicitely specified, default settings will be used. Parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
setFontProperties($Format="");
```

Where :

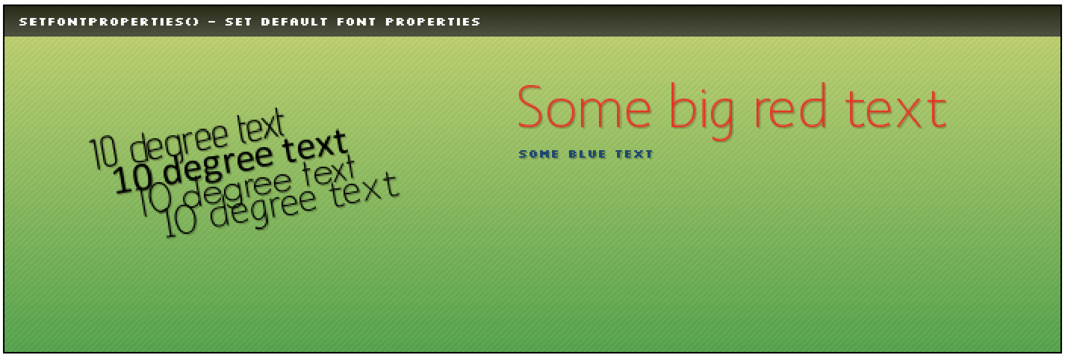
- Format is an array containing the drawing parameters of the arrow.

## Customisation array - Tune up your texts!

It is possible to customize the fonts rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the texts will be written in black with the "GeosansLight.ttf" font file.

- The font name can be set with FontName.
- The font size can be set with FontSize.
- The font color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.

## Sample script



Code sample

```
/* Enable shadow support */
```

```
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20);

/* Force the font properties and color */
$myPicture->setFontProperties(array("FontName"=>"fonts/MankSans.ttf","FontSize"=>30,"R"=>231,"G"=>50,"B"=>36));
$myPicture->drawText(340,90,"Some big red text");

/* Force the font properties and color */
$myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6,"R"=>29,"G"=>70,"B"=>111));
$myPicture->drawText(340,100,"Some blue text");
```

This will write some text areas while forcing the font file name & color with the **setFontProperties()** function.

# Drawing functions

# drawText - Write text

This function allows you to write text on your pictures. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the [Format array guide](). Lines like all simple drawing functions are supporting anti-alias and shadows.

This function is also used internally by some drawing functions.

Information

This function was a bit tricky to write as the GD function imagettfbbox isn‘t accurate when rotating text. The workaround was to re-create it. You can use it calling the getTextBox function.

## Calling this function

```
drawText($X,$Y,$Text,$Format="")
```

Where :

- X,Y are the coordinate of the text (see the Align option).
- Text is the text that will be written.
- Format is an array containing the drawing parameters of the arrow.

This function returns an array or arrays with the coordinates of the surrounding box. This array can be accessed through the internal **FontBox** variable. You can access the points values with FontBox[0-3][0-1], the first value is the index of the point, the second is either 0 for the X coordinate and 1 of the Y one.

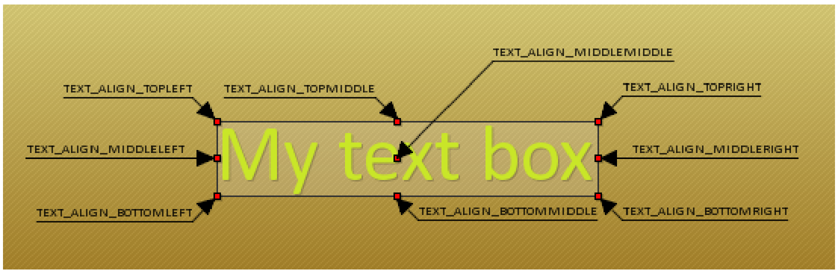
Code sample

```
/* Write some text */
$myPicture->drawText(60,115,"10 degree text",$TextSettings);

/* Print the coordinates of the top left corner of the written text */
echo "X = ".$myPicture->FontBox[0][0].", Y = ".$myPicture->FontBox[0][1];
```

## Customisation array - Tune up your line!

It is possible to customize the line rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the text will be written black with no transparency. The text will be aligned using the bottom left corner. (TEXT_ALIGN_BOTTOMLEFT)

- The line color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.
- The angle can be set with Angle.
- The font name can be overridden with FontName.
- The font size can be overridden with FontSize.
- The text alignment can be set with Align. Depending of the value you‘ll put, the X,Y coordinates of the text will be translated to the red marker, you can see bellow all the alignment options :



Information

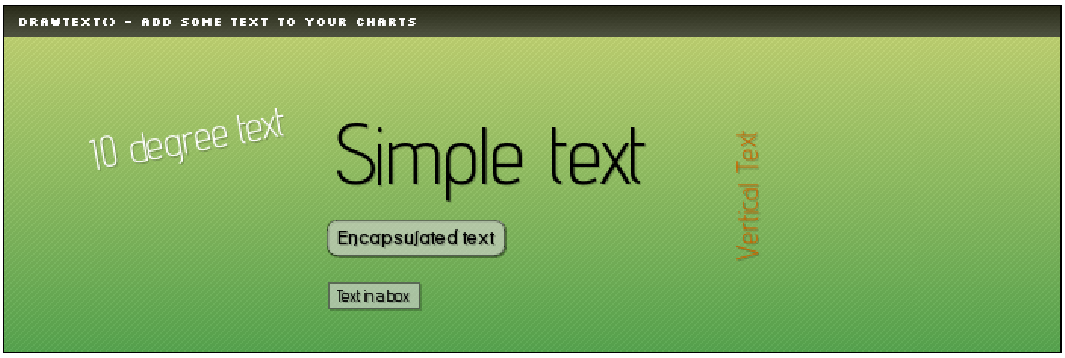
The alignment is computed based on the text size and angle, this means that even if you rotate your text, the control points will be automatically re-computed.

It is possible to draw a filled rectangle under the text, following parameters can be tunned :

- To enable this feature, set DrawBox to TRUE.
- You can specify the box padding with BorderOffset. (default is 6)

- The background color can be set with BoxR, BoxG, BoxB.
- The background alpha can be set with BoxAlpha.
- You can draw rounded corners setting BoxRounded to TRUE.
- The corner radius can be set withRoundedRadius.
- You can specify if the box will have a border setting DrawBoxBorder to TRUE.
- You can specify a border surrounding factor with BoxSurrounding.
- The border color can be set with BoxBorderR, BoxBorderG, BoxBorderB.
- The border alpha can be set with BoxBorderAlpha.

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(700,230);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>80));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawText() - add some text to your charts",array("R"=>255,"G"=>255,"B"=>255));

/* Enable shadow computing */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* Write some text */
$myPicture->setFontProperties(array("FontName"=>"../fonts/advent_light.ttf","FontSize"=>20));
$TextSettings = array("R"=>255,"G"=>255,"B"=>255,"Angle"=>10);
```

```
$myPicture->drawText(60,115,"10 degree text",$TextSettings);

/* Write some text */
$TextSettings = array("R"=>0,"G"=>0,"B"=>0,"Angle"=>0,"FontSize"=>40);
$myPicture->drawText(220,130,"Simple text",$TextSettings);

/* Write some text */
$TextSettings = array("R"=>200,"G"=>100,"B"=>0,"Angle"=>90,"FontSize"=>14);
$myPicture->drawText(500,170,"Vertical Text",$TextSettings);

/* Write some text */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Bedizen.ttf","FontSize"=>6));
$TextSettings = array("DrawBox"=>TRUE,"BoxRounded"=>TRUE,"R"=>0,"G"=>0,"B"=>0,"Angle"=>0,"FontSize"=>10);
$myPicture->drawText(220,160,"Encapsulated text",$TextSettings);

/* Write some text */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>6));
$TextSettings = array("DrawBox"=>TRUE,"R"=>0,"G"=>0,"B"=>0,"Angle"=>0,"FontSize"=>10);
$myPicture->drawText(220,195,"Text in a box",$TextSettings);

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawText.png");
```

## drawGradientArea - Rectangular gradient area

This function allows you to draw rectangular gradient areas. Most of the time, you‘ll use this function to build your image background. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide. This functions are supporting anti-alias and shadows.

### Calling this function

```
drawGradientArea($X1,$Y1,$X2,$Y2,$Direction,$Format="");
```

Where :

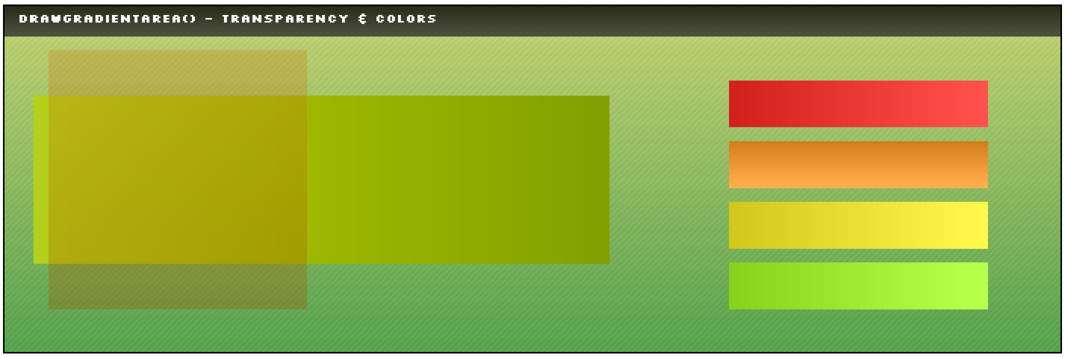
- X1,Y1 are the start coordinate of the rectangle.
- X2,Y2 are the end coordinate of the rectangle.
- Direction is either DIRECTION_VERTICAL or DIRECTION_HORIZONTAL .
- Format is an array containing the drawing parameters.

### Customisation array - Tune up your gradient area!

It is possible to customize the rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the gradient will be drawn with grey levels and no transparency.

- The start color can be set with StartR, StartG, StartB.
- The end color can be set with EndR, EndG, EndB.
- The alpha transparency factor can be set with Alpha.
- If Levels is specified, this will override the end colors by adding this value to the start colors. Levels can be either a positive or negative integer.

### Sample script



Code sample

```
/* Left areas*/
$GradientSettings = array("StartR"=>181,"StartG"=>209,"StartB"=>27,"Alpha"=>100,"Levels"=>-50);
$myPicture->drawGradientArea(20,60,400,170,DIRECTION_HORIZONTAL,$GradientSettings);

$GradientSettings = array("StartR"=>209,"StartG"=>134,"StartB"=>27,"Alpha"=>30,"Levels"=>-50);
$myPicture->drawGradientArea(30,30,200,200,DIRECTION_VERTICAL,$GradientSettings);

 /* Right areas*/
$GradientSettings = array("StartR"=>209,"StartG"=>31,"StartB"=>27,"Alpha"=>100,"Levels"=>50);
$myPicture->drawGradientArea(480,50,650,80,DIRECTION_HORIZONTAL,$GradientSettings);

$GradientSettings = array("StartR"=>209,"StartG"=>125,"StartB"=>27,"Alpha"=>100,"Levels"=>50);
$myPicture->drawGradientArea(480,90,650,120,DIRECTION_VERTICAL,$GradientSettings);

$GradientSettings = array("StartR"=>209,"StartG"=>198,"StartB"=>27,"Alpha"=>100,"Levels"=>50);
$myPicture->drawGradientArea(480,130,650,160,DIRECTION_HORIZONTAL,$GradientSettings);

$GradientSettings = array("StartR"=>134,"StartG"=>209,"StartB"=>27,"Alpha"=>100,"Levels"=>50);
$myPicture->drawGradientArea(480,170,650,200,DIRECTION_HORIZONTAL,$GradientSettings);
```

This will draw 5 gradient areas with different directions and alpha factors.

# drawCircle - Circles

This function allows you to draw circles. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide. This functions are supporting anti-alias and shadows.

## Calling this function

```
drawCircle($X1,$Y1,$Width,$Height,$Format="");
```

Where :

- X1,Y1 are the center coordinate of the circle.
- Width is the width of the circle.
- Height is the heightof the circle.
- Format is an array containing the drawing parameters.

Information

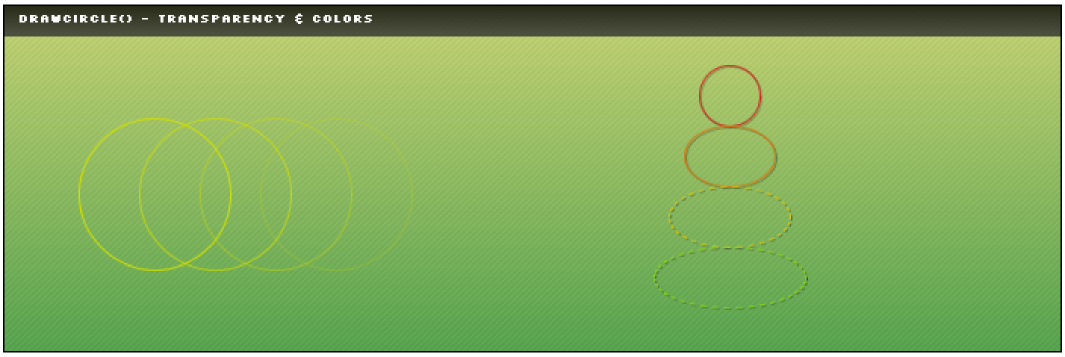
This function can draw an ellipse by using different values for width and height.

## Customisation array - Tune up your circle!

It is possible to customize the circle rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the circles will be drawn black with no transparency.

- The border color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.
- You can draw dashed lines using the Ticks parameter.

### Sample script



Code sample

```
/* Left circles with different alpha transparency */
$myPicture->drawCircle(100,125,50,50,array("R"=>213,"G"=>226,"B"=>0,"Alpha"=>100));
$myPicture->drawCircle(140,125,50,50,array("R"=>213,"G"=>226,"B"=>0,"Alpha"=>70));
$myPicture->drawCircle(180,125,50,50, array("R"=>213,"G"=>226,"B"=>0,"Alpha"=>40));
$myPicture->drawCircle(220,125,50,50,array("R"=>213,"G"=>226,"B"=>0,"Alpha"=>20));

/* Active shadow support */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* Right circles */
$CircleSettings = array("R"=>209,"G"=>31,"B"=>27,"Alpha"=>100);
$myPicture->drawCircle(480,60,20,20,$CircleSettings );

$CircleSettings = array("R"=>209,"G"=>125,"B"=>27,"Alpha"=>100);
$myPicture->drawCircle(480,100,30,20,$CircleSettings );

$CircleSettings = array("R"=>209,"G"=>198,"B"=>27,"Alpha"=>100,"Ticks"=>4);
$myPicture->drawCircle(480,140,40,20,$CircleSettings );

$CircleSettings = array("R"=>134,"G"=>209,"B"=>27,"Alpha"=>100,"Ticks"=>4);
$myPicture->drawCircle(480,180,50,20,$CircleSettings );
```

This will draw some transparency test on the left side of the picture and 4 colored boxes on the right. Shadow support is enabled on a (+1,+1) basis.

# drawRoundedRectangle - boxes with rounded corners

This function allows you to draw rectangles with rounded corners. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide. This functions are supporting anti-alias and shadows.

### Calling this function

```
drawRoundedRectangle($X1,$Y1,$X2,$Y2,$Radius,$Format="");
```

Where :

- X1,Y1 are the start coordinate of the rectangle.
- X2,Y2 are the end coordinate of the rectangle.
- Radius is the radius of the rounded corners.
- Format is an array containing the drawing parameters.

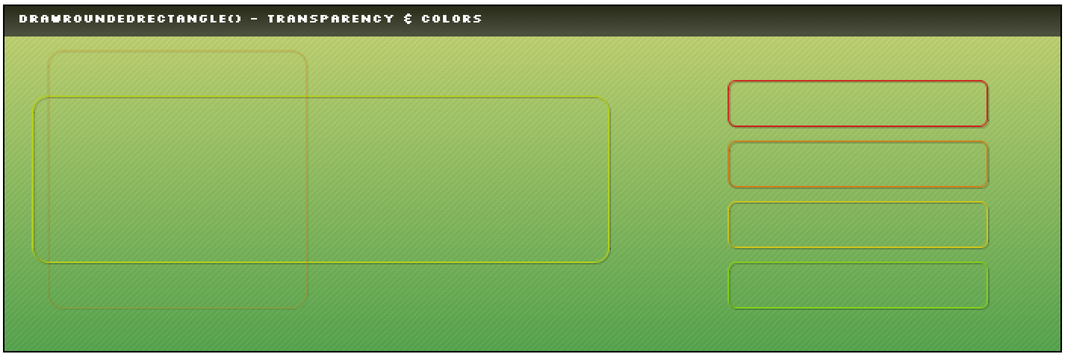
### Customisation array - Tune up your box!

It is possible to customize the rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the rectangle will be drawn black with no transparency.

- The line color can be set with R, G, B.

- The alpha transparency factor can be set with Alpha.

## Sample script



Code sample

```
/* Enable shadow support */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* Left green box */
$RectangleSettings = array("R"=>181,"G"=>209,"B"=>27,"Alpha"=>100);
$myPicture->drawRoundedRectangle(20,60,400,170,10,$RectangleSettings);

/* Left orange surrounding box with an alpha factor of 30% */
$RectangleSettings = array("R"=>209,"G"=>134,"B"=>27,"Alpha"=>30);
$myPicture->drawRoundedRectangle(30,30,200,200,10,$RectangleSettings);

/* Right Red box */
$RectangleSettings = array("R"=>209,"G"=>31,"B"=>27,"Alpha"=>100);
$myPicture->drawRoundedRectangle(480,50,650,80,5,$RectangleSettings);

/* Right Orange box */
$RectangleSettings = array("R"=>209,"G"=>125,"B"=>27,"Alpha"=>100);
$myPicture->drawRoundedRectangle(480,90,650,120,5,$RectangleSettings);

 /* Right Yellow box */
$RectangleSettings = array("R"=>209,"G"=>198,"B"=>27,"Alpha"=>100);
$myPicture->drawRoundedRectangle(480,130,650,160,5,$RectangleSettings);

/* Right Green box */
$RectangleSettings = array("R"=>134,"G"=>209,"B"=>27,"Alpha"=>100);
$myPicture->drawRoundedRectangle(480,170,650,200,5,$RectangleSettings);
```

This will draw some transparency test on the left side of the picture and 4 colored boxes on the right. Shadow support is enabled on a (+1,+1) basis.

# drawAntialiasPixel - Drawing alpha pixels

This function allows you to draw aliased alpha pixels on your pictures. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide. Pixels like all simple drawing functions are supporting anti-alias and shadows.

drawAntialiasPixel is a core function of pChart. A lot of internal drawing methods are based on it as the whole antialiasing processing is done trough it. Antialias can only be done on a pixel unit. To learn more on antialiasing, read this article.

## Calling this function

```
drawAntialiasPixel($X,$Y,$Format="");
```

Where :

- X,Y are the coordinate of the pixel.

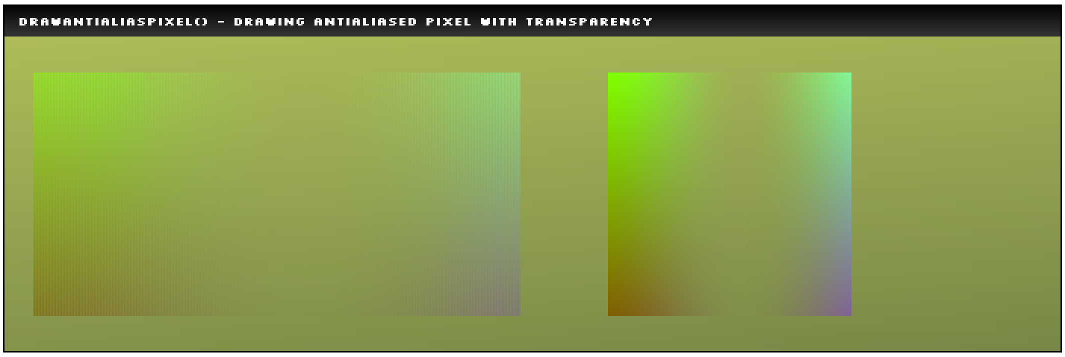
- Format is an array containing the drawing parameters of the pixel.

### Customisation array - Tune up your pixel!

It is possible to customize the pixel rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the pixel will be drawn black with no transparency.

- The pixel color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.

### Sample script


DRAWANTIALIASPIXEL() - DRAWING ANTIALIASED PIXEL WITH TRANSPARENCY

Code sample

```
for($X=0;$X<=160;$X++)
 {
  for($Y=0;$Y<=160;$Y++)
   {
    /* Play with the color / transparency factor */
    $PixelSettings = array("R"=>128,"G"=>255-$Y,"B"=>$X,"Alpha"=>cos(deg2rad($X*2))*50+50);

    /* Draw with antialiasing (adding +.4 to the pixel X position) */
    $myPicture->drawAntialiasPixel($X*2+20.4,$Y+45,$PixelSettings);

    /* Draw continuous pixels */
    $myPicture->drawAntialiasPixel($X+400,$Y+45,$PixelSettings);
   }
 }
```

This will draw two areas with a computed gradient of color/alpha.

## drawArrowLabel - Drawing labels with a pointing arrows

This function has been introduced in the 2.0 trunk of the pChart library. It allows you to draw a label with a pointing arrow. Some possible customisation like the size, angle and color can be made. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

Depending of the specified angle, the label will be positionned either on the right side of the arrow (1°-179°) or on the left side of the arrow (180°-361°).

### Calling this function

```
drawArrowLabel($X,$Y,$Text,$Format="");
```

Where :

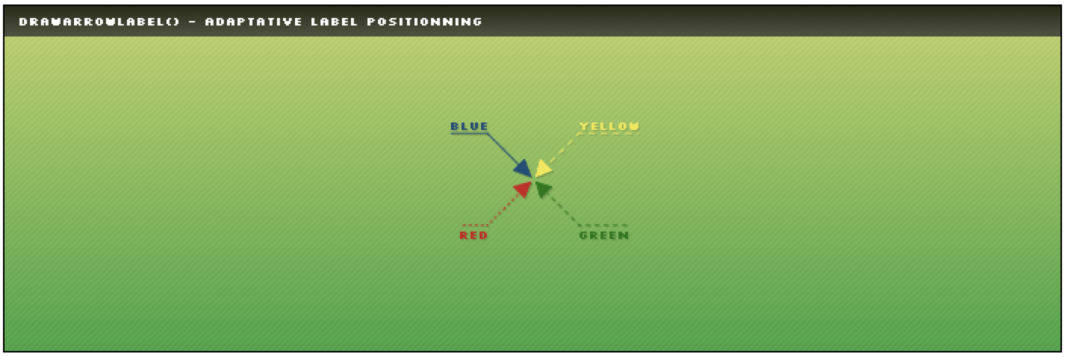
- X,Y are the coordinate of the arrow head.
- Text the caption of the label.
- Format is an array containing the drawing parameters of the arrow.

## Customisation array - Tune up your label and it‘s arrow!

It is possible to customize the arrow rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the caption (and the arrow) will be drawn black with a size of 10, a ratio of 1/2, and angle of 315°.

- The fill color can be set with FillR, FillG, FillB.
- The border color can be set with BorderR, BorderG, BorderB.
- The alpha transparency factor can be set with Alpha.
- The size of the arrow can be set with Size.
- The ratio (internal angle of the head) can be set with Ratio.
- The angle can be set with Angle.
- The font file name can be set with FontName.
- The font size can be set with FontSize.
- The length of the arrow line can be set with Length.
- You can specify if the text will be placed on top or bellow the arrow with Position. Default is POSITION_TOP but you can also use POSITION_BOTTOM.
- You can disable the antialiasing of the line located under the label caption by settin RoundPos to TRUE
- You can draw dashed lines using the Ticks parameter.

## Sample script



Code sample

```
/* Enable shadow support */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Draw blue label */
$ArrowSettings = array("FillR"=>37,"FillG"=>78,"FillB"=>117,"Length"=>40,"Angle"=>45);
$myPicture->drawArrowLabel(348,113,"Blue",$ArrowSettings);

/* Draw red label */
$ArrowSettings = array("FillR"=>188,"FillG"=>49,"FillB"=>42,"Length"=>40,"Angle"=>135,"Position"=>POSITION_BOTTOM, "Ticks"=>2);
$myPicture->drawArrowLabel(348,117,"Red",$ArrowSettings);

/* Draw green label */
$ArrowSettings = array("FillR"=>51,"FillG"=>119,"FillB"=>35,"Length"=>40,"Angle"=>225,"Position"=>POSITION_BOTTOM, "Ticks"=>3);
$myPicture->drawArrowLabel(352,117,"Green",$ArrowSettings);

/* Draw yellow label */
$ArrowSettings = array("FillR"=>239,"FillG"=>231,"FillB"=>97,"Length"=>40,"Angle"=>315,"Ticks"=>4);
$myPicture->drawArrowLabel(352,113,"Yellow",$ArrowSettings);
```

This will draw 4 labels centered on (350,115). As the shadow support is enabled on line 2, the arrow will be surrounded by a small (+2,+2) shadow.

# drawLine - Drawing lines

This function allows you to draw aliased lines on your pictures. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide. Lines like all simple drawing functions are supporting anti-alias and shadows.

This function is also used internally by a lot of drawing functions.

## Calling this function

```
drawLine($X1,$Y1,$X2,$Y2,$Format="");
```

Where :

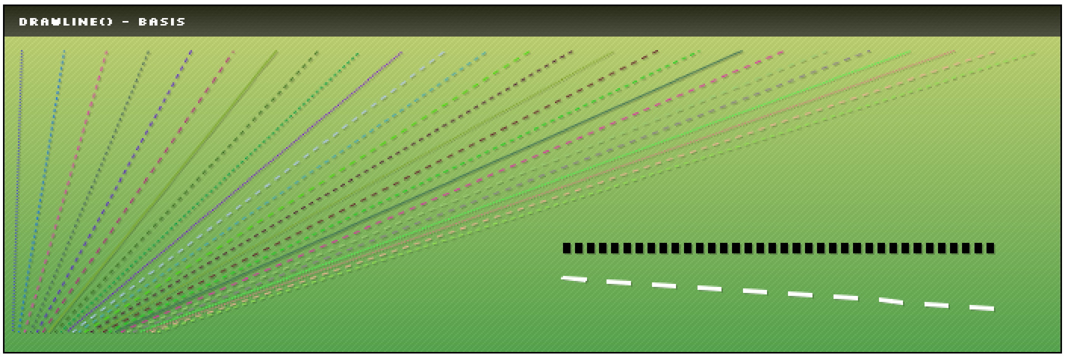
- X1,Y1 are the start coordinate of the line.
- X2,Y2 are the end coordinate of the line.
- Format is an array containing the drawing parameters of the arrow.

## Customisation array - Tune up your line!

It is possible to customize the line rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the line will be drawn black with no transparency.

- The line color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.
- You can draw dashed lines using the Ticks parameter.
- You can specify the weight of the line with the Weight parameter.

## Sample script



Code sample

```
/* Enable shadow support */
/* pChart library inclusions */
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(700,230);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);
```

```
$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>80));

/* Draw the picture border */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawLine() - Basis",array("R"=>255,"G"=>255,"B"=>255));

/* Turn on shadow computing */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* Draw some lines */
for($i=1;$i<=100;$i=$i+4)
 $myPicture->drawLine($i+5,215,$i*7+5,30,array("R"=>rand(0,255),"G"=>rand(0,255),"B"=>rand(0,255),"Ticks"=>rand(0,4)));

/* Draw an horizontal dashed line with extra weight */
$myPicture->drawLine(370,160,650,160,array("R"=>0,"G"=>0,"B"=>0,"Ticks"=>4,"Weight"=>3));

/* Another example of extra weight */
$myPicture->drawLine(370,180,650,200,array("R"=>255,"G"=>255,"B"=>255,"Ticks"=>15,"Weight"=>1));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawLine.png");
```

This will draw 25 lines with random colors. As the shadow support is enabled on line 2, the line will be surrounded by a small (+2,+2) shadow.

# drawThresholdArea - Draw an area between two threshold

This function will draw an area between two threshold. Using areas can help you to highlight specific values. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide.

Information

By default, the area will be drawn using the 1st Y Axis (0) as reference. If you chart contains multiple Y Axis, please use the AxisID parameter of the format array.

## Calling this function

```
drawThresholdArea($Value1,$Value2,$Format="");
```

Where :

- Value1 is the value of the 1st threshold.
- Value2 is the value of the 1st threshold.
- Format is an array containing the drawing parameters of the arrow.

This function returns an array containing the Y position of the up & down boundaries of the area.

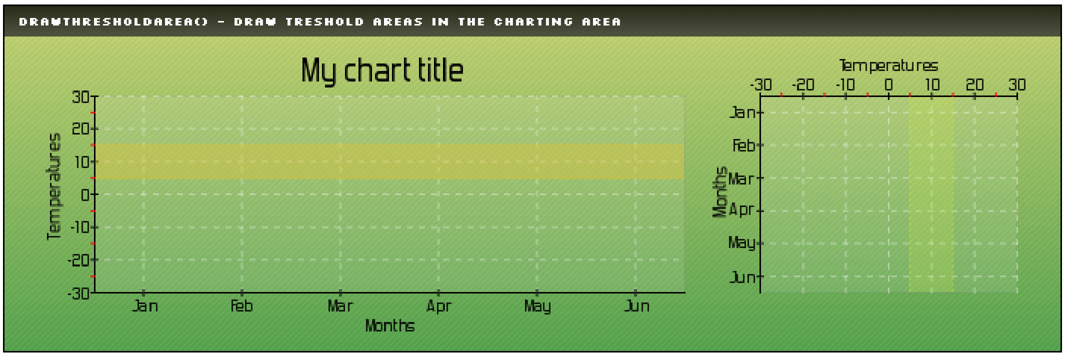
## Customisation array - Tune up your area!

It is possible to customize the way your threshold will be rendered by playing with this array. Providing a detailled configuration is not mandatory, by default the line will be drawn dashed in red.

- You can specify which axis you‘ll use to compute the value position in the chart area AxisID
- You can specify the filled area color using R,G,B.
- You can specify the filled area alpha factor using Alpha.

- You can draw the area bordes setting Border to TRUE or FALSE.
- You can specify the border color using BorderR,BorderG,BorderB.
- You can specify the border area alpha factor using BorderAlpha.
- You can specify the border tick width with BorderTicks.

### Sample script



Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData = new pData();
$MyData->addPoints(array(24,-25,26,25,25),"Temperature");
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(60,60,660,190);
$myPicture->drawFilledRectangle(60,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));

/* Draw the scale */
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));

/* Draw two thresholds */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));
$myPicture->drawThresholdArea(5,15,array("R"=>226,"G"=>194,"B"=>54,"Alpha"=>40));
```

This example will draw an area between Y=5 and Y=15.

## drawPolygon - Draw polygons

This function allows you to draw aliased polygons on your pictures. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide. Lines like all simple drawing functions are supporting anti-alias and shadows.

This function is also used internally by a lot of drawing functions (arrows, ...).

### Calling this function

```
drawPolygon($Points,$Format="");
```

Where :

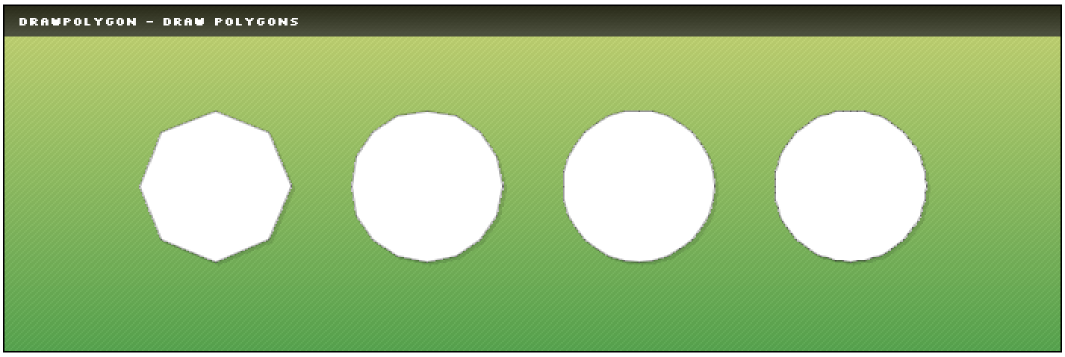
- Points is an array of points.
- Format is an array containing the drawing parameters of the polygon.

### Customisation array - Tune up your polygon!

It is possible to customize the line rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the polygon will be drawn black with no transparency.

- The background color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.
- You can specify the border color BorderR,BorderG,BorderB parameter.
- The alpha transparency factor of the border can be set with BorderAlpha.
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.

### Sample script



Code sample

```
/* Enable shadow support */
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

$White = array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-100);

$Step  = 8;
for($i=1;$i<=4;$i++)
 {
  $Points = "";
  for($j=0;$j<=360;$j=$j+(360/$Step))
   {
    $Points[] = cos(deg2rad($j))*50+($i*140);
    $Points[] = sin(deg2rad($j))*50+120;
   }
  $myPicture->drawPolygon($Points,$White);
  $Step = $Step * 2;
 }
```

This will draw 4 polygons. As the shadow support is enabled on line 2, the polygons will be surrounded by a small (+2,+2) shadow.

## drawRoundedFilledRectangle - self explicit no?

This function allows you to draw rounded filled rectangle : basically, filled rectangles with rounded corners. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide. This functions are supporting anti-alias and shadows.

### Calling this function

```
drawRoundedFilledRectangle($X1,$Y1,$X2,$Y2,$Radius,$Format="");
```

Where :

- X1,Y1 are the start coordinate of the rectangle.
- X2,Y2 are the end coordinate of the rectangle.
- Radius is the radius of the rounded corners.

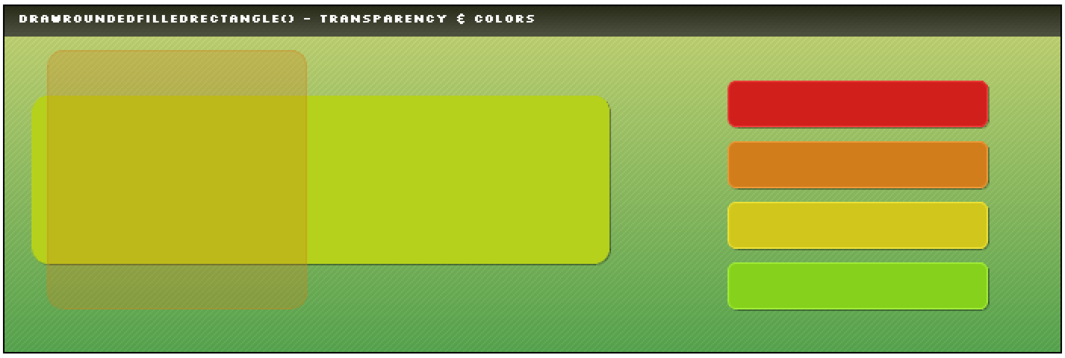
- Format is an array containing the drawing parameters.

### Customisation array - Tune up your area!

It is possible to customize the rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the rectangle will be drawn black with no transparency and no border.

- The fill color can be set with R, G, B.
- The line color can be set with BorderR, BorderG, BorderB.
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.
- The alpha transparency factor can be set with Alpha.

### Sample script



Code sample

```
/* Enable shadow support */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* Left green box */
$RectangleSettings = array("R"=>181,"G"=>209,"B"=>27,"Alpha"=>100);
$myPicture->drawRoundedFilledRectangle(20,60,400,170,10,$RectangleSettings);

/* Left orange surrounding box with an alpha factor of 30% */
$RectangleSettings = array("R"=>209,"G"=>134,"B"=>27,"Alpha"=>30);
$myPicture->drawRoundedFilledRectangle(30,30,200,200,10,$RectangleSettings);

/* Right Red box */
$RectangleSettings = array("R"=>209,"G"=>31,"B"=>27,"Alpha"=>100,"Surrounding"=>30);
$myPicture->drawRoundedFilledRectangle(480,50,650,80,5,$RectangleSettings);

/* Right Orange box */
$RectangleSettings = array("R"=>209,"G"=>125,"B"=>27,"Alpha"=>100,"Surrounding"=>30);
$myPicture->drawRoundedFilledRectangle(480,90,650,120,5,$RectangleSettings);

 /* Right Yellow box */
$RectangleSettings = array("R"=>209,"G"=>198,"B"=>27,"Alpha"=>100,"Surrounding"=>30);
$myPicture->drawRoundedFilledRectangle(480,130,650,160,5,$RectangleSettings);

/* Right Green box */
$RectangleSettings = array("R"=>134,"G"=>209,"B"=>27,"Alpha"=>100,"Surrounding"=>30);
$myPicture->drawRoundedFilledRectangle(480,170,650,200,5,$RectangleSettings);
```

This will draw some transparency test on the left side of the picture and 4 colored boxes on the right. The right boxes will be surrounded by a +30 colored border (R+30,G+30,B+30). Shadow support is enabled on a (+1,+1) basis.

## drawLegend - Write the legend of the active data series

This function will write the legend of your data series based on the description tag specified using the setSerieDescription function. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide.

Information

If you want to specify multi-line labels then you can use the "backslash n" line break character in the serie description tag.

Information

This function preserves your serie special settings (ticks, weight) only in the LEGEND_FAMILY_LINE mode.

## Calling this function

```
drawLegend($X,$Y,$Format="");
```

Where :

- X and Y are the coordinates where will be drawn the legend.
- Format is an array containing the drawing parameters of the arrow.

## Customisation array - Tune up your legend!

It is possible to customize the way your legend will be rendered by playing with this array. Providing a detailled configuration is not mandatory, by default the legend will be drawn in a soft grey box with curvy corners.

- You can specify the font file that will be used with FontName.
- You can specify the font size with FontSize.
- You can specify the font color with FontR, FontG, FontB.
- You can specify the width of the colored boxes with BoxWidth.
- You can specify the height of the colored boxes with BoxHeight.
- You can specify the inner margins with Margin.
- You can specify the background color with R, G, B.
- You can specify the background alpha with Alpha.
- You can specify the border color using BorderR, BorderG, BorderB.
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.

You can choose the style that will be applied to you legend box using the **Style** parameter :

- LEGEND_NOBORDER no borders will be drawn around the legend.
- LEGEND_BOX a rectangle will be drawn around the legend.
- LEGEND_ROUND a rounded rectangle will be drawn around the legend.

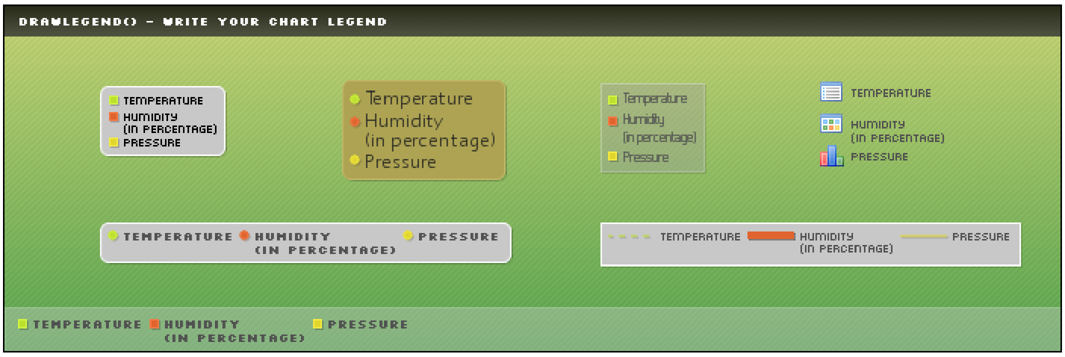
You can also define the way the legend will be written using the **Mode** parameter :

- LEGEND_VERTICAL that will stack vertically the series.
- LEGEND_HORIZONTAL that will stack horizontally the series.

Finally you can choose the way your data serie will be drawn using the **Family** parameter :

- LEGEND_FAMILY_BOX to use filled rectangles.
- LEGEND_FAMILY_CIRCLE to use filled circles.
- LEGEND_FAMILY_LINE to use lines. (preserving the Width & Ticks options)

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(24,25,26,25,25),"My Serie 1");
$MyData->addPoints(array(80,85,84,81,82),"My Serie 2");
$MyData->addPoints(array(17,16,18,18,15),"My Serie 3");
$MyData->setSerieTicks("My Serie 1",4);
$MyData->setSerieWeight("My Serie 2",2);
$MyData->setSerieDescription("My Serie 1","Temperature");
$MyData->setSerieDescription("My Serie 2","Humidity
(in percentage)");
$MyData->setSerieDescription("My Serie 3","Pressure");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>80));

/* Draw the picture border */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawLegend() - Write your chart legend",array("R"=>255,"G"=>255,"B"=>255));

/* Write a legend box */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawLegend(70,60);
```

```
/* Write a legend box */
$myPicture->setFontProperties(array("FontName"=>"../fonts/MankSans.ttf","FontSize"=>10,"R"=>30,"G"=>30,"B"=>30));
$myPicture->drawLegend(230,60,array("BoxSize"=>4,"R"=>173,"G"=>163,"B"=>83,"Surrounding"=>20,"Family"=>LEGEND_FAMILY_CIRCLE));

/* Write a legend box */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>9,"R"=>80,"G"=>80,"B"=>80));
$myPicture->drawLegend(400,60,array("Style"=>LEGEND_BOX,"BoxSize"=>4,"R"=>200,"G"=>200,"B"=>200,"Surrounding"=>20,"Alpha"=>30));

/* Write a legend box */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawLegend(70,150,array("Mode"=>LEGEND_HORIZONTAL, "Family"=>LEGEND_FAMILY_CIRCLE));

/* Write a legend box */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

$myPicture->drawLegend(400,150,array("Style"=>LEGEND_BOX,"Mode"=>LEGEND_HORIZONTAL,

"BoxWidth"=>30,"Family"=>LEGEND_FAMILY_LINE));


/* Write a legend box */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawFilledRectangle(1,200,698,228,array("Alpha"=>30,"R"=>255,"G"=>255,"B"=>255));
$myPicture->drawLegend(10,208,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Define series icons */
$MyData->setSeriePicture("My Serie 1","resources/application_view_list.png");
$MyData->setSeriePicture("My Serie 2","resources/application_view_tile.png");
$MyData->setSeriePicture("My Serie 3","resources/chart_bar.png");

/* Write a legend box */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawLegend(540,50,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_VERTICAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawLegend.png");
```

## drawFilledRectangle - rectangles

This function allows you to draw rectangle. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide. This functions are supporting anti-alias and shadows.

### Calling this function

```
drawRectangle($X1,$Y1,$X2,$Y2,$Format="");
```

Where :

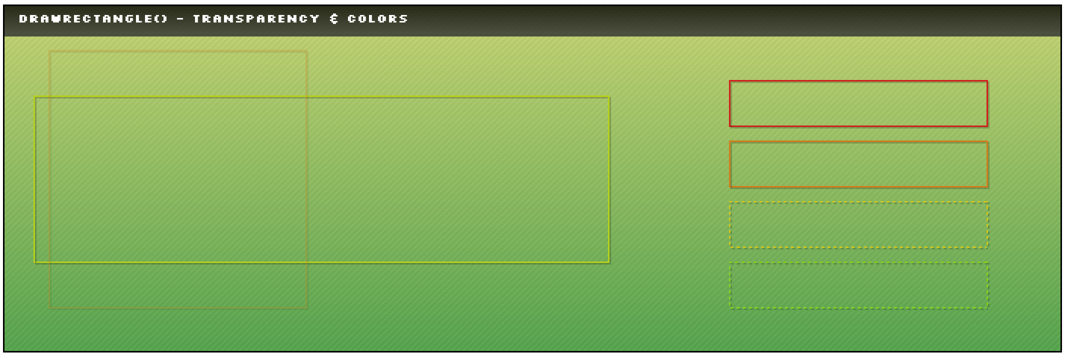
- X1,Y1 are the start coordinate of the rectangle.
- X2,Y2 are the end coordinate of the rectangle.
- Format is an array containing the drawing parameters.

### Customisation array - Tune up your box!

It is possible to customize the box rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the rectangle will be drawn black with no transparency.

- The fill color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.
- You can draw dashed border lines using the Ticks parameter.

### Sample script



Code sample

```
/* Enable shadow support */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* Left green box */
$RectangleSettings = array("R"=>181,"G"=>209,"B"=>27,"Alpha"=>100);
$myPicture->drawRectangle(20,60,400,170,$RectangleSettings);

/* Left orange surrounding box with an alpha factor of 30% */
$RectangleSettings = array("R"=>209,"G"=>134,"B"=>27,"Alpha"=>30);
$myPicture->drawRectangle(30,30,200,200,$RectangleSettings);

/* Right Red box */
$RectangleSettings = array("R"=>209,"G"=>31,"B"=>27,"Alpha"=>100);
$myPicture->drawRectangle(480,50,650,80,$RectangleSettings);

/* Right Orange box */
$RectangleSettings = array("R"=>209,"G"=>125,"B"=>27,"Alpha"=>100);
$myPicture->drawRectangle(480,90,650,120,$RectangleSettings);

 /* Right Yellow box */
$RectangleSettings = array("R"=>209,"G"=>198,"B"=>27,"Alpha"=>100,"Ticks"=>2);
$myPicture->drawRectangle(480,130,650,160,$RectangleSettings);

/* Right Green box */
$RectangleSettings = array("R"=>134,"G"=>209,"B"=>27,"Alpha"=>100,"Ticks"=>2);
$myPicture->drawRectangle(480,170,650,200,$RectangleSettings);
```

This will draw some transparency test on the left side of the picture and 4 colored boxes on the right. Shadow support is enabled on a (+1,+1) basis.

## drawFromJPG - Add pictures to your charts

This function allows you to add JPG pictures to your charts. If you turn on shadow support before calling this function, the merged picture will be drawn with the specified shadow parameters.

### Calling this function

```
drawFromJPG($X,$Y,$FileName);
```

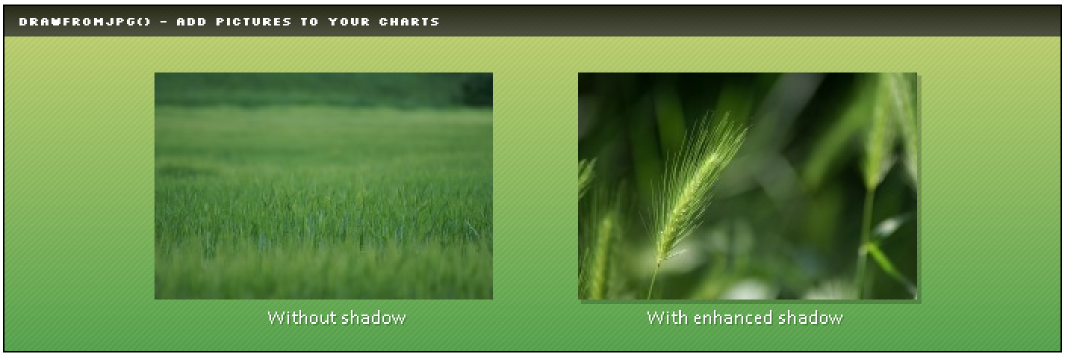
Where :

- X,Y are the coordinate where will be drawn the top left corner of the picture.
- FileName is the path to the picture file.

Information

It is recommended to use PNG files when you want to include pictures into your charts. This is the only real portable format that can handle alpha channels, thus the pictures will look crystal clear.

### Sample script



Code sample

```
/* Turn off shadow */
$myPicture->setShadow(FALSE);
$myPicture->drawFromJPG(100,45,"resources/landscape1.jpg");

/* Enable shadow */
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));
$myPicture->drawFromJPG(380,45,"resources/landscape2.jpg");
```

This will draw two pictures, the first one without shadow and the second one with the internal pChart shadow algorithm.

## writeBounds - Write the min/max values over your chart

This function allows you to write the series boundaries (max, min, both) over your charts. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

Information

This function must be called after a charting function because some of them introduce a X offset on the series

### Calling this function

```
writeBoundaries($Type=BOUND_BOTH,$Format=NULL);
```

Where :

- Type can be either BOUND_MIN, BOUND_MAX or BOUND_BOTH depending of what you want to display.
- Format is an array containing the drawing parameters of the arrow.

### Customisation array - Tune up your rendering!

It is possible to customize the rendering by playing with this array. Providing a detailled configuration is not mandatory, by default both min & max boundaries will be written with different colors.

- The text displayed for the max. value can be set with MaxLabelTxt.
- The text displayed for the min. value can be set with MinLabelTxt.
- The series you want to exclude can be listed in an array named ExcludedSeries.
- The offset between the data point and the label can be adjusted with DisplayOffset.
- You can decide to draw a box under the caption setting DrawBox. to TRUE.

If you choose to draw a box under the caption, all the drawText() extended styles can be used (DrawBoxBorder,BorderOffset,BoxRounded, ...) to have the complete list of parameters, take a look at the drawText() page.

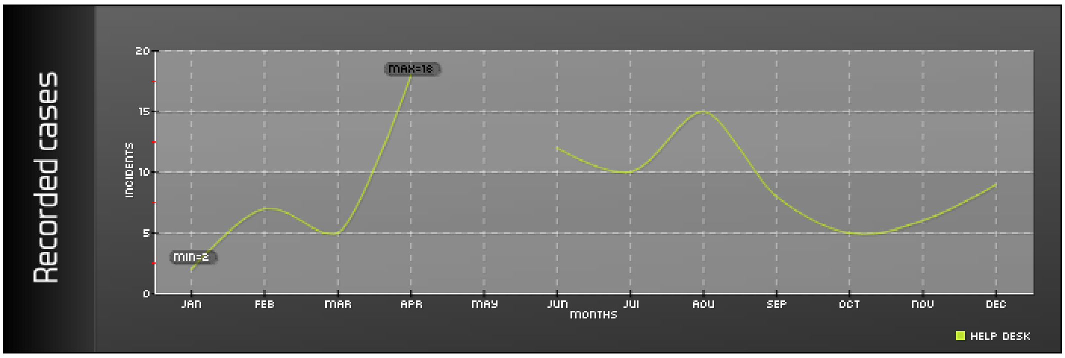
The label coloring method can be defined setting **DisplayColor** to :

- DISPLAY_MANUAL if you want to define specific colors, then use MaxDisplayR, MaxDisplayG, MaxDisplayB and MinDisplayR, MinDisplayG, MinDisplayB to define it.
- DISPLAY_AUTO if you want automatic coloring.

You can specify if the labels will be written on top or under the data point setting **MinLabelPos** and **MaxLabelPos** to :

- BOUND_LABEL_POS_TOP to display the label on top of the point.
- BOUND_LABEL_POS_BOTTOM to display the label under the point.
- BOUND_LABEL_POS_AUTO for automatic positioning.

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(2,7,5,18,VOID,12,10,15,8,5,6,9),"Help Desk");
$MyData->setAxisName(0,"Incidents");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun","Jui","Aou","Sep","Oct","Nov","Dec"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,array("StartR"=>100,"StartG"=>100,"StartB"=>100,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>100));

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_HORIZONTAL,array("StartR"=>100,"StartG"=>100,"StartB"=>100,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>20));

$myPicture->drawGradientArea(0,0,60,230,DIRECTION_HORIZONTAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>100));
```

```
/* Do some cosmetics */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawLine(60,0,60,230,array("R"=>70,"G"=>70,"B"=>70));
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));
$myPicture->drawText(35,115,"Recorded

cases",array("R"=>255,"G"=>255,"B"=>255,"FontSize"=>20,"Angle"=>90,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));


/* Draw a spline chart */
$myPicture->setGraphArea(100,30,680,190);
$myPicture->drawFilledRectangle(100,30,680,190,array("R"=>255,"G"=>255,"B"=>255,"Alpha"=>20));
$myPicture->setFontProperties(array("R"=>255,"G"=>255,"B"=>255,"FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawScale(array("AxisR"=>255,"AxisG"=>255,"AxisB"=>255,"DrawSubTicks"=>TRUE,"CycleBackground"=>TRUE));
$myPicture->drawSplineChart();

/* Write the data bounds */
$myPicture->writeBounds();
$myPicture->setShadow(FALSE);

/* Write the chart legend */
$myPicture->drawLegend(630,215,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.writeBounds.png");
```

# drawFromGIF - Add pictures to your charts

This function allows you to add GIF pictures to your charts. If you turn on shadow support before calling this function, the merged picture will be drawn with the specified shadow parameters.

## Calling this function

```
drawFromGIF($X,$Y,$FileName);
```

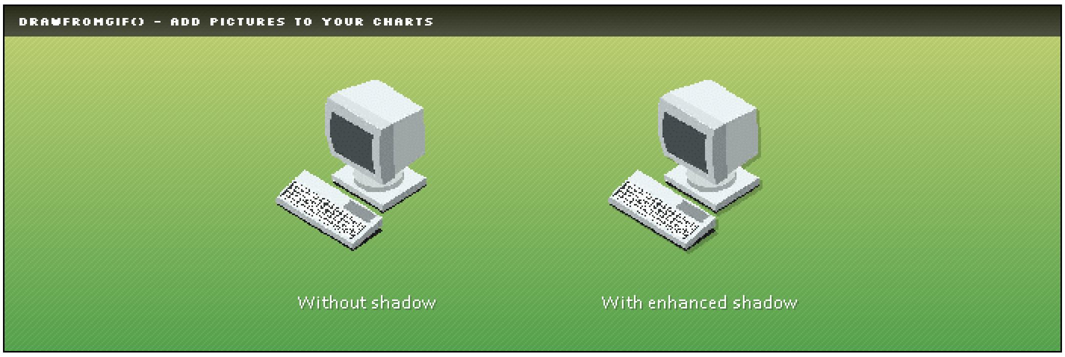
Where :

- X,Y are the coordinate where will be drawn the top left corner of the picture.
- FileName is the path to the picture file.

Information

It is recommended to use PNG files when you want to include pictures into your charts. This is the only real portable format that can handle alpha channels, thus the pictures will look crystal clear.

## Sample script



Code sample

```
/* Turn off shadow */
$myPicture->setShadow(FALSE);
```

```
$myPicture->drawFromPNG(180,50,"resources/computer.gif");

/* Enable shadow */
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));
$myPicture->drawFromPNG(400,50,"resources/computer.gif");
```

This will draw two pictures, the first one with only transparency channel included in the GIF file and the second one with the internal pChart shadow algorithm.

# drawSpline - Chain bezier curves

This function allows you to draw splines composed of many bezier curves on your pictures. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide.

A spline is a curve composed of mandatory way points. All other positions of the curve are computed using the bezier algorithm. As you may have noticed, you can't specify the control points used to compute the bezier segments, those points are computed internally using a trigonometric algorithm. The force applied to the control points can be adjusted with the **Force** parameter of the configuration array.

## Calling this function

```
drawSpline($Coordinates,$Format="")
```

Where :

- Coordinates is an array of arrays containing the way points. (see bellow)
- Format is an array containing the drawing parameters of the pixel.

The coordinates must be specified in an array or points, a point is a two dimension array :

Code sample

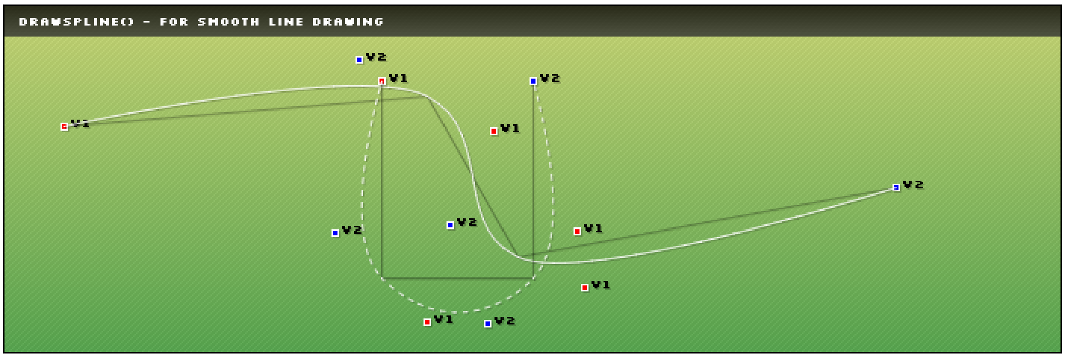
```
/* This is a point, X=10 and Y=30 */
$Point = array(10,30);

/* This is a list of points (10,30) , (20,50) , (30,70) */
$Coordinates = array(array(10,30),array(20,50),array(30,70));
```

## Customisation array - Tune up your spline!

It is possible to customize the pixel rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the spline will be drawn black with no transparency.

- The pixel color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.
- The force of the direction algorithm can be set with Force, default is 30.
- If you want to display the computed control points, turn ShowControl to TRUE.
- You can optionally provide the number of Segments that will compose the spline. It is not recommended to force this value, the best eye-candy / speed ratio is automatically computed based on the spline length.
- You can draw dashed curves using the Ticks parameter.

## Sample script



Code sample

```
/* Enable shadow support on a (+1,+1) basis */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* 1st spline drawn in white with control points visible */
$SplineSettings = array("R"=>255,"G"=>255,"B"=>255,"ShowControl"=>TRUE);
$Coordinates = array(array(40,80),array(280,60),array(340,166),array(590,120));
$myPicture->drawSpline($Coordinates,$SplineSettings);

/* 2nd spline dashed drawn in white with control points visible  */
$SplineSettings = array("R"=>255,"G"=>255,"B"=>255,"ShowControl"=>TRUE,"Ticks"=>4);
$Coordinates = array(array(250,50),array(250,180),array(350,180),array(350,50));
$myPicture->drawSpline($Coordinates,$SplineSettings);
```

This will draw two splines with visible control points. As the shadow support is enabled on line 2, the arrow will be surrounded by a small (+1,+1) shadow.

# drawArrow - Drawing arrows

This function has been introduced in the 2.0 trunk of the pChart library. It allows you to draw an arrow with some possible customisation like the size, angle and color. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

This function is also used internally while drawing the chart axis and the floating labels.

## Calling this function

```
drawArrow($X1,$Y1,$X2,$Y2,$Format="");
```

Where :

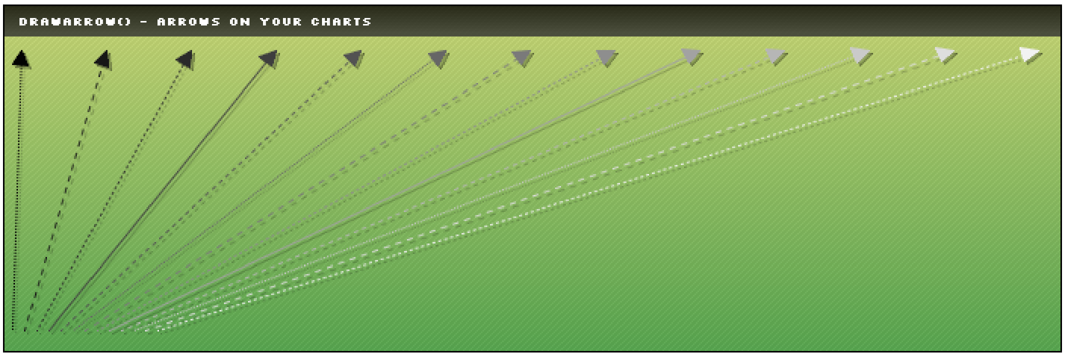
- X1,Y1 are the start coordinate of the arrow line.
- X2,Y2 are the end coordinate of the arrow line. (arrow head)
- Format is an array containing the drawing parameters of the arrow.

## Customisation array - Tune up your arrow!

It is possible to customize the arrow rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the arrow will be drawn black with a size of 10 and a ratio of 1/2, filled with black.

- The fill color can be set with FillR, FillG, FillB.
- The border color can be set with BorderR, BorderG, BorderB.
- The alpha transparency factor can be set with Alpha.
- The size of the arrow can be set with Size.
- The ratio (angle of the head) can be set with Ratio.
- If you want to draw one head on both sides, set TwoHeads to TRUE
- You can draw dashed lines using the Ticks parameter.

### Sample script



Code sample

```
/* Enable shadow support */
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* We'll draw 25 arrow with different angles and transparency */
for($i=1;$i<=100;$i=$i+4)
 {
  /* Settings of the arrow, we'll play with shades of grey */
  $arrowSettings = array("FillR"=>$i*2.5,"FillG"=>$i*2.5,"FillB"=>$i*2.5,"Ticks"=>$i % 5);

  /* Draw the arrow */
  $myPicture->drawArrow($i+5,215,$i*7+5,30,$arrowSettings);
 }
```

This will draw 25 arrows filled with progrssive grey shades and different Ticks length. This sample show that the head angle is internally computed. As the shadow support is enabled on line 2, the arrow will be surrounded by a small (+2,+2) shadow.

# drawDerivative - Drawing the slop factors of your data series

*Require pChart 2.0.13*

This function allows you to draw the slope values of the data series with gradient areas. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

### Calling this function

```
drawDerivative($Format=NULL);
```

Where :

- Format is an array containing the drawing parameters.

### Customisation array - Tune up your rendering!

It is possible to customize the rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the slopes will be rendered with solid patterns.

- The space between the bottom of your scale and the top of the derivative area can be set with Offset.
- The space between two data series can be set with SerieSpacing.
- The height of the slop area can be set with DerivativeHeight.
- You can choose to draw a gradient pattern instead of a solid one setting ShadedSlopeBox to TRUE.
- You can choose to draw a background color under the slope colored area setting DrawBackground to TRUE.
- The color of the background area can be set with BackgroundR, BackgroundG, BackgroundB.
- The alpha transparency factor of the background area can be set with BackgroundAlpha.
- You can choose to draw a border around the slope colored area setting DrawBorder to TRUE.
- The color of the border can be set with BorderR, BorderG, BorderB.

- The alpha transparency factor of the border can be set with BorderAlpha.

Caption area specifics parameters :

- You can choose if the serie caption will be written setting Caption to TRUE.
- You can set the height of the serie caption box with CaptionHeight.
- You can set the width of the serie caption box with CaptionWidth.
- You can choose the space between the caption and the slope area with CaptionMargin.
- You can choose if the caption will be represented as a line instead than a solid box setting CaptionLine to TRUE.
- You can choose if the caption will be written in a box setting CaptionBox to TRUE.
- The box border color can be adjusted with CaptionBorderR, CaptionBorderG, CaptionBorderB.
- The box filling color can be adjusted with CaptionFillR, CaptionFillG, CaptionFillB, CaptionFillAlpha.

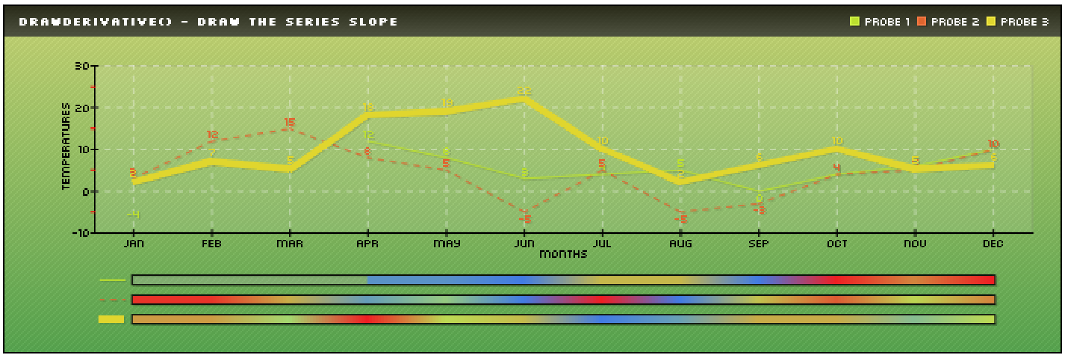
Gradient area specifics parameters for positive slopes :

- The slop 0 factor color can be set with PositiveSlopeStartR, PositiveSlopeStartG, PositiveSlopeStartB.
- The slop max factor color can be set with PositiveSlopeEndR, PositiveSlopeEndG, PositiveSlopeEndB.

Gradient area specifics parameters for negative slopes :

- The slop 0 factor color can be set with NegativeSlopeStartR, NegativeSlopeStartG, NegativeSlopeStartB.
- The slop max factor color can be set with NegativeSlopeEndR, NegativeSlopeEndG, NegativeSlopeEndB.

## Sample script #1

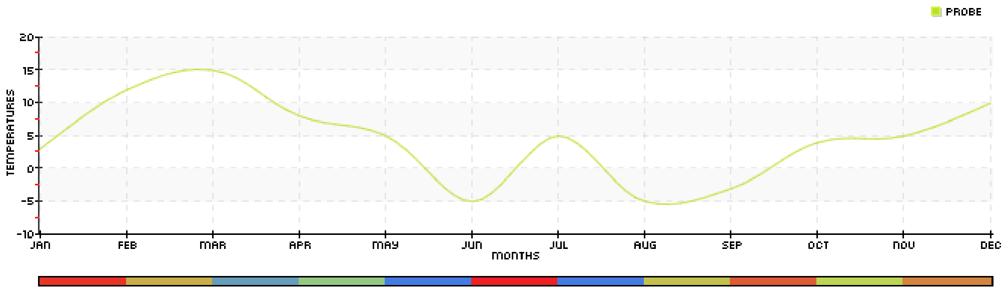


Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(-4,VOID,VOID,12,8,3,4,5,0,4,6,10),"Probe 1");
$MyData->addPoints(array(3,12,15,8,5,-5,5,-5,-3,4,5,10),"Probe 2");
$MyData->addPoints(array(2,7,5,18,19,22,10,2,6,10,5,6),"Probe 3");
$MyData->setSerieTicks("Probe 2",4);
$MyData->setSerieWeight("Probe 3",2);
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);
```

```
/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>80));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawDerivative() - draw the series slope",array("R"=>255,"G"=>255,"B"=>255));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

/* Draw the scale and the 1st chart */
$myPicture->setGraphArea(60,40,680,150);
$myPicture->drawFilledRectangle(60,40,680,150,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawLineChart(array("DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO));

/* Draw the series derivative graph */
$myPicture->drawDerivative(array("ShadedSlopeBox"=>TRUE,"CaptionLine"=>TRUE));

/* Write the chart legend */
$myPicture->setFontProperties(array("R"=>255,"G"=>255,"B"=>255));
$myPicture->drawLegend(560,8,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawDerivative.png");
```

## Sample script #2



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
```

```
/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(3,12,15,8,5,-5,5,-5,-3,4,5,10),"Probe");
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Turn of AAliasing */
$myPicture->Antialias = FALSE;

/* Set the default font */
$myPicture->setFontProperties(array("R"=>0,"G"=>0,"B"=>0,"FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

/* Draw the scale and the 1st chart */
$myPicture->setGraphArea(50,40,680,170);

$scaleSettings = array("GridR"=>200,"GridG"=>200,"GridB"=>200,"DrawSubTicks"=>TRUE,"CycleBackground"=>TRUE,"XMargin"=>1);
$myPicture->drawScale($scaleSettings);

/* Draw the chart */
$myPicture->Antialias = TRUE;
$myPicture->drawSplineChart();
$myPicture->Antialias = FALSE;

/* Draw the series derivative graph */
$myPicture->drawDerivative(array("Caption"=>FALSE));

/* Write the chart legend */
$myPicture->drawLegend(640,20,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawDerivative.simple.png");
```

# drawFilledCircle - Circles

This function allows you to draw filled circles. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide. This functions are supporting anti-alias and shadows.

## Calling this function

```
drawFilledCircle($X1,$Y1,$Radius,$Format="");
```

Where :

- X1,Y1 are the center coordinate of the circle.
- Radius is the radius of the circle.
- Format is an array containing the drawing parameters.

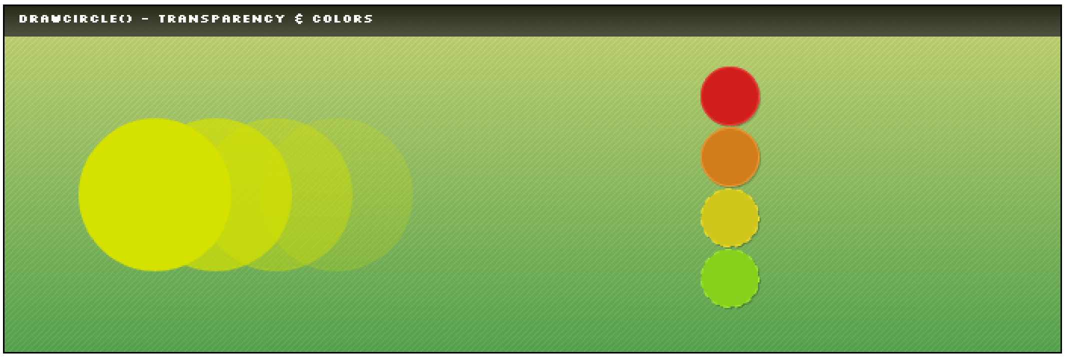
## Customisation array - Tune up your circle!

It is possible to customize the circle rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the circles will be drawn black with no transparency.

- The fill color can be set with R, G, B.
- The border color can be set with BorderR, BorderG, BorderB.
- The border alpha factor can be set with BorderAlpha.
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.

- The alpha transparency factor can be set with Alpha.
- You can draw dashed border lines using the Ticks parameter.

### Sample script



Code sample

```
/* Left circles with different alpha transparency */
$myPicture->drawFilledCircle(100,125,50,array("R"=>213,"G"=>226,"B"=>0,"Alpha"=>100));
$myPicture->drawFilledCircle(140,125,50,array("R"=>213,"G"=>226,"B"=>0,"Alpha"=>70));
$myPicture->drawFilledCircle(180,125,50,array("R"=>213,"G"=>226,"B"=>0,"Alpha"=>40));
$myPicture->drawFilledCircle(220,125,50,array("R"=>213,"G"=>226,"B"=>0,"Alpha"=>20));

/* Active shadow support */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* Right circles */
$CircleSettings = array("R"=>209,"G"=>31,"B"=>27,"Alpha"=>100,"Surrounding"=>30);
$myPicture->drawFilledCircle(480,60,19,$CircleSettings);

$CircleSettings = array("R"=>209,"G"=>125,"B"=>27,"Alpha"=>100,"Surrounding"=>30);
$myPicture->drawFilledCircle(480,100,19,$CircleSettings);

$CircleSettings = array("R"=>209,"G"=>198,"B"=>27,"Alpha"=>100,"Surrounding"=>30,"Ticks"=>4);
$myPicture->drawFilledCircle(480,140,19,$CircleSettings);

$CircleSettings = array("R"=>134,"G"=>209,"B"=>27,"Alpha"=>100,"Surrounding"=>30,"Ticks"=>4);
$myPicture->drawFilledCircle(480,180,19,$CircleSettings);
```

This will draw some transparency test on the left side of the picture and 4 colored boxes on the right. Shadow support is enabled on a (+1,+1) basis.

## drawFromPNG - Add pictures to your charts

This function allows you to add PNG pictures to your charts. If you turn on shadow support before calling this function, the merged picture will be drawn with the specified shadow parameters.

### Calling this function

```
drawFromPNG($X,$Y,$FileName);
```

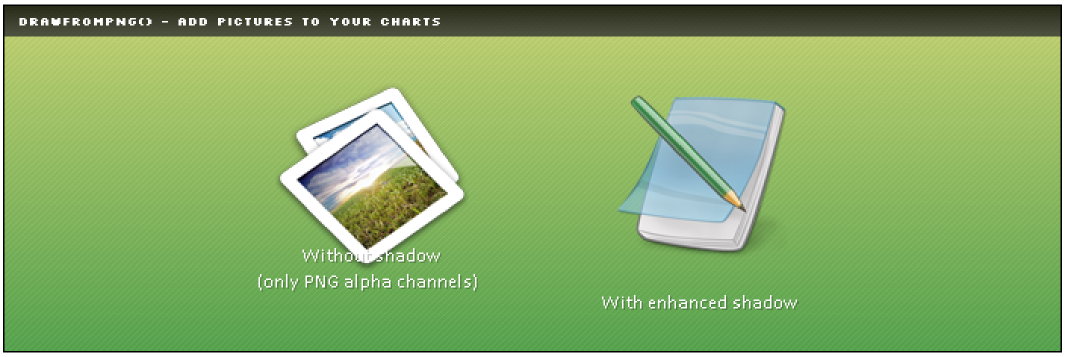
Where :

- X,Y are the coordinate where will be drawn the top left corner of the picture.
- FileName is the path to the picture file.

Information

It is recommended to use PNG files when you want to include pictures into your charts. This is the only real portable format that can handle alpha channels, thus the pictures will look crystal clear.

## Sample script


DRAWFROMPNG() - ADD PICTURES TO YOUR CHARTS
Without shadow
(only PNG alpha channels)
With enhanced shadow

Code sample

```
/* pChart library inclusions */
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(700,230);

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,array("StartR"=>180,"StartG"=>193,"StartB"=>91,"EndR"=>120,"EndG"=>137,"EndB"=>72,"Alpha"=>100));

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_HORIZONTAL,array("StartR"=>180,"StartG"=>193,"StartB"=>91,"EndR"=>120,"EndG"=>137,"EndB"=>72,"Alpha"=>20));

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>100));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawFromPNG() - add pictures to your charts",array("R"=>255,"G"=>255,"B"=>255));

/* Turn off shadow computing */
$myPicture->setShadow(FALSE);

/* Draw a PNG object */
$myPicture->drawFromPNG(180,50,"resources/hologram.png");
```

```
 /* Turn on shadow computing */
 $myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

 /* Draw a PNG object */
 $myPicture->drawFromPNG(400,50,"resources/blocnote.png");

 /* Write the legend */
 $myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));
 $TextSettings = array("R"=>255,"G"=>255,"B"=>255,"FontSize"=>10,"FontName"=>"../fonts/calibri.ttf","Align"=>TEXT_ALIGN_BOTTOMMIDDLE);
 $myPicture->drawText(240,190,"          Without shadow
(only PNG alpha channels)",$TextSettings);
 $myPicture->drawText(460,200,"With enhanced shadow",$TextSettings);

 /* Render the picture (choose the best way) */
 $myPicture->autoOutput("pictures/example.drawFromPNG.png");
```

This will draw two pictures, the first one with only alpha channels included in the PNG file (that may contains a shadow effect!!) and the second one with the internal pChart shadow algorithm.

# drawThreshold - Draw a threshold in the chart area

This function will draw a threshold in the chart area. Using thresholds can help you to highlight specific values. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide.

Information

By default, the threshold will be drawn using the 1st Y Axis (0) as reference. If you chart contains multiple Y Axis, please use the AxisID parameter of the format array.

## Calling this function

```
drawThreshold($Value,$Format="");
```

Where :

- Value is the value of the threshold to draw.
- Format is an array containing the drawing parameters of the arrow.

This function returns an array containing the Y position of the threshold line.

## Customisation array - Tune up your threshold!

It is possible to customize the way your threshold will be rendered by playing with this array. Providing a detailled configuration is not mandatory, by default the line will be drawn dashed in red.

- You can specify which axis you'll use to compute the value position in the chart area with AxisID.
- You can specify the line color using R,G,B.
- You can specify the line alpha factor using Alpha.
- You can specify the line tick width with Ticks.
- You can choose to write a caption over the threshold setting WriteCaption to TRUE.
- By default the value of the threshold will be written, this can be override setingCaption to whatever you want.
- You can specify an offset to the alignement axis with CaptionOffset to TRUE.
- The color of the caption can be set using CaptionR,CaptionG,CaptionB.
- The alpha transparency of the caption can be set using CaptionAlpha.
- You can decide to draw a box under the caption setting DrawBox. to TRUE.

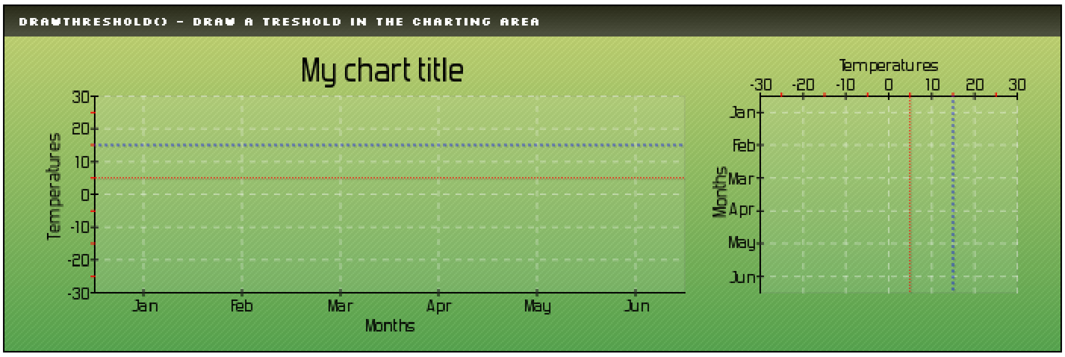
If you choose to draw a box under the caption, all the drawText() extended styles can be used (DrawBoxBorder,BorderOffset,BoxRounded, ...) to have the complete list of parameters, take a look at the drawText() page.

The captions can be aligned using **CaptionAlign** the following ways :

- CAPTION_LEFT_TOP on the left or top side (depending of the scale orientation).

- CAPTION_RIGHT_BOTTOM on the right or bottom side (depending of the scale orientation).

## Sample script

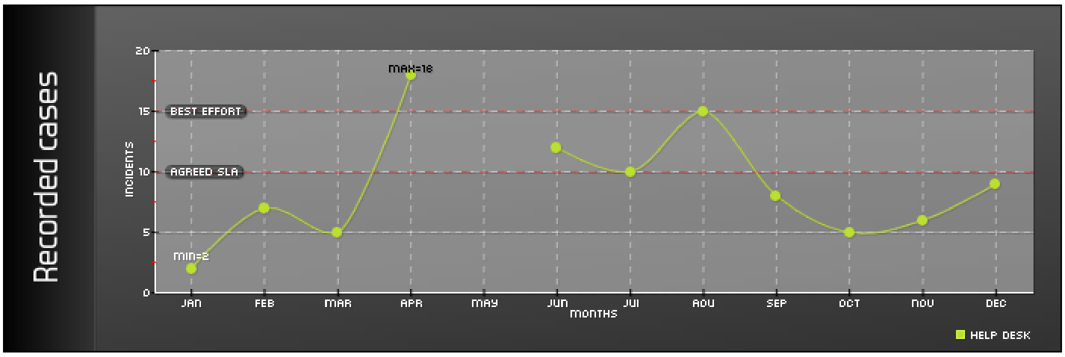


Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData = new pData();
$MyData->addPoints(array(24,-25,26,25,25),"Temperature");
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(60,60,660,190);
$myPicture->drawFilledRectangle(60,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));

/* Draw the scale */
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));

/* Draw two thresholds */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));
$myPicture->drawThreshold(5,array("Alpha"=>70,"Ticks"=>1));
$myPicture->drawThreshold(15,array("Alpha"=>70,"Ticks"=>2,"R"=>0,"G"=>0,"B"=>255));
```

This example will draw two threshold at Y=5 and Y=15.

## Sample script #2



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
```

```
 $MyData = new pData();
 $MyData->addPoints(array(2,7,5,18,VOID,12,10,15,8,5,6,9),"Help Desk");
 $MyData->setAxisName(0,"Incidents");
 $MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun","Jui","Aou","Sep","Oct","Nov","Dec"),"Labels");
 $MyData->setSerieDescription("Labels","Months");
 $MyData->setAbscissa("Labels");

 /* Create the pChart object */
 $myPicture = new pImage(700,230,$MyData);

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,array("StartR"=>100,"StartG"=>100,"StartB"=>100,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>100));

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_HORIZONTAL,array("StartR"=>100,"StartG"=>100,"StartB"=>100,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>20));

$myPicture->drawGradientArea(0,0,60,230,DIRECTION_HORIZONTAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>100));

 /* Do some cosmetics */
 $myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
 $myPicture->drawLine(60,0,60,230,array("R"=>70,"G"=>70,"B"=>70));
 $myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));
 $myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));
 $myPicture->drawText(35,115,"Recorded cases",array("R"=>255,"G"=>255,"B"=>255,"FontSize"=>20,"Angle"=>90,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));

 /* Prepare the chart area */
 $myPicture->setGraphArea(100,30,680,190);
 $myPicture->drawFilledRectangle(100,30,680,190,array("R"=>255,"G"=>255,"B"=>255,"Alpha"=>20));
 $myPicture->setFontProperties(array("R"=>255,"G"=>255,"B"=>255,"FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
 $myPicture->drawScale(array("AxisR"=>255,"AxisG"=>255,"AxisB"=>255,"DrawSubTicks"=>TRUE,"CycleBackground"=>TRUE));

 /* Write two thresholds over the chart */
 $myPicture->drawThreshold(10,array("WriteCaption"=>TRUE,"Caption"=>"Agreed SLA"));
 $myPicture->drawThreshold(15,array("WriteCaption"=>TRUE,"Caption"=>"Best effort"));

 /* Draw the chart */
 $myPicture->drawSplineChart();
 $myPicture->drawPlotChart();

 /* Write the data bounds */
 $myPicture->writeBounds();
```

```
/* Write the chart legend */
$myPicture->drawLegend(630,215,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawThreshold.labels.png");
```

# drawScale - Compute & draw the chart scale

This function will compute and draw the scale of your charts. It handle both X & X-Y charts (which wasn't the case in pChart 1.x). If no series is defined as abcsissa or as X axis, a virtual X axis will be created numbered with data points IDs. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide.

Be careful

This function must be called before any charting functions.

## Calling this function

```
drawScale($Format="");
```

Where :

- Format is an array containing the drawing parameters of the arrow.

## Customisation array - Tune up your scales!

It is possible to customize the way your scale will be rendered by playing with this array. Providing a detailled configuration is not mandatory, by default the axis will be drawn black in the Left / Right mode.

- The scale orientation can be set with Pos, valid values are SCALE_POS_LEFTRIGHT and SCALE_POS_TOPBOTTOM
- You can specify the rotation of the X Axis labels with LabelRotation.
- The minimum height of a scale div can be set using MinDivHeight.
- The scaling factors can be set using Factors.
- The X Axis margin can be set using XMargin. By default the margin on this axis are set to AUTO but you can specify an integer.
- The Y Axis margin can be set using YMargin.
- The space between two scales on the same side can be set using ScaleSpacing.
- You can specify if you want to draw the X grid lines setting DrawXLines to TRUE or FALSE.
- You can specify if you want to draw the Y grid lines setting DrawYLines to ALL, NONE or to an array containing the axis IDs.
- You can draw dashed grid lines setting GridTicks to the width of the ticks.
- You can specify the grid color using GridR,GridG,GridB.
- You can specify the grid alpha factor using GridAlpha.
- You can specify the axis color using AxisR,AxisG,AxisB.
- You can specify the axis alpha factor using AxisAlpha.
- The inner ticks width can be set using InnerTickWidth.
- The outer ticks width can be set using OuterTickWidth.
- You can specify the ticks color using TickR,TickG,TickB.
- You can specify the ticks alpha factor using TickAlpha.
- If you want to draw the subticks of the Y axis set DrawSubTicks to TRUE.
- The inner subticks width can be set using InnerSubTickWidth.
- The outer subticks width can be set using OuterSubTickWidth.
- You can specify the subticks color using SubTickR,SubTickG,SubTickB.
- You can specify the subticks alpha factor using SubTickAlpha.
- You can specify if you want to draw the axis arrows setting DrawArrows to TRUE or FALSE.
- You can set the arrow size usingArrowSize.
- You can choose to display a 2-colors cycling background setting CycleBackground to TRUE.
- First background cycling color can be defined with BackgroundR1,BackgroundG1,BackgroundB1,BackgroundAlpha1.
- 2nd background cycling color can be defined with BackgroundR2,BackgroundG2,BackgroundB2,BackgroundAlpha2.

- You can skip specified number of X labels using LabelSkip.

If you choose to skip some labels using the **LabelSkip** parameter then you can adjust the rendering of the ticks that will have no labels with the following parameters :

- You can change the dash interval of the Y lines setting GridTicks to the width of the ticks.
- You can specify the grid color using SkippedAxisR,SkippedAxisG,SkippedAxisB.
- You can specify the alpha factor of the grid using SkippedAxisAlpha.
- You can specify the ticks color using SkippedTickR,SkippedTickG,SkippedTickB.
- You can specify the alpha factor of the ticks using SkippedTickAlpha.
- The inner ticks width can be set using SkippedInnerTickWidth.
- The outer ticks width can be set using SkippedOuterTickWidth.

You can choose to display all X labels or only the ones that are different from the previously written. This can be usefull when using dates to only display different dates. Use the **LabelingMethod** parameter to set :

- LABELING_ALL will write all X labels.
- LABELING_DIFFERENT will only write labels that are different than the previously written one..

Depending on the chart you'll draw, you can define the way the scale will be computed with the **Mode** parameter :

- SCALE_MODE_FLOATING for min/max automatic scaling.
- SCALE_MODE_START0 for fixed min to 0.
- SCALE_MODE_ADDALL for stacked charts.
- SCALE_MODE_ADDALL_START0 for stacked charts with a fixed min to 0.
- SCALE_MODE_MANUAL to fix manually the min & max values.

If you choose the manual option, you'll have to provide an array with the **ManualScale** parameter to provide the min & max values per axis. This array should have the following structure :

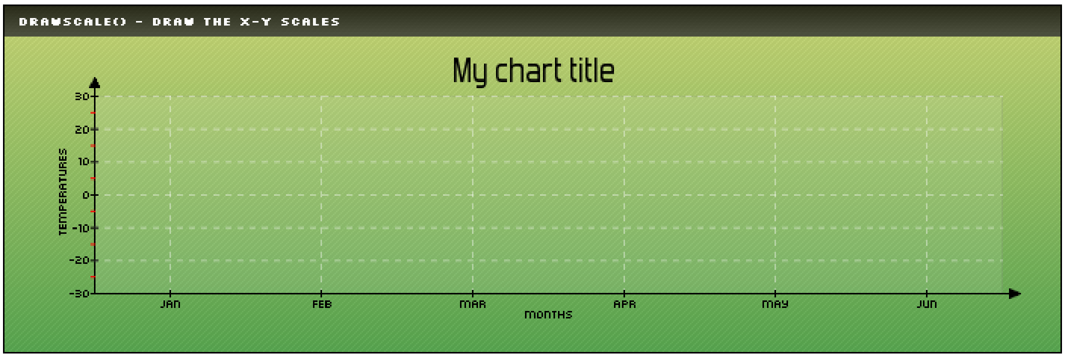
Code sample

```
$AxisBoundaries = array(0=>array("Min"=>0,"Max"=>100),1=>array("Min"=>10,"Max"=>20));
```

Information

This function may be a bit complex to understand, you will see more example within the charting functions section.

## Sample script #1



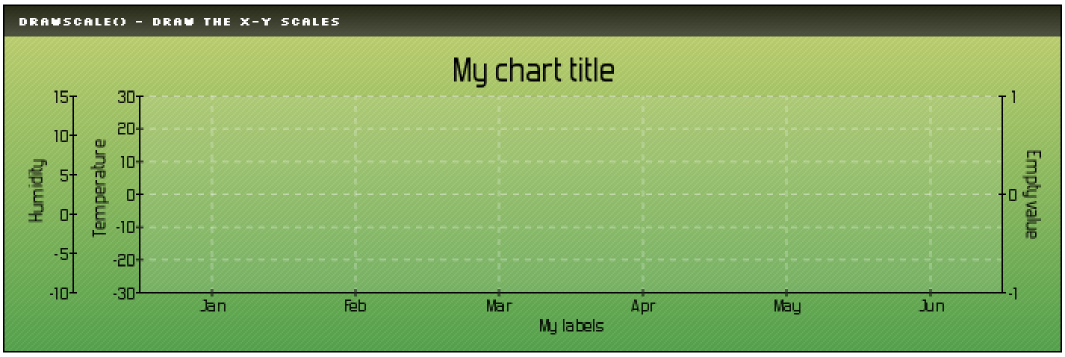
Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData->addPoints(array(24,-25,26,25,25),"Temperature");
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Define the graph area and do some makeup */
```

```
$myPicture->setGraphArea(60,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(60,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE,"DrawArrows"=>TRUE,"ArrowSize"=>6));
```

This is a simple example using two data serie, one numeric to chart and one to use as abscissa labels.

## Sample script #2

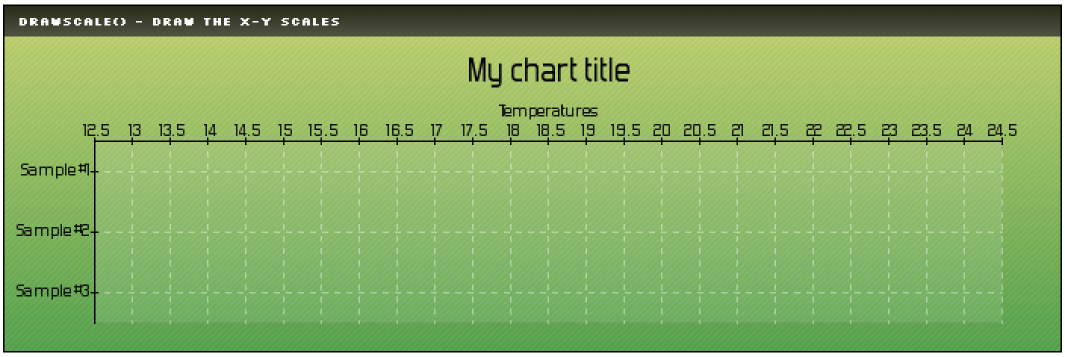


Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData->addPoints(array(24,-25,26,25,25),"Temperature");
$MyData->addPoints(array(1,2,VOID,9,10),"Humidity 1");
$MyData->addPoints(array(1,VOID,7,-9,0),"Humidity 2");
$MyData->addPoints(array(-1,-1,-1,-1,-1),"Humidity 3");
$MyData->addPoints(array(0,0,0,0,0),"Vide");
$MyData->setSerieOnAxis("Temperature",0);
$MyData->setSerieOnAxis("Humidity 1",1);
$MyData->setSerieOnAxis("Humidity 2",1);
$MyData->setSerieOnAxis("Humidity 3",1);
$MyData->setSerieOnAxis("Vide",2);
$MyData->setAxisPosition(2,AXIS_POSITION_RIGHT);
$MyData->setAxisName(0,"Temperature");
$MyData->setAxisName(1,"Humidity");
$MyData->setAxisName(2,"Empty value");

/* Bind a data serie to the X axis */
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","My labels");
$MyData->setAbscissa("Labels");

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(90,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(90,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
$myPicture->drawScale(array("DrawYLines"=>array(0)));
```

This example compute & draw the scale. Only the Axis with an ID of 0 (Temperature) will have horizontal lines because of the **DrawYLines** option specified in the *drawScale* call.

## Sample script #3

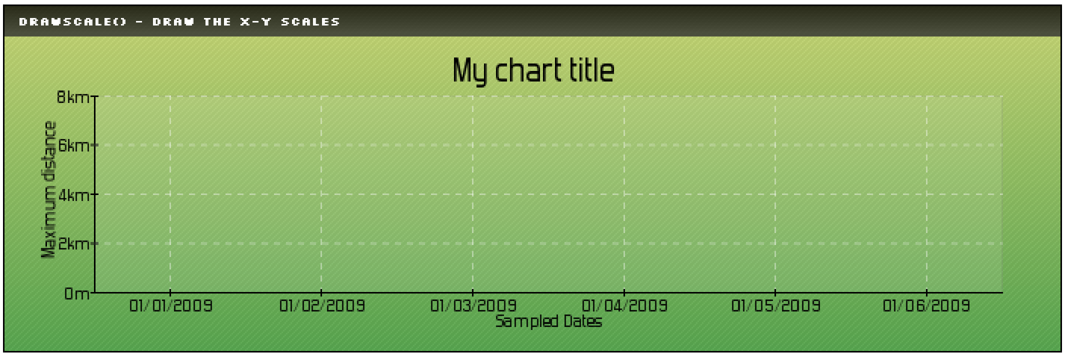


Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData = new pData();
$MyData->addPoints(array(24,13,14),"Temperature");
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Sample #1","Sample #2","Sample #3"),"Labels");
$MyData->setAbscissa("Labels");

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(60,90,660,210);
$myPicture->drawText(360,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(60,90,660,210,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
$myPicture->drawScale(array("AutoAxisLabels"=>FALSE,"Pos"=>SCALE_POS_TOPBOTTOM));
```

This example shows how you can invert the X and Y axis.

## Sample script #4



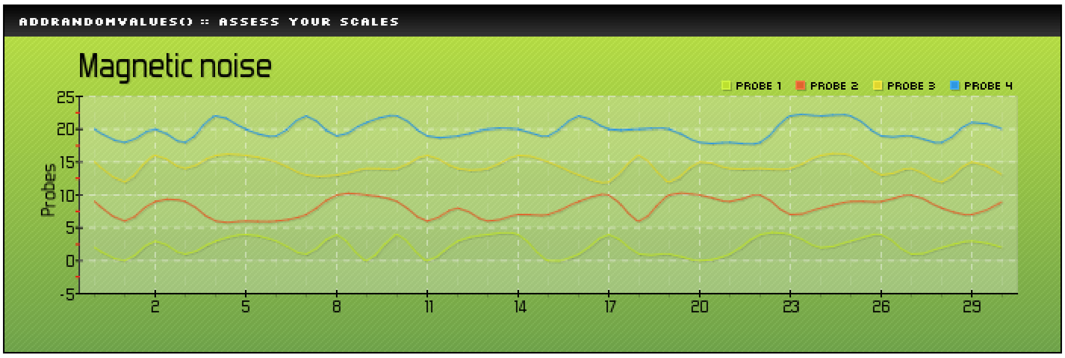
Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData = new pData();
$MyData->addPoints(array(1700,2500,7800,4500,3150),"Distance");
$MyData->setAxisName(0,"Maximum distance");
$MyData->setAxisUnit(0,"m");
$MyData->setAxisDisplay(0,AXIS_FORMAT_METRIC);

/* Create the X serie */
$MyData->addPoints(array(1230768000,1233446400,1235865600,1238544000,1241136000,1243814400),"Timestamp");
$MyData->setSerieDescription("Timestamp","Sampled Dates");
$MyData->setAbscissa("Timestamp");
$MyData->setXAxisDisplay(AXIS_FORMAT_DATE);
```

```
/* Define the graph area and do some makeup */
$myPicture->setGraphArea(60,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(60,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
$myPicture->drawScale();
```

This example shows how the drawScale function handle the axis values formatting.

## Sample script #5



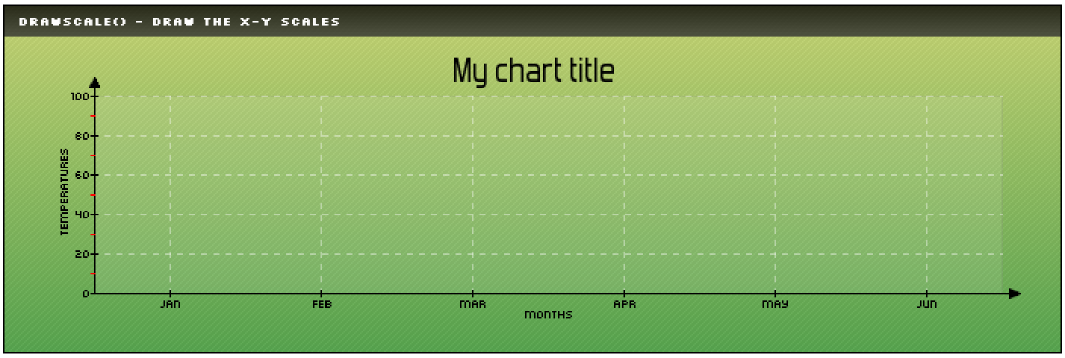
Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create the pData object with some random values*/
$MyData = new pData();
$MyData->addRandomValues("Probe 1",array("Values"=>30,"Min"=>0,"Max"=>4));
$MyData->addRandomValues("Probe 2",array("Values"=>30,"Min"=>6,"Max"=>10));
$MyData->addRandomValues("Probe 3",array("Values"=>30,"Min"=>12,"Max"=>16));
$MyData->addRandomValues("Probe 4",array("Values"=>30,"Min"=>18,"Max"=>22));
$MyData->setAxisName(0,"Probes");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Create a solid background */
$Settings = array("R"=>179, "G"=>217, "B"=>91, "Dash"=>1, "DashR"=>199, "DashG"=>237, "DashB"=>111);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Do a gradient overlay */
$Settings = array("StartR"=>194, "StartG"=>231, "StartB"=>44, "EndR"=>43, "EndG"=>107, "EndB"=>58, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>100));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));
```

```
/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"addRandomValues() :: assess your scales",array("R"=>255,"G"=>255,"B"=>255));

/* Draw the scale */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));
$myPicture->setGraphArea(50,60,670,190);
$myPicture->drawFilledRectangle(50,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("CycleBackground"=>TRUE,"LabelSkip"=>3,"DrawSubTicks"=>TRUE));

/* Graph title */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawText(50,52,"Magnetic noise",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMLEFT));

/* Draw the data series */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawSplineChart();
$myPicture->setShadow(FALSE);

/* Write the legend */
$myPicture->drawLegend(475,50,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.addRandomValues.png");
```

This example shows how to play with the skipped X axis labels.

## Sample script #6



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(24,-25,26,25,25),"Temperature");
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
```

```
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>80));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawScale() - draw the X-Y scales",array("R"=>255,"G"=>255,"B"=>255));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));

/* Write the chart title */
$myPicture->setGraphArea(60,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(60,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

/* Draw the scale */
$AxisBoundaries = array(0=>array("Min"=>0,"Max"=>100));
$ScaleSettings  =
array("Mode"=>SCALE_MODE_MANUAL,"ManualScale"=>$AxisBoundaries,"DrawSubTicks"=>TRUE,"DrawArrows"=>TRUE,"ArrowSize"=>6);
$myPicture->drawScale($ScaleSettings);

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawScale.png");
```

This example will create a manual scale going from 0 to 100.

# drawBezier - Drawing bezier curve

This function allows you to draw aliased bezier curves on your pictures. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide.

A bezier curve (within pChart) is defined by a starting point, and ending point and two control points. It can be pretty tricky to determine how to calculate the position of the control points, the Spline function is using simple trigonometric concepts to calculate it automatically for best rendering.This function is used internally by the drawSpline function to draw complex structures.

Information

In the case where (X1,Y1) = (Xv1,Yv1) and (X2,Y2) = (Xv2,Yv2), this function will draw a line.

## Calling this function

```
drawBezier($X1,$Y1,$X2,$Y2,$Xv1,$Yv1,$Xv2,$Yv2,$Format="");
```

Where :

- X1,Y1 are the coordinate of the starting point.
- X2,Y2 are the coordinate of the ending point.

- Xv1,Yv1 are the coordinate of the 1[SUP]st[/SUP] force vector.
- Xv2,Yv2 are the coordinate of the 2[SUP]nd[/SUP] force vector.
- Format is an array containing the drawing parameters of the pixel.

### Customisation array - Tune up your curve!

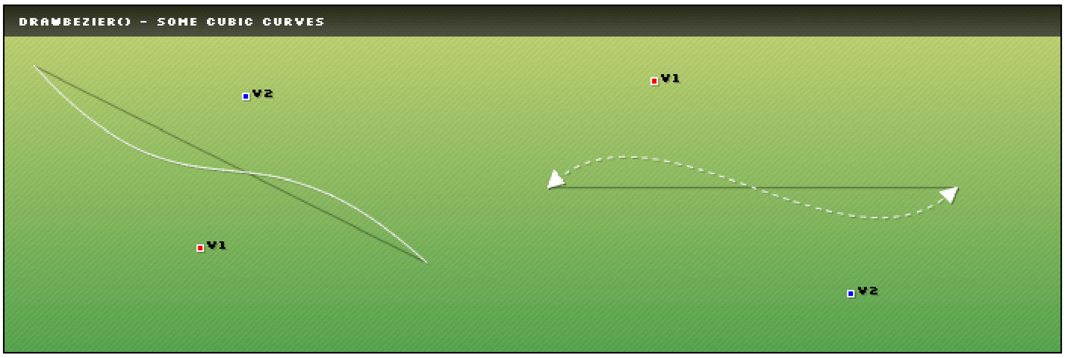
It is possible to customize the pixel rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the curve will be drawn black with no transparency.

- The pixel color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.
- If you want to display the control points, turn ShowControl to TRUE.
- You can optionally provide the number of Segments that will compose the spline. It is recommended to not force this value, the best eye-candy / speed ratio is automatically computed based on the spline length.
- You can draw dashed curves using the Ticks parameter.

..and the arrow settings :

- DrawArrow define if an arrow will be drawn at the end of the curve if set to TRUE.
- ArrowSize is the size of the arrow. Set to 10 by default.
- ArrowRatio is the ratio of the arrow head. Set to 0.5 by default.
- ArrowTwoHeads define if an arrow will also be drawn at the beginning of the curve if set to TRUE.

### Sample script



Code sample

```
/* Enable shadow support on a (+1,+1) basis */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* We want to draw a white bezier curves with visible control points */
$BezierSettings = array("R"=>255,"G"=>255,"B"=>255,"ShowControl"=>TRUE);
$myPicture->drawBezier(20,40,280,170,130,160,160,60,$BezierSettings);

/* We want to draw a dashed bezier curves with visible control points */
/* Additionnaly we'll also draw arrows on both curve ends */
$BezierSettings = array("R"=>255,"G"=>255,"B"=>255,"ShowControl"=>TRUE,"Ticks"=>4,"DrawArrow"=>TRUE,"ArrowTwoHeads"=>TRUE);
$myPicture->drawBezier(360,120,630,120,430,50,560,190,$BezierSettings);
```

This will draw two bezier curves with visible control points. As the shadow support is enabled on line 2, the arrow will be surrounded by a small (+1,+1) shadow.

## drawFilledRectangle - Filled rectangles

This function allows you to draw filled rectangle, optionally you can set a border color. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide. This functions are supporting anti-alias and shadows.

## Calling this function

```
drawFilledRectangle($X1,$Y1,$X2,$Y2,$Format="");
```

Where :

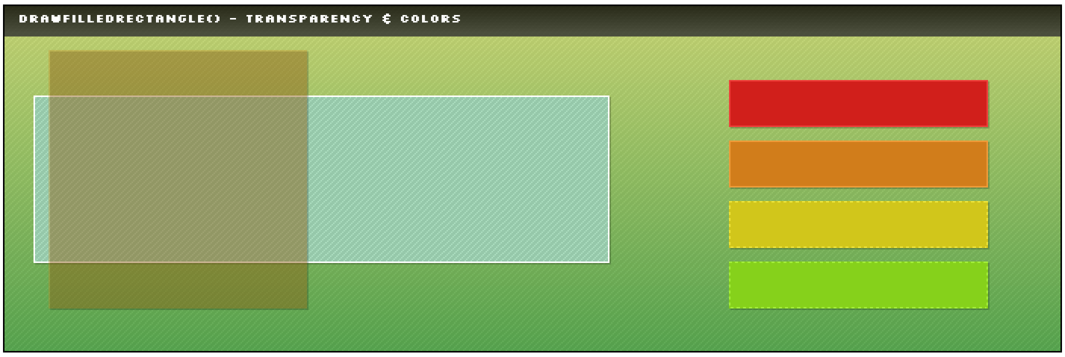
- X1,Y1 are the start coordinate of the rectangle.
- X2,Y2 are the end coordinate of the rectangle.
- Format is an array containing the drawing parameters.

## Customisation array - Tune up your area!

It is possible to customize the box rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the rectangle will be drawn black with no transparency and no border.

- The fill color can be set with R, G, B.
- The line color can be set with BorderR, BorderG, BorderB.
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.
- The alpha transparency factor can be set with Alpha.
- You can draw dashed border lines using the Ticks parameter.
- You can enable zebra lines setting Dash to TRUE.
- you can set the zebra steps with DashStep.
- you can set the zebra color with DashR, DashG, DashB.

## Sample script



Code sample

```
/* Enable shadow support */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* Left green box */

$RectangleSettings = array("R"=>150,"G"=>200,"B"=>170,"Dash"=>TRUE,"DashR"=>170,"DashG"=>220,"DashB"=>190,"BorderR"=>255,

"BorderG"=>255,"BorderB"=>255);

$myPicture->drawFilledRectangle(20,60,400,170,$RectangleSettings);

/* Left orange surrounding box with an alpha factor of 30% */
$RectangleSettings = array("R"=>209,"G"=>134,"B"=>27,"Alpha"=>30);
$myPicture->drawFilledRectangle(30,30,200,200,$RectangleSettings);

/* Right Red box */
$RectangleSettings = array("R"=>209,"G"=>31,"B"=>27,"Alpha"=>100,"Surrounding"=>30);
$myPicture->drawFilledRectangle(480,50,650,80,$RectangleSettings);

/* Right Orange box */
$RectangleSettings = array("R"=>209,"G"=>125,"B"=>27,"Alpha"=>100,"Surrounding"=>30);
$myPicture->drawFilledRectangle(480,90,650,120,$RectangleSettings);
```

```
 /* Right Yellow box */
$RectangleSettings = array("R"=>209,"G"=>198,"B"=>27,"Alpha"=>100,"Surrounding"=>30,"Ticks"=>2);
$myPicture->drawFilledRectangle(480,130,650,160,$RectangleSettings);

/* Right Green box */
$RectangleSettings = array("R"=>134,"G"=>209,"B"=>27,"Alpha"=>100,"Surrounding"=>30,"Ticks"=>2);
$myPicture->drawFilledRectangle(480,170,650,200,$RectangleSettings);
```

This will draw some transparency test on the left side of the picture and 4 colored boxes on the right. The right boxes will be surrounded by a +30 colored border (R+30,G+30,B+30). Shadow support is enabled on a (+1,+1) basis.

# Charting functions

# drawStackedAreaChart - Draw a stacked area chart

This function allows you to draw a stacked area chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

Be careful

For this function to work correctly, the scale must be drawn first in the SCALE_MODE_ADDALL mode.

## Calling this function

```
drawStackedAreaChart($Format="");
```

Where :

- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

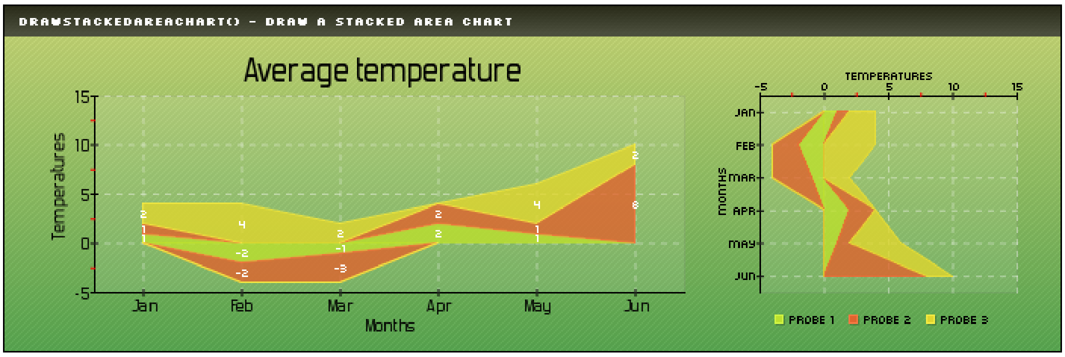
It is possible to customize the stacked bar chart rendering by playing with this array. Providing a detailled configuration is not mandatory.

- You can specify if the values must be written on the chart setting DisplayValues to TRUE or FALSE.
- You can specify the labels color with DisplayR,DisplayG,DisplayB.
- You can specify an unique border color with BorderR,BorderG,BorderB,.
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.
- You can force the area transparency playing with ForceTransparency.

You can define if the labels color will be the same than the serie color of if you want to force it manually setting **DisplayColor** to :

- DISPLAY_MANUAL, color will be specified manually with DisplayR,DisplayG,DisplayB.
- DISPLAY_AUTO, will use the serie color.

## Sample script

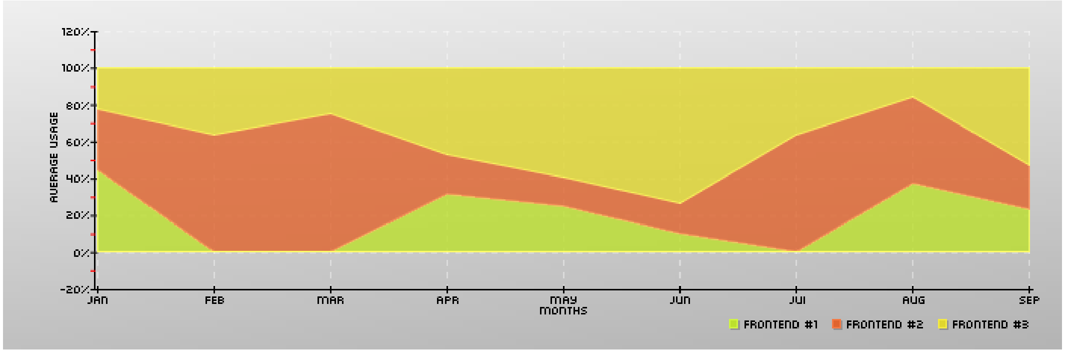


Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(1,-2,-1,2,1,0),"Probe 1");
$MyData->addPoints(array(1,-2,-3,2,1,8),"Probe 2");
$MyData->addPoints(array(2,4,2,0,4,2),"Probe 3");
$MyData->setSerieTicks("Probe 2",4);
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");
```

```
/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);



$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>80));


/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));


/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawStackedAreaChart() - draw a stacked area chart",array("R"=>255,"G"=>255,"B"=>255));

/* Write the chart title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));
$myPicture->drawText(250,55,"Average temperature",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));

/* Draw the scale and the 1st chart */
$myPicture->setGraphArea(60,60,450,190);
$myPicture->drawFilledRectangle(60,60,450,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE,"Mode"=>SCALE_MODE_ADDALL));
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->setShadow(FALSE);
$myPicture->drawStackedAreaChart(array("DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO,"Surrounding"=>20));

/* Draw the scale and the 2nd chart */
$myPicture->setGraphArea(500,60,670,190);
$myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"Mode"=>SCALE_MODE_ADDALL,"DrawSubTicks"=>TRUE));
$myPicture->setShadow(FALSE);
$myPicture->drawStackedAreaChart(array("Surrounding"=>10));

/* Write the chart legend */
$myPicture->drawLegend(510,205,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawStackedAreaChart.png");
```

## 2nd Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class&.phpquot;);
include("../class/pImage.class&.phpquot;);

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(4,0,0,12,8,3,0,12,8),"Frontend #1");
$MyData->addPoints(array(3,12,15,8,5,5,12,15,8),"Frontend #2");
$MyData->addPoints(array(2,7,5,18,19,22,7,5,18),"Frontend #3");
$MyData->setAxisName(0,"Average Usage");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun","Jui","Aug","Sep"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Normalize the data series to 100% */
$MyData->normalize(100,"%");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,"En
dB"=>180,"Alpha"=>100));

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_HORIZONTAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,
"EndB"=>180,"Alpha"=>20));

/* Set the default font properties */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

/* Draw the scale and the chart */
$myPicture->setGraphArea(60,20,680,190);
$myPicture->drawScale(array("XMargin"=>2,"DrawSubTicks"=>TRUE,"Mode"=>SCALE_MODE_ADDALL));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawStackedAreaChart(array("Surrounding"=>60));
$myPicture->setShadow(FALSE);

/* Write the chart legend */
```

```
$myPicture->drawLegend(480,210,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawStackedAreaChart.normalized.png");
```

# drawSplineChart - Draw a spline chart

This function allows you to draw a spline chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
drawSplineChart($Format="");
```

Where :

- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

It is possible to customize the spline chart rendering by playing with this array. Providing a detailled configuration is not mandatory.

- You can specify if the values must be written on the chart setting DisplayValues to TRUE or FALSE.
- You can specify the text UP/RIGHT offset with DisplayOffset.
- You can specify the labels color with DisplayR,DisplayG,DisplayB.

You can define if the labels color will be the same than the serie color of if you want to force it manually setting **DisplayColor** to :

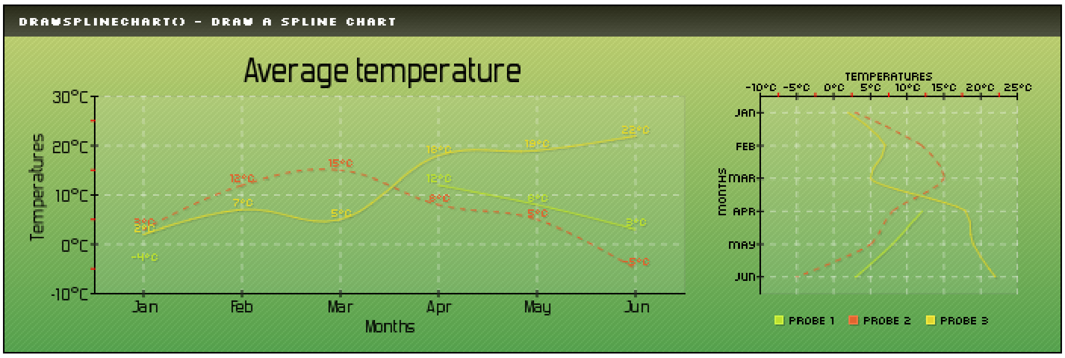
- DISPLAY_MANUAL, color will be specified manually with DisplayR,DisplayG,DisplayB.
- DISPLAY_AUTO, will use the serie color.

## Handling of missing points

By default, the chart will break everytime a **VOID** value will be found. If you want to draw a straight line between the missing points then you can set the **BreakVoid** parameter to FALSE. If you don't set a custom color, the serie color will be used. The missing values rendering parameters can be tuned with :

- The size of the ticks can be set with VoidTicks.
- The line color can be set with BreakR,BreakG,BreakB.

## Sample script



Code sample

```
/* Build a dataset */
$MyData = new pData();
$MyData->addPoints(array(-4,VOID,VOID,12,8,3),"Probe 1");
$MyData->addPoints(array(3,12,15,8,5,-5),"Probe 2");
$MyData->addPoints(array(2,7,5,18,19,22),"Probe 3");
$MyData->setSerieTicks("Probe 2",4);
```

```
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the 1st chart*/
$myPicture->setGraphArea(60,60,450,190);
$myPicture->drawFilledRectangle(60,60,450,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawSplineChart(array("DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO));
$myPicture->setShadow(FALSE);

/* Create the 2nd chart */
$myPicture->setGraphArea(500,60,670,190);
$myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawSplineChart();
$myPicture->setShadow(FALSE);

/* Write the legend*/
$myPicture->drawLegend(510,205,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));
```

# drawProgress - Draw percentage progress bars

This function allows you to draw customisable progress bars. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

> Information
>
> To draw progress bars you don‘t need to have a pData object. Only one value between 0-100 is required.

## Calling this function

```
drawProgress($X,$Y,$Percent,$Format="");
```

Where :

- X1,Y1 are the coordinates where the box will start.
- Percent is a value between 0-100.
- Format is an array containing the drawing parameters of the progress bar.

## Customisation array - Tune up your bars!

It is possible to customize the progress bar rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the bar will be drawn with a width of 200px, a height of 20px and basic color sets.

- The filled area color can be set with R, G, B.
- The filled area border color can be set with BorderR, BorderG, BorderB.
- The control area filling color can be set with BoxBackrR, BoxBackG, BoxBackB.
- The control area border color can be set with BoxBorderR, BoxBorderG, BoxBorderB.
- You can use the Surrounding option to define the border color of the filled area. This value will be added to the R, G, B factors to define the border color.
- You can use the BoxSurrounding option to define the border color of the control. This value will be added to the BoxBackrR, BoxBackrG, BoxBackrB factors to define the border color.
- You can make labels visible setting ShowLabel to TRUE.
- You can set the text spacing with Margin.
- You can define the control orientation with Orientation (See below).
- You can define where the labels will be put with LabelPos (See below).

- The alpha transparency factor can be set with Alpha.
- You can break the angles of surrounding area by setting NoAngle to TRUE.

If you want to fill the area with a gradient box instead of a solid color, you can specify :

- The gradient end color can be set with RFade, GFade, BFade.

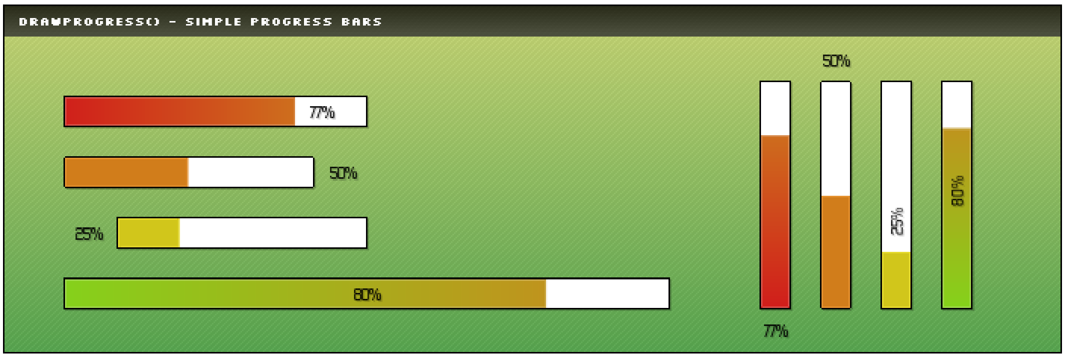
Information

If you use the gradient option, you can define an alpha surrounding box around the drawn aread playing with the Surrounding parameter. This parameter will be used as the alpha value of the surrounding box (small values like 30 will produce the best visual effect).

The control orientation can be either set to *ORIENTATION_HORIZONTAL* or *ORIENTATION_VERTICAL*.

Depending of the orientation choosen, the label position can be set to *LABEL_POS_LEFT*, *LABEL_POS_CENTER*, *LABEL_POS_RIGHT*, *LABEL_POS_TOP*, *LABEL_POS_BOTTOM*, *LABEL_POS_INSIDE*.

## Sample script



Code sample

```
/* Enable shadow support */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));

/* Left Red bar */

$progressOptions = array $progressOptions = array("R"=>209, "G"=>31, "B"=>27, "Surrounding"=>20, "BoxBorderR"=>0, "BoxBorderG"=>0,

"BoxBorderB"=>0, "BoxBackR"=>255, "BoxBackG"=>255, "BoxBackB"=>255, "RFade"=>206, "GFade"=>133, "BFade"=>30, "ShowLabel"=>TRUE);

$myPicture->drawProgress(40,60,77,$progressOptions);

/* Left Orange bar */

$progressOptions = array("Width"=>165, "R"=>209, "G"=>125, "B"=>27, "Surrounding"=>20, "BoxBorderR"=>0, "BoxBorderG"=>0, "BoxBorderB"=>0,

"BoxBackR"=>255, "BoxBackG"=>255, "BoxBackB"=>255, "NoAngle"=>TRUE, "ShowLabel"=>TRUE, "LabelPos"=>LABEL_POS_RIGHT);

$myPicture->drawProgress(40,100,50,$progressOptions);

/* Left Yellow bar */

$progressOptions = array("Width"=>165, "R"=>209, "G"=>198, "B"=>27, "Surrounding"=>20, "BoxBorderR"=>0, "BoxBorderG"=>0, "BoxBorderB"=>0,

"BoxBackR"=>255, "BoxBackG"=>255, "BoxBackB"=>255, "ShowLabel"=>TRUE, "LabelPos"=>LABEL_POS_LEFT);

$myPicture->drawProgress(75,140,25,$progressOptions);

/* Left Green bar */

$progressOptions = array("Width"=>400, "R"=>134, "G"=>209, "B"=>27, "Surrounding"=>20, "BoxBorderR"=>0, "BoxBorderG"=>0, "BoxBorderB"=>0,


"BoxBackR"=>255, "BoxBackG"=>255, "BoxBackB"=>255, "RFade"=>206, "GFade"=>133, "BFade"=>30, "ShowLabel"=>TRUE,
"LabelPos"=>LABEL_POS_CENTER);
$myPicture->drawProgress(40,180,80,$progressOptions);
/* Right vertical Red bar */
$progressOptions = array("Width"=>20, "Height"=>150, "R"=>209, "G"=>31, "B"=>27, "Surrounding"=>20, "BoxBorderR"=>0, "BoxBorderG"=>0,
"BoxBorderB"=>0, "BoxBackR"=>255, "BoxBackG"=>255, "BoxBackB"=>255, "RFade"=>206, "GFade"=>133, "BFade"=>30, "ShowLabel"=>TRUE,
"Orientation"=>ORIENTATION_VERTICAL, "LabelPos"=>LABEL_POS_BOTTOM);
$myPicture->drawProgress(500,200,77,$progressOptions);
/* Right vertical Orange bar */
$progressOptions = array("Width"=>20, "Height"=>150, "R"=>209, "G"=>125, "B"=>27, "Surrounding"=>20, "BoxBorderR"=>0, "BoxBorderG"=>0,
"BoxBorderB"=>0, "BoxBackR"=>255, "BoxBackG"=>255, "BoxBackB"=>255, "NoAngle"=>TRUE, "ShowLabel"=>TRUE,
"Orientation"=>ORIENTATION_VERTICAL, "LabelPos"=>LABEL_POS_TOP);
$myPicture->drawProgress(540,200,50,$progressOptions);
/* Right vertical Yellow bar */
$progressOptions = array("Width"=>20, "Height"=>150, "R"=>209, "G"=>198, "B"=>27, "Surrounding"=>20, "BoxBorderR"=>0, "BoxBorderG"=>0,
"BoxBorderB"=>0, "BoxBackR"=>255, "BoxBackG"=>255, "BoxBackB"=>255, "ShowLabel"=>TRUE, "Orientation"=>ORIENTATION_VERTICAL,
"LabelPos"=>LABEL_POS_INSIDE);
$myPicture->drawProgress(580,200,25,$progressOptions);
/* Right vertical Green bar */
$progressOptions = array("Width"=>20, "Height"=>150, "R"=>134, "G"=>209, "B"=>27, "Surrounding"=>20, "BoxBorderR"=>0, "BoxBorderG"=>0,
"BoxBorderB"=>0, "BoxBackR"=>255, "BoxBackG"=>255, "BoxBackB"=>255, "RFade"=>206, "GFade"=>133, "BFade"=>30, "ShowLabel"=>TRUE,
"Orientation"=>ORIENTATION_VERTICAL, "LabelPos"=>LABEL_POS_CENTER);
$myPicture->drawProgress(620,200,80,$progressOptions);
```

This sample script may look a bit tricky because of the multiple options that can be configured on the progress bars controls. You can also call this function without any additional parameters :

[STARTCOL]

[NEXTCOL]

[NEXTCOL]

Code sample

```
/* Draw a basic progress bar */
$myPicture->drawProgress(40,60,77);
```

[ENDCOL]

# drawSplitPath - Draw splitted path charts

This function allows you to draw a splitted path chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

Information

This drawing function is stored in an external class called pSplit.class.php, you must include it to your project before calling it (see the example below)

## Calling this function

```
drawSplitPath($Object,$Values,$Format);
```

Where :

- Object is reference to a pImage object.
- Values is reference to a pData object.
- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

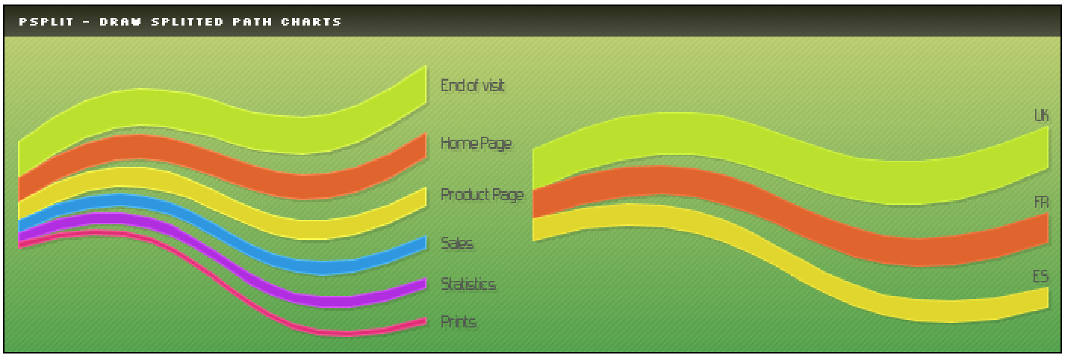
It is possible to customize the splitted path chart rendering by playing with this array. Providing a detailled configuration is not mandatory.

- You can specify if the vertical spacing between two path with Spacing.
- You can specify the text padding with TextPadding.
- You can specify the text position with TextPos
- You can specify the force of the wave with Force (default is 70)
- You can specify the number of segments with Segment (default is 15)
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.

The text position (TextPos) can be :

- TEXT_POS_TOP, this will put the path legend on top of the curve.
- TEXT_POS_RIGHT, this will put the path legent at the right side.

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pSplit.class.php");
include("../class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(700,230);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);


$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>80));


/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pSplit - Draw splitted path charts",array("R"=>255,"G"=>255,"B"=>255));

/* Set the default font properties */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>10,"R"=>80,"G"=>80,"B"=>80));

/* Enable shadow computing */
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(30,20,15,10,8,4),"Score");
$MyData->addPoints(array("End of visit","Home Page","Product Page","Sales","Statistics","Prints"),"Labels");
$MyData->setAbscissa("Labels");

/* Create the pSplit object */
$SplitChart = new pSplit();

/* Draw the split chart */
$Settings = array("TextPos"=>TEXT_POS_RIGHT,"TextPadding"=>10,"Spacing"=>20,"Surrounding"=>40);
$myPicture->setGraphArea(10,20,340,230);
$SplitChart->drawSplitPath($myPicture,$MyData,$Settings);

/* Create and populate the pData object */
$MyData2 = new pData();
$MyData2->addPoints(array(30,20,15),"Score");
$MyData2->addPoints(array("UK","FR","ES"),"Labels");
$MyData2->setAbscissa("Labels");

/* Draw the split chart */
$Settings = array("TextPadding"=>4,"Spacing"=>30,"Surrounding"=>20);
$myPicture->setGraphArea(350,50,690,200);
$SplitChart->drawSplitPath($myPicture,$MyData2,$Settings);

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.split.png");
```

# drawPlotChart - Draw a plot chart

This function allows you to draw a plot chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
drawPlotChart($Format="");
```

Where :

- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

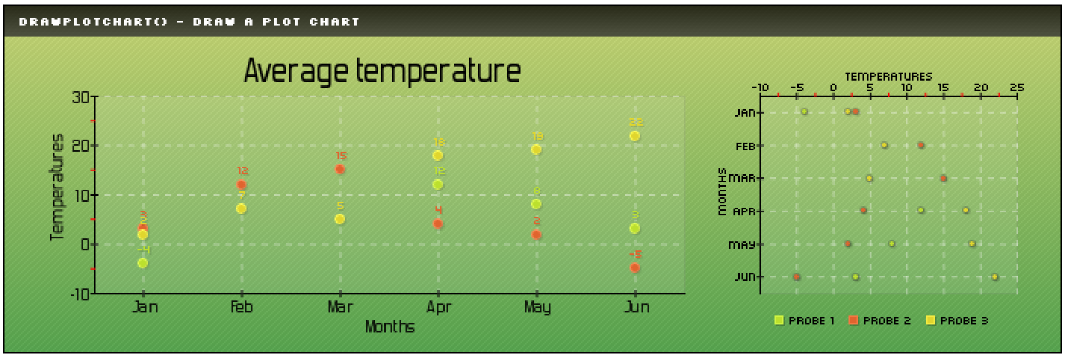
It is possible to customize the plot chart rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the bar will be drawn with a width of 200px, a height of 20px and basic color sets.

- You can specify the size of the plot with PlotSize.
- You can draw a border around the plot setting PlotBorder to TRUE or FALSE.
- You can set the plot border color with BorderR, BorderG, BorderB.
- You can set the plot border alpha with BorderAlpha.
- You can set the width of the border with BorderSize.
- You can use the Surrounding option to define the border color. This value will be added to the R, G, B factors to define the new color.
- You can specify if the values must be written on the chart setting DisplayValues to TRUE or FALSE.
- You can specify the text UP/RIGHT offset with DisplayOffset.
- You can specify the labels color with DisplayR,DisplayG,DisplayB.

You can define if the labels color will be the same than the serie color of if you want to force it manually setting **DisplayColor** to :

- DISPLAY_MANUAL, color will be specified manually with DisplayR,DisplayG,DisplayB.
- DISPLAY_AUTO, will use the serie color.

## Sample script



Code sample

```
/* Build a dataset */
$MyData = new pData();
$MyData->addPoints(array(-4,VOID,VOID,12,8,3),"Probe 1");
$MyData->addPoints(array(3,12,15,8,5,-5),"Probe 2");
$MyData->addPoints(array(2,7,5,18,19,22),"Probe 3");
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the 1st chart*/
$myPicture->setGraphArea(60,60,450,190);
```

```
$myPicture->drawFilledRectangle(60,60,450,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));

$myPicture->drawPlotChart(array("BorderSize"=>1, "Surrounding"=>40, "BorderAlpha"=>100, "PlotSize"=>2, "PlotBorder"=>TRUE,

"DisplayValues"=>TRUE, "DisplayColor"=>DISPLAY_AUTO));

$myPicture->setShadow(FALSE);

/* Create the 2nd chart */
$myPicture->setGraphArea(500,60,670,190);
$myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawPlotChart(array("PlotSize"=>1,"PlotBorder"=>TRUE,"BorderSize"=>1));
$myPicture->setShadow(FALSE);

/* Write the legend*/
$myPicture->drawLegend(510,205,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));
```

# drawLineChart - Draw a line chart

This function allows you to draw a line chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
drawLineChart($Format="");
```

Where :

- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

It is possible to customize the spline chart rendering by playing with this array. Providing a detailled configuration is not mandatory.

- You can specify if the values must be written on the chart setting DisplayValues to TRUE or FALSE.
- You can specify the text UP/RIGHT offset with DisplayOffset.
- You can specify the labels color with DisplayR,DisplayG,DisplayB.

You can define if the labels color will be the same than the serie color of if you want to force it manually setting **DisplayColor** to :

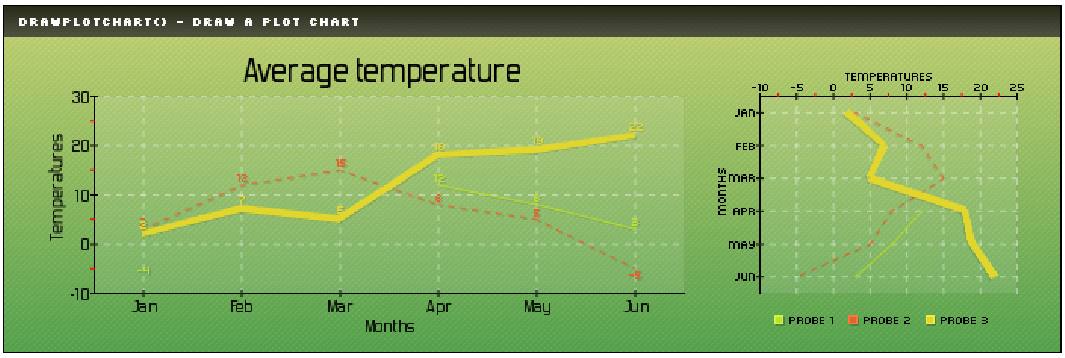
- DISPLAY_MANUAL, color will be specified manually with DisplayR,DisplayG,DisplayB.
- DISPLAY_AUTO, will use the serie color.

## Handling of missing points

By default, the chart will break everytime a **VOID** value will be found. If you want to draw a straight line between the missing points then you can set the **BreakVoid** parameter to FALSE. If you don't set a custom color, the serie color will be used. The missing values rendering parameters can be tuned with :

- The size of the ticks can be set with VoidTicks.
- The line color can be set with BreakR,BreakG,BreakB.

## Sample script



Code sample

```
/* Build a dataset */
$MyData = new pData();
$MyData->addPoints(array(-4,VOID,VOID,12,8,3),"Probe 1");
$MyData->addPoints(array(3,12,15,8,5,-5),"Probe 2");
$MyData->addPoints(array(2,7,5,18,19,22),"Probe 3");
$MyData->setSerieTicks("Probe 2",4);
$MyData->setSerieWeight("Probe 3",2);
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the 1st chart*/
$myPicture->setGraphArea(60,60,450,190);
$myPicture->drawFilledRectangle(60,60,450,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawLineChart(array("DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO));
$myPicture->setShadow(FALSE);

/* Create the 2nd chart */
$myPicture->setGraphArea(500,60,670,190);
$myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>-1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawLineChart();
$myPicture->setShadow(FALSE);

/* Write the legend*/
$myPicture->drawLegend(510,205,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));
```

# drawBarChart - Draw a bar chart

This function allows you to draw a bar chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

Information

Gradients overlay can only work today on boxed bars. If you turn the Rounded parameter to TRUE, only the solid color will be drawn.

[ERR]It is recommended to keep the **XMargin** parameter of the drawScale function to **AUTO** (default value) in order to avoid out-of-scale rendering.[/ERR]

## Calling this function

```
drawBarChart($Format="");
```

Where :

- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

It is possible to customize the bar chart rendering by playing with this array. Providing a detailled configuration is not mandatory.

- You can specify if the values must be written on the chart setting DisplayValues to TRUE or FALSE.
- You can specify the text UP/RIGHT offset with DisplayOffset.
- You can specify if you want that a shadow will be drawn under the labels setting DisplayShadow to TRUE.
- You can specify the labels color with DisplayR,DisplayG,DisplayB.
- You can specify if the area are wrapped around the 0 line setting AroundZero to TRUE.
- You can draw area with rounded corners setting Rounded to TRUE.
- You can specify an unique border color with BorderR,BorderG,BorderB,.
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.
- You can draw the bars with a gradient overlay setting Gradient to TRUE.
- You can change the way the gradient bars will be drawn setting GradientMode to GRADIENT_EFFECT_CAN.
- You can specify the gradient starting color with GradientStartR,GradientStartG,GradientStartB.
- You can specify the gradient ending color with GradientEndR,GradientEndG,GradientEndtB.
- You can specify the gradient alpha with GradientAlpha.
- You can override the 0 value by another static one with Floating0Value.
- You can override the 0 value by using another data serie giving it's name through Floating0Serie.
- You can highlight the 0-bottom of the bars setting Draw0Line to TRUE.

If you only have one data serie, you may want to override the palette to have each bar represented with a different color. To achieve this, just pass a palette formated array with **OverrideColors**. (see the last sample script)

You can define if the labels color will be the same than the serie color of if you want to force it manually setting **DisplayColor** to :

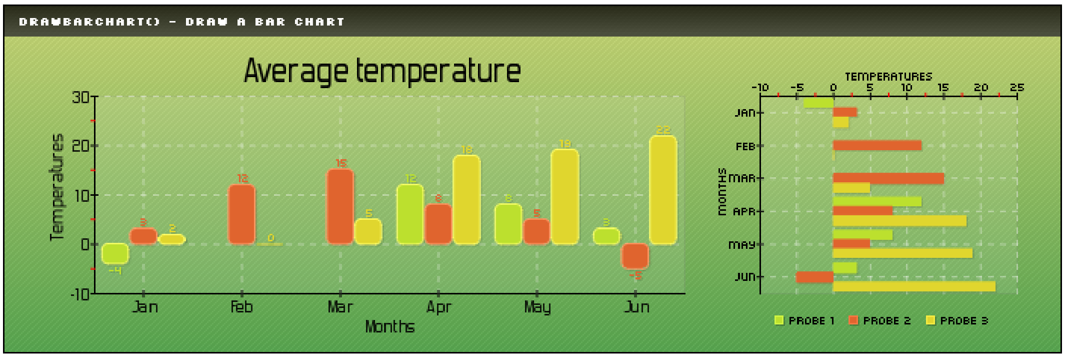
- DISPLAY_MANUAL, color will be specified manually with DisplayR,DisplayG,DisplayB.
- DISPLAY_AUTO, will use the serie color.

You can specify if the labels will be displayed inside or outside the bars seting **DisplayColor** to :

- LABEL_POS_INSIDE, values will be written inside the bars (if possible).
- LABEL_POS_OUTSIDE, values will be written outside the bars (default).

The default gradient overlay (if you only set **Gradient** to *TRUE*) will be a white to black with 40% alpha.

## Sample script

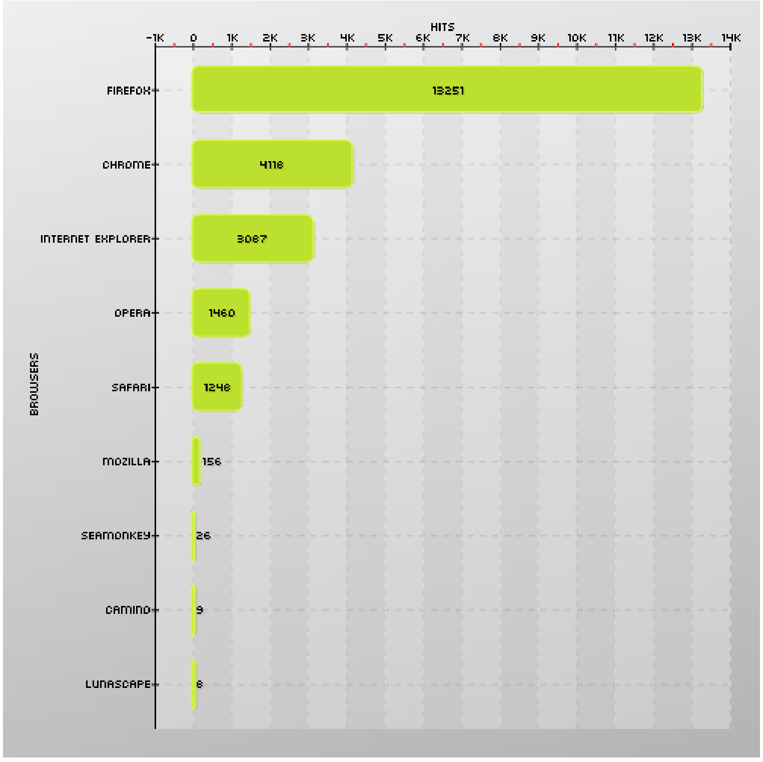


Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
```

```
$MyData = new pData();
$MyData->addPoints(array(-4,VOID,VOID,12,8,3),"Probe 1");
$MyData->addPoints(array(3,12,15,8,5,-5),"Probe 2");
$MyData->addPoints(array(2,0,5,18,19,22),"Probe 3");
$MyData->setSerieTicks("Probe 2",4);
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);



$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>80));


/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));


/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawBarChart() - draw a bar chart",array("R"=>255,"G"=>255,"B"=>255));

/* Write the chart title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));
$myPicture->drawText(250,55,"Average temperature",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));

/* Draw the scale and the 1st chart */
$myPicture->setGraphArea(60,60,450,190);
$myPicture->drawFilledRectangle(60,60,450,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawBarChart(array("DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO,"Rounded"=>TRUE,"Surrounding"=>60));
$myPicture->setShadow(FALSE);

/* Draw the scale and the 2nd chart */
$myPicture->setGraphArea(500,60,670,190);
$myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawBarChart();
$myPicture->setShadow(FALSE);

/* Write the chart legend */
$myPicture->drawLegend(510,205,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawBarChart.png");
```

## Second Sample script

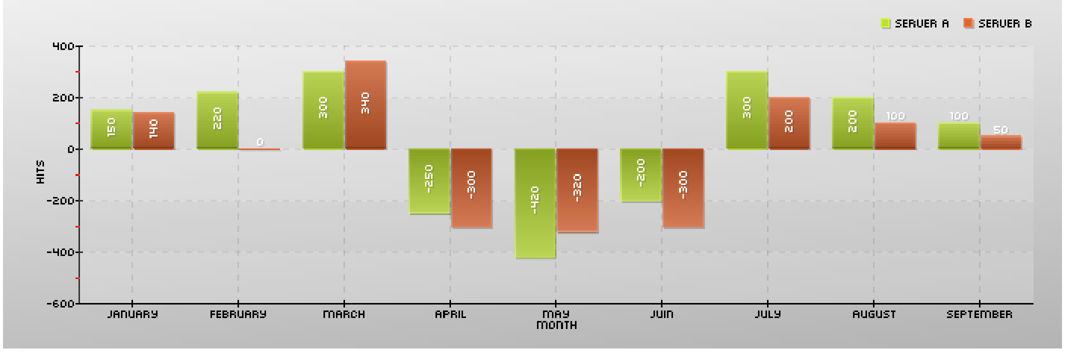


Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(13251,4118,3087,1460,1248,156,26,9,8),"Hits");
$MyData->setAxisName(0,"Hits");
$MyData->addPoints(array("Firefox","Chrome","Internet Explorer","Opera","Safari","Mozilla","SeaMonkey","Camino","Lunascape"),"Browsers");
$MyData->setSerieDescription("Browsers","Browsers");
$MyData->setAbscissa("Browsers");

/* Create the pChart object */
$myPicture = new pImage(500,500,$MyData);

$myPicture->drawGradientArea(0,0,500,500,DIRECTION_VERTICAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,"EndB"=>180,"Alpha"=>100));

$myPicture->drawGradientArea(0,0,500,500,DIRECTION_HORIZONTAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,"EndB"=>180,"Alpha"=>20));

$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
```

```
/* Draw the chart scale */
$myPicture->setGraphArea(100,30,480,480);

$myPicture->drawScale(array("CycleBackground"=>TRUE,"DrawSubTicks"=>TRUE,"GridR"=>0,"GridG"=>0,"GridB"=>0,"GridAlpha"=>10,

"Pos"=>SCALE_POS_TOPBOTTOM));


/* Turn on shadow computing */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Draw the chart */
$myPicture->drawBarChart(array("DisplayPos"=>LABEL_POS_INSIDE,"DisplayValues"=>TRUE,"Rounded"=>TRUE,"Surrounding"=>30));

/* Write the legend */
$myPicture->drawLegend(570,215,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawBarChart.vertical.png");
```

## Third Sample script

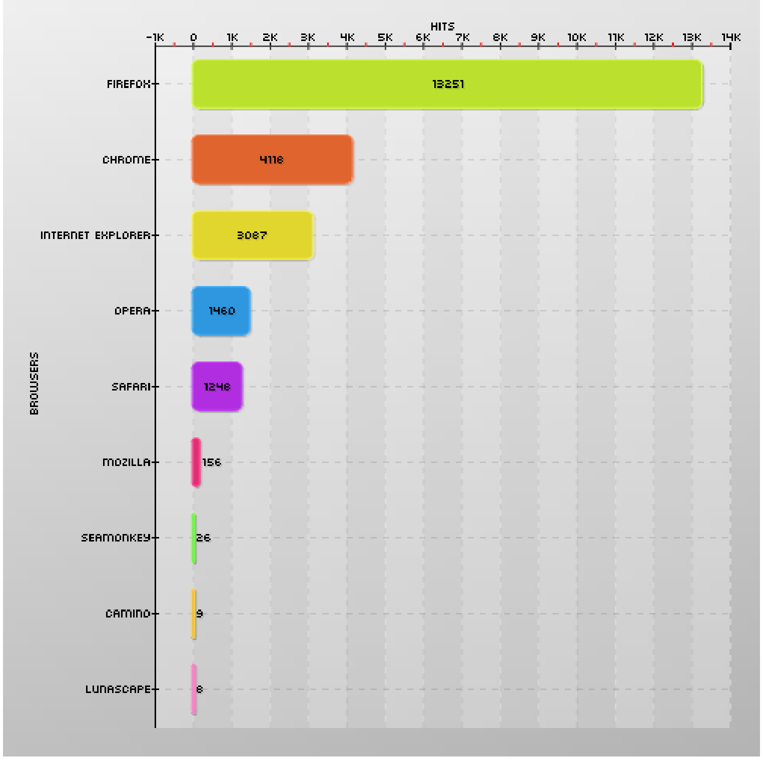


Code sample

```
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pImage.class.php");

$MyData = new pData();
$MyData->addPoints(array(150,220,300,-250,-420,-200,300,200,100),"Server A");
$MyData->addPoints(array(140,240,340,-300,-320,-300,200,100,50),"Server B");
$MyData->setAxisName(0,"Hits");

$MyData->addPoints(array("January","February","March","April","May","Juin","July","August","September"),"Months");
$MyData->setSerieDescription("Months","Month");
$MyData->setAbscissa("Months");

$myPicture = new pImage(800,300,$MyData);



$myPicture->drawGradientArea(0,0,800,300,DIRECTION_VERTICAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,"En

dB"=>180,"Alpha"=>100));
```

```
$myPicture->drawGradientArea(0,0,800,300,DIRECTION_HORIZONTAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,

"EndB"=>180,"Alpha"=>20));

 $myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));

 $myPicture->setGraphArea(50,30,780,270);
 $myPicture->drawScale(array("CycleBackground"=>TRUE,"DrawSubTicks"=>TRUE,"GridR"=>0,"GridG"=>0,"GridB"=>0,"GridAlpha"=>10));

 $myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

 $settings = array("Gradient"=>TRUE, "DisplayPos"=>LABEL_POS_INSIDE, "DisplayValues"=>TRUE, "DisplayR"=>255, "DisplayG"=>255,

"DisplayB"=>255, "DisplayShadow"=>TRUE, "Surrounding"=>30);
 $myPicture->drawBarChart($settings);

 $myPicture->drawLegend(650,12,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

 $myPicture->Render("drawbarchart3.png");
```

## Customized palette sample script

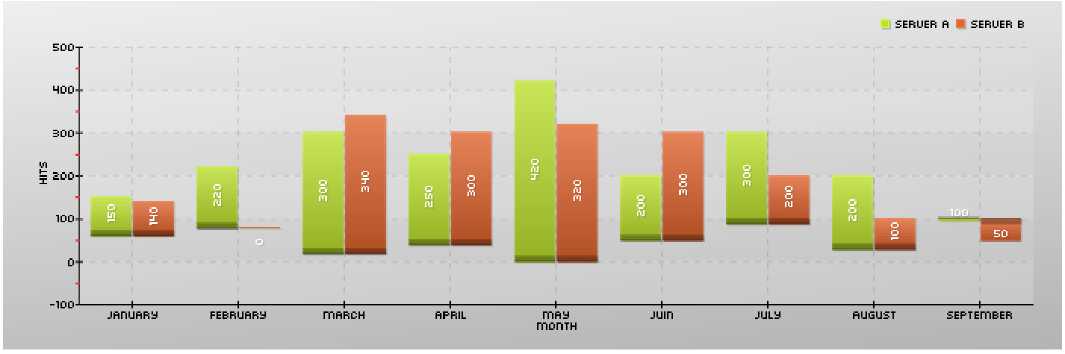


Code sample

```
 /* pChart library inclusions */
 include("../class/pData.class.php");
 include("../class/pDraw.class.php");
 include("../class/pImage.class.php");

 /* Create and populate the pData object */
 $MyData = new pData();
 $MyData->addPoints(array(13251,4118,3087,1460,1248,156,26,9,8),"Hits");
 $MyData->setAxisName(0,"Hits");
 $MyData->addPoints(array("Firefox","Chrome","Internet Explorer","Opera","Safari","Mozilla","SeaMonkey","Camino","Lunascape"),"Browsers");
 $MyData->setSerieDescription("Browsers","Browsers");
```

```
$MyData->setAbscissa("Browsers");

/* Create the pChart object */
$myPicture = new pImage(500,500,$MyData);

$myPicture->drawGradientArea(0,0,500,500,DIRECTION_VERTICAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,"En
dB"=>180,"Alpha"=>100));

$myPicture->drawGradientArea(0,0,500,500,DIRECTION_HORIZONTAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,
"EndB"=>180,"Alpha"=>20));
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

/* Draw the chart scale */
$myPicture->setGraphArea(100,30,480,480);
$myPicture->drawScale(array("CycleBackground"=>TRUE,"DrawSubTicks"=>TRUE,"GridR"=>0,"GridG"=>0,"GridB"=>0,"GridAlpha"=>10,
"Pos"=>SCALE_POS_TOPBOTTOM));

/* Turn on shadow computing */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Create the per bar palette */
$Palette = array("0"=>array("R"=>188,"G"=>224,"B"=>46,"Alpha"=>100),
                 "1"=>array("R"=>224,"G"=>100,"B"=>46,"Alpha"=>100),
                 "2"=>array("R"=>224,"G"=>214,"B"=>46,"Alpha"=>100),
                 "3"=>array("R"=>46,"G"=>151,"B"=>224,"Alpha"=>100),
                 "4"=>array("R"=>176,"G"=>46,"B"=>224,"Alpha"=>100),
                 "5"=>array("R"=>224,"G"=>46,"B"=>117,"Alpha"=>100),
                 "6"=>array("R"=>92,"G"=>224,"B"=>46,"Alpha"=>100),
                 "7"=>array("R"=>224,"G"=>176,"B"=>46,"Alpha"=>100));

/* Draw the chart */

$myPicture->drawBarChart(array("DisplayPos"=>LABEL_POS_INSIDE,"DisplayValues"=>TRUE,"Rounded"=>TRUE,"Surrounding"=>30,"OverrideColors
"=>$Palette));

/* Write the legend */
$myPicture->drawLegend(570,215,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawBarChart.palette.png");
```

## Floating 0 sample script



Code sample

```
 /* pChart library inclusions */
 include("../class/pData.class.php");
 include("../class/pDraw.class.php");
 include("../class/pImage.class.php");

 /* Create and populate the pData object */
 $MyData = new pData();
 $MyData->addPoints(array(150,220,300,250,420,200,300,200,100),"Server A");
 $MyData->addPoints(array(140,0,340,300,320,300,200,100,50),"Server B");
 $MyData->setAxisName(0,"Hits");
 $MyData->addPoints(array("January","February","March","April","May","Juin","July","August","September"),"Months");
 $MyData->setSerieDescription("Months","Month");
 $MyData->setAbscissa("Months");

 /* Create the floating 0 data serie */
 $MyData->addPoints(array(60,80,20,40,0,50,90,30,100),"Floating 0");
 $MyData->setSerieDrawable("Floating 0",FALSE);

 /* Create the pChart object */
 $myPicture = new pImage(700,230,$MyData);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,"EndB"=>180,"Alpha"=>100));
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_HORIZONTAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,"EndB"=>180,"Alpha"=>20));
 $myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

 /* Draw the scale  */
 $myPicture->setGraphArea(50,30,680,200);
 $myPicture->drawScale(array("CycleBackground"=>TRUE,"DrawSubTicks"=>TRUE,"GridR"=>0,"GridG"=>0,"GridB"=>0,"GridAlpha"=>10));

 /* Turn on shadow computing */
 $myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

 /* Draw the chart */
```

```
$settings = array("Floating0Serie"=>"Floating 0","Draw0Line"=>TRUE,"Gradient"=>TRUE,

"DisplayPos"=>LABEL_POS_INSIDE,"DisplayValues"=>TRUE,"DisplayR"=>255,

"DisplayG"=>255,"DisplayB"=>255,"DisplayShadow"=>TRUE,"Surrounding"=>10);

$myPicture->drawBarChart($settings);

/* Write the chart legend */
$myPicture->drawLegend(580,12,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawBarChart.floating.png");
```

# drawStackedBarChart - Draw a bar chart

This function allows you to draw a stacked bar chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

Be careful

For this function to work correctly, the scale must be drawn first in the SCALE_MODE_ADDALL mode.

Information

Gradients overlay can only work today on boxed bars. If you turn the Rounded parameter to TRUE, only the solid color will be drawn.

## Calling this function

```
drawStackedBarChart($Format="");
```

Where :

- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

It is possible to customize the stacked bar chart rendering by playing with this array. Providing a detailled configuration is not mandatory.

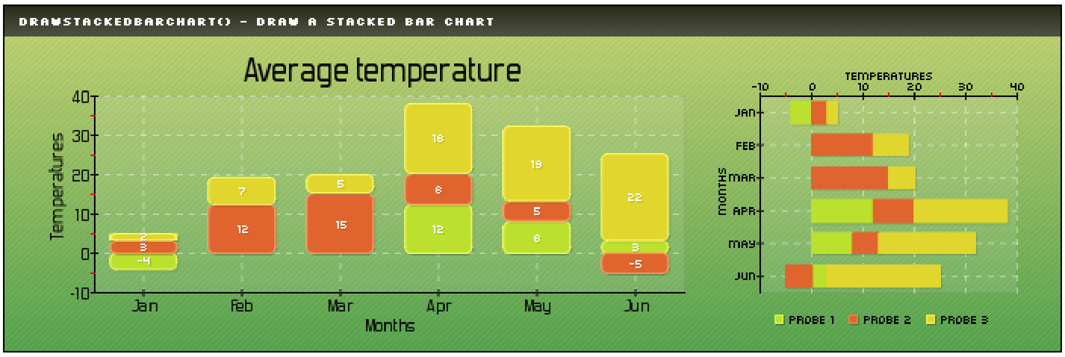
- You can specify if the values must be written on the chart setting DisplayValues to TRUE or FALSE.
- You can specify the labels color with DisplayR,DisplayG,DisplayB.
- You can specify if the area are wrapped around the 0 line setting AroundZero to TRUE.
- You can specify the interleave between two bars with Interleave (default is .5)
- You can draw area with rounded corners setting Rounded to TRUE.
- You can specify an unique border color with BorderR,BorderG,BorderB,.
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.
- You can draw the bars with a gradient overlay setting Gradient to TRUE.
- You can specify the gradient starting color with GradientStartR,GradientStartG,GradientStartB.
- You can specify the gradient ending color with GradientEndR,GradientEndG,GradientEndtB.
- You can specify the gradient alpha with GradientAlpha.

You can define if the labels color will be the same than the serie color of if you want to force it manually setting **DisplayColor** to :

- DISPLAY_MANUAL, color will be specified manually with DisplayR,DisplayG,DisplayB.
- DISPLAY_AUTO, will use the serie color.

The default gradient overlay (if you only set **Gradient** to *TRUE*) will be a white to black with 40% alpha.

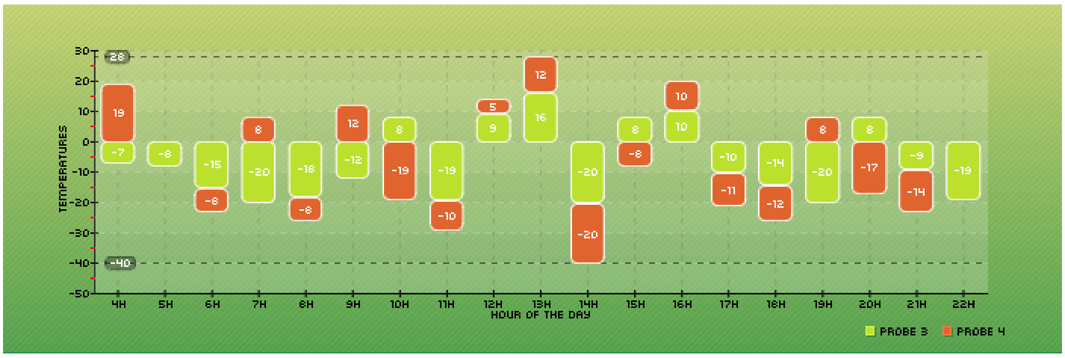
## Sample script



Code sample

```
/* Build a dataset */
$MyData = new pData();
$MyData->addPoints(array(-4,VOID,VOID,12,8,3),"Probe 1");
$MyData->addPoints(array(3,12,15,8,5,-5),"Probe 2");
$MyData->addPoints(array(2,7,5,18,19,22),"Probe 3");
$MyData->setSerieTicks("Probe 2",4);
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the 1st chart*/
$myPicture->setGraphArea(60,60,450,190);
$myPicture->drawFilledRectangle(60,60,450,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE,"Mode"=>SCALE_MODE_ADDALL));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawStackedBarChart(array("DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO,"Rounded"=>TRUE,"Surrounding"=>60));
$myPicture->setShadow(FALSE);

/* Create the 2nd chart */
$myPicture->setGraphArea(500,60,670,190);
$myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"DrawSubTicks"=>TRUE,"Mode"=>SCALE_MODE_ADDALL));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawStackedBarChart();
$myPicture->setShadow(FALSE);

/* Write the legend*/
$myPicture->drawLegend(510,205,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));
```

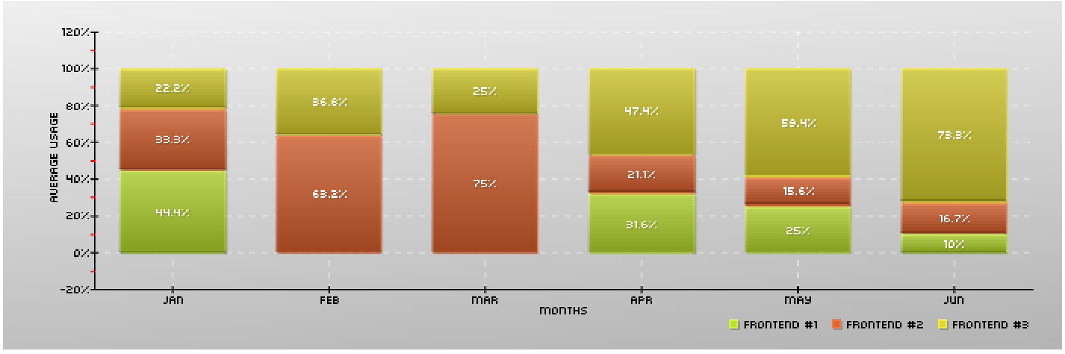
## Second Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(-7,-8,-15,-20,-18,-12,8,-19,9,16,-20,8,10,-10,-14,-20,8,-9,-19),"Probe 3");
$MyData->addPoints(array(19,0,-8,8,-8,12,-19,-10,5,12,-20,-8,10,-11,-12,8,-17,-14,0),"Probe 4");
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array(4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22),"Time");
$MyData->setSerieDescription("Time","Hour of the day");
$MyData->setAbscissa("Time");
$MyData->setXAxisUnit("h");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

/* Set the default font properties */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

/* Draw the scale */
$myPicture->setGraphArea(60,30,650,190);
$myPicture->drawScale(array("CycleBackground"=>TRUE,"DrawSubTicks"=>TRUE,"GridR"=>0,"GridG"=>0,"GridB"=>0,"GridAlpha"=>10,

"Mode"=>SCALE_MODE_ADDALL));


/* Turn on shadow computing */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Draw some thresholds */
$myPicture->setShadow(FALSE);
$myPicture->drawThreshold(-40,array("WriteCaption"=>TRUE,"R"=>0,"G"=>0,"B"=>0,"Ticks"=>4));
$myPicture->drawThreshold(28,array("WriteCaption"=>TRUE,"R"=>0,"G"=>0,"B"=>0,"Ticks"=>4));

/* Draw the chart */



$myPicture->drawStackedBarChart(array("Rounded"=>TRUE,"DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO,"DisplaySize"=>6,"BorderR"=>


255, "BorderG"=>255,"BorderB"=>255));


/* Write the chart legend */
$myPicture->drawLegend(570,212,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawStackedBarChart.rounded.png");
```

## Third Sample script



Code sample

```
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pImage.class.php");

$MyData = new pData();
$MyData->addPoints(array(-4,VOID,VOID,12,8,3),"Frontend #1");
$MyData->addPoints(array(3,12,15,8,5,-5),"Frontend #2");
$MyData->addPoints(array(2,7,5,18,19,22),"Frontend #3");
$MyData->setAxisName(0,"Average Usage");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

$MyData->normalize(100,"%");

$myPicture = new pImage(700,230,$MyData);

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,"En
dB"=>180,"Alpha"=>100));

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_HORIZONTAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,
"EndB"=>180,"Alpha"=>20));

$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));

$myPicture->setGraphArea(60,20,680,190);
$myPicture->drawScale(array("DrawSubTicks"=>TRUE,"Mode"=>SCALE_MODE_ADDALL));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawStackedBarChart(array("DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO,"Gradient"=>TRUE,"Surrounding"=>60));
$myPicture->setShadow(FALSE);

$myPicture->drawLegend(480,210,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

$myPicture->Render("drawStackedBarChart3.png");
```

# drawFilledSplineChart - Draw a filled spline chart

This function allows you to draw a filled spline chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
drawFilledSplineChart($Format="");
```

Where :

- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

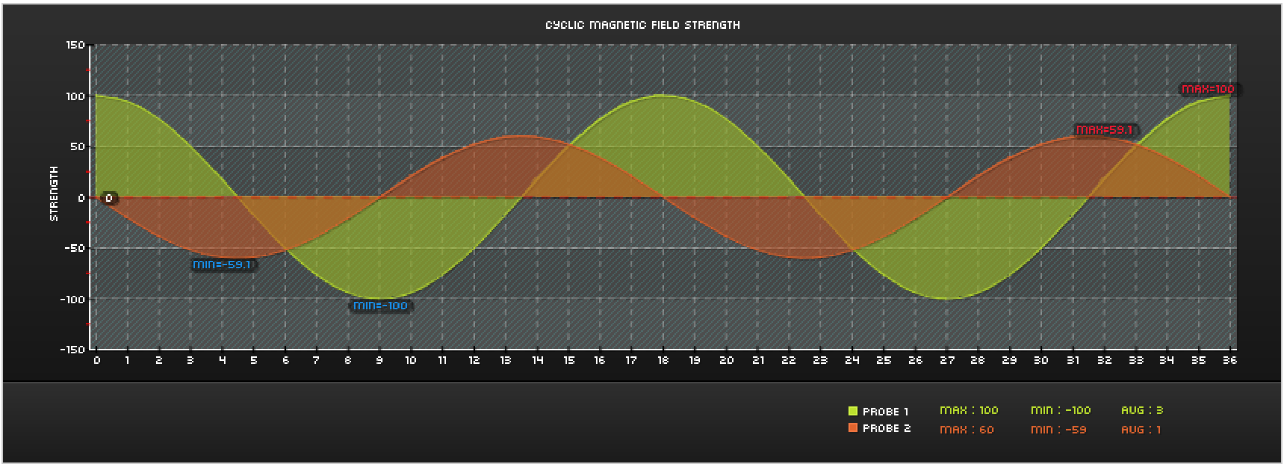
It is possible to customize the area chart rendering by playing with this array. Providing a detailled configuration is not mandatory.

- You can specify if the values must be written on the chart setting DisplayValues to TRUE or FALSE.
- You can specify the text UP/RIGHT offset with DisplayOffset.
- You can specify the labels color with DisplayR,DisplayG,DisplayB.
- You can specify if the area are wrapped around the 0 line setting AroundZero to TRUE.

You can define if the labels color will be the same than the serie color of if you want to force it manually setting **DisplayColor** to :

- DISPLAY_MANUAL, color will be specified manually with DisplayR,DisplayG,DisplayB.
- DISPLAY_AUTO, will use the serie color.

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->setAxisName(0,"Strength");
for($i=0;$i<=720;$i=$i+20)
 {
  $MyData->addPoints(cos(deg2rad($i))*100,"Probe 1");
  $MyData->addPoints(cos(deg2rad($i+90))*60,"Probe 2");
 }

/* Create the pChart object */
$myPicture = new pImage(847,304,$MyData);
```

```
$myPicture->drawGradientArea(0,0,847,304,DIRECTION_VERTICAL,array("StartR"=>47,"StartG"=>47,"StartB"=>47,

"EndR"=>17,"EndG"=>17,"EndB"=>17,"Alpha"=>100));

$myPicture->drawGradientArea(0,250,847,304,DIRECTION_VERTICAL,array("StartR"=>47,"StartG"=>47,"StartB"=>47,

"EndR"=>27,"EndG"=>27,"EndB"=>27,"Alpha"=>100));
$myPicture->drawLine(0,249,847,249,array("R"=>0,"G"=>0,"B"=>0));
$myPicture->drawLine(0,250,847,250,array("R"=>70,"G"=>70,"B"=>70));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,846,303,array("R"=>204,"G"=>204,"B"=>204));


/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawText(423,14,"Cyclic magnetic field strength",array("R"=>255,"G"=>255,"B"=>255,"Align"=>TEXT_ALIGN_MIDDLEMIDDLE));

/* Define the chart area */
$myPicture->setGraphArea(58,27,816,228);

/* Draw a rectangle */
$myPicture->drawFilledRectangle(58,27,816,228,array("R"=>0,"G"=>0,"B"=>0,"Dash"=>TRUE,"DashR"=>0,

"DashG"=>51,"DashB"=>51,"BorderR"=>0,"BorderG"=>0,"BorderB"=>0));


/* Turn on shadow computing */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));


/* Draw the scale */
$myPicture->setFontProperties(array("R"=>255,"G"=>255,"B"=>255));
$ScaleSettings = array("XMargin"=>4,"DrawSubTicks"=>TRUE,"GridR"=>255,

"GridG"=>255,"GridB"=>255,"AxisR"=>255,"AxisG"=>255,"AxisB"=>255,"GridAlpha"=>30,"CycleBackground"=>TRUE);
$myPicture->drawScale($ScaleSettings);

/* Draw the spline chart */
$myPicture->drawFilledSplineChart();

/* Write the chart boundaries */
$BoundsSettings = array("MaxDisplayR"=>237,"MaxDisplayG"=>23,"MaxDisplayB"=>48, "MinDisplayR"=>23,"MinDisplayG"=>144,"MinDisplayB"=>237);
$myPicture->writeBounds(BOUND_BOTH,$BoundsSettings);

/* Write the 0 line */
$myPicture->drawThreshold(0,array("WriteCaption"=>TRUE));

/* Write the chart legend */
$myPicture->setFontProperties(array("R"=>255,"G"=>255,"B"=>255));
$myPicture->drawLegend(560,266,array("Style"=>LEGEND_NOBORDER));

/* Write the 1st data series statistics */
$Settings = array("R"=>188,"G"=>224,"B"=>46,"Align"=>TEXT_ALIGN_BOTTOMLEFT);
$myPicture->drawText(620,270,"Max : ".ceil($MyData->getMax("Probe 1")),$Settings);
$myPicture->drawText(680,270,"Min : ".ceil($MyData->getMin("Probe 1")),$Settings);
$myPicture->drawText(740,270,"Avg : ".ceil($MyData->getSerieAverage("Probe 1")),$Settings);

/* Write the 2nd data series statistics */
$Settings = array("R"=>224,"G"=>100,"B"=>46,"Align"=>TEXT_ALIGN_BOTTOMLEFT);
$myPicture->drawText(620,283,"Max : ".ceil($MyData->getMax("Probe 2")),$Settings);
$myPicture->drawText(680,283,"Min : ".ceil($MyData->getMin("Probe 2")),$Settings);
```

```
$myPicture->drawText(740,283,"Avg : ".ceil($MyData->getSerieAverage("Probe 2")),$Settings);

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawFilledSplineChart.png");
```

# drawAreaChart - Draw an area chart

This function allows you to draw an area chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
drawAreaChart($Format="");
```

Where :

- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

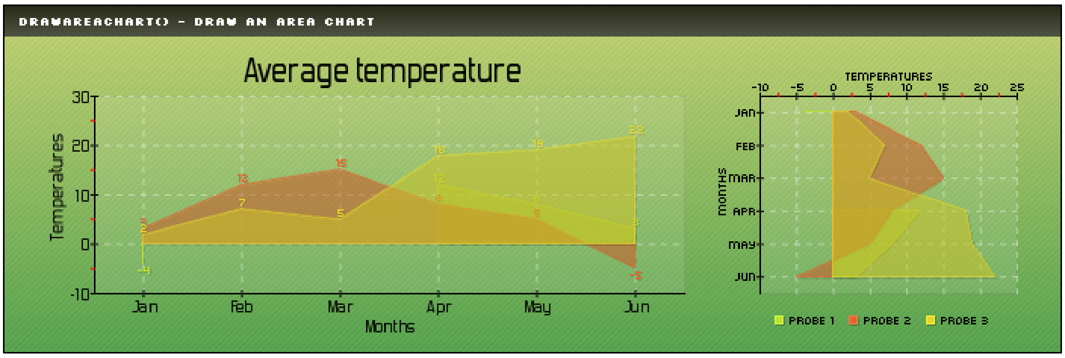
It is possible to customize the area chart rendering by playing with this array. Providing a detailled configuration is not mandatory.

- You can specify if the values must be written on the chart setting DisplayValues to TRUE or FALSE.
- You can specify the text UP/RIGHT offset with DisplayOffset.
- You can specify the labels color with DisplayR,DisplayG,DisplayB.
- You can specify an override factor for the area transparency with ForceTransparency.
- You can specify if the area are wrapped around the 0 line setting AroundZero to TRUE.

You can define if the labels color will be the same than the serie color of if you want to force it manually setting **DisplayColor** to :

- DISPLAY_MANUAL, color will be specified manually with DisplayR,DisplayG,DisplayB.
- DISPLAY_AUTO, will use the serie color.

## Sample script



Code sample

```
/* Build a dataset */
$MyData = new pData();
$MyData->addPoints(array(-4,VOID,VOID,12,8,3),"Probe 1");
$MyData->addPoints(array(3,12,15,8,5,-5),"Probe 2");
$MyData->addPoints(array(2,7,5,18,19,22),"Probe 3");
$MyData->setSerieTicks("Probe 2",4);
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the 1st chart*/
$myPicture->setGraphArea(60,60,450,190);
```

```
$myPicture->drawFilledRectangle(60,60,450,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawAreaChart(array("DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO));
$myPicture->setShadow(FALSE);

/* Create the 2nd chart */
$myPicture->setGraphArea(500,60,670,190);
$myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawAreaChart();
$myPicture->setShadow(FALSE);

/* Write the legend*/
$myPicture->drawLegend(510,205,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));
```

## drawBestFit - Draw the "line of best fit"

This function allows you to draw the line of best fit associated with your given data series. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide. The mathematical formula used to compute it is described below :

$$Slope = m = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum(x^2) - (\sum x)^2}$$

$$Intercept = b = \frac{\sum y - m(\sum x)}{n}$$

### Calling this function

```
drawBestFit($Format="");
```

Where :

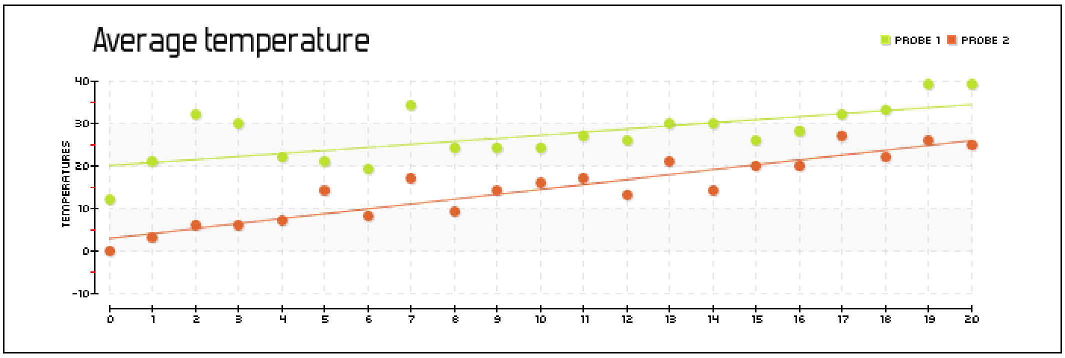
- Format is an array containing the drawing parameters of the chart. (today none are defined)

### Customisation array - Tune up your chart!

It is possible to customize the rendering by playing with this array. Providing a detailled configuration is not mandatory. You'll see below a representation of all the customization possible :

- You can optionally specify the line ticks width with Ticks.

### Sample script



Code sample

```
 /* pChart library inclusions */
 include("../class/pData.class.php");
 include("../class/pDraw.class.php");
 include("../class/pImage.class.php");

 /* Create and populate the pData object */
 $MyData = new pData();
 for($i=0;$i<=20;$i++) { $MyData->addPoints(rand(10,30)+$i,"Probe 1"); }
 for($i=0;$i<=20;$i++) { $MyData->addPoints(rand(0,10)+$i,"Probe 2"); }
 $MyData->setAxisName(0,"Temperatures");

 /* Create the pChart object */
 $myPicture = new pImage(700,230,$MyData);

 /* Turn of Antialiasing */
 $myPicture->Antialias = FALSE;

 /* Add a border to the picture */
 $myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));


 /* Write the chart title */
 $myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));
 $myPicture->drawText(150,35,"Average temperature",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));

 /* Set the default font */
 $myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

 /* Define the chart area */
 $myPicture->setGraphArea(60,40,650,200);

 /* Draw the scale */

 $scaleSettings = array("XMargin"=>10,"YMargin"=>10, "Floating"=>TRUE,"GridR"=>200, "GridG"=>200,"GridB"=>200,"DrawSubTicks"=>TRUE,

"CycleBackground"=>TRUE);

 $myPicture->drawScale($scaleSettings);

 /* Turn on Antialiasing */
 $myPicture->Antialias = TRUE;

 /* Draw the line of best fit */
 $myPicture->drawBestFit();

 /* Turn on shadows */
 $myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

 /* Draw the line chart */
 $myPicture->drawPlotChart();

 /* Write the chart legend */
 $myPicture->drawLegend(580,20,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

 /* Render the picture (choose the best way) */
 $myPicture->autoOutput("pictures/example.drawBestFit.png");
```

# Misc functions

# getHeight- Return the height of your pChart object

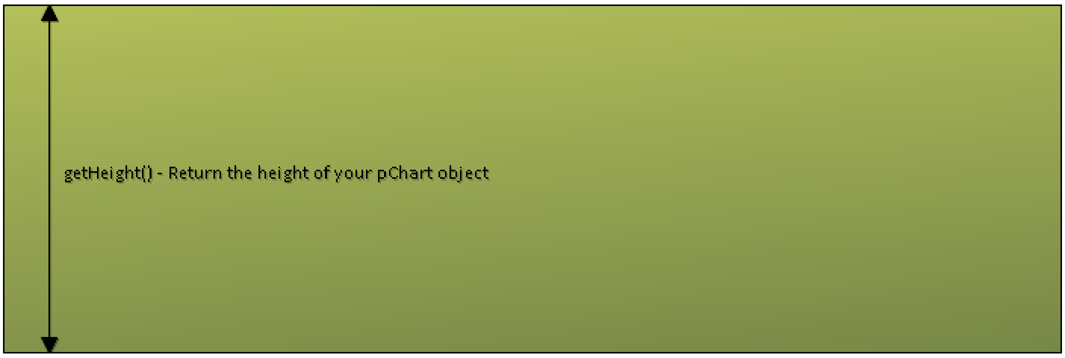
This function returns the height in pixels of your Pchart object. It is used internally by drawing functions.

## Calling this function

```
getHeight();
```

No parameters are required for this function.

## Sample script



Code sample

```
/* Create your pChart object */
$myPicture = new pImage(700,230);

/* Return the angle made by this line and the horizontal axis */
echo $myPicture->getHeight();
```

This will print 230.

# loadPalette - Load a color scheme and apply it

This function allows you to load a color scheme from a flat file or an internet location. You can specify it the loaded palette will be appended to the current one or if it will overwrite the current color scheme.

## Calling this function

```
setPalette($FileName,$Overwrite=FALSE);
```

Where :

- FileName is the name of the file to load. It can be a local path or a remote location.
- Overwrite is an optional parameter. Set to TRUE it will overwrite the current palette.

## What is a color scheme

A color scheme is a text file containing a list of colors. Colors are composed of their Red, Green, Blue and Alpha properties. To create your own color scheme, just create a CSV file using a coma as delimiter between the parameters and a carriage return to separate the colors :

```
239,210,121,100
149,203,233,100
2,71,105,100
175,215,117,100
44,87,0,100
222,157,127,100
```

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Will append the "autumn" palette to the current one */
$MyData->loadPalette("palettes/autumn.color");

/* Will replace the whole color scheme by the "desert" palette */
$MyData->loadPalette("palettes/desert.color", TRUE);
```

## getLegendSize- Determine the size of the legend box

This function returns the width and height of the legend box that would be drawn with the specified parameters. You can use this function to align your legend boxes depending of its printed size.

This function will return an associative array containing the **Width** and **Height** keys.

Information

Basically you‘ll use the same Format array with this function than the one used for the drawLegend() function.

### Calling this function

```
getLegendSize($Format="");
```

Where :

- Format is an array containing the drawing parameters of the legend.

### Customisation array - Tune up your legend!

It is possible to customize the way your legend will be rendered by playing with this array. Providing a detailled configuration is not mandatory, by default the legend will be drawn in a soft grey box with curvy corners.

- You can specify the font file that will be used with FontName.
- You can specify the font size with FontSize.
- You can specify the size of the colored boxes with BoxSize.
- You can specify the inner margins with Margin.

You can choose the style that will be applied to you legend box using the **Style** parameter :

- LEGEND_NOBORDER no borders will be drawn around the legend.
- LEGEND_BOX a rectangle will be drawn around the legend.
- LEGEND_ROUND a rounded rectangle will be drawn around the legend.

You can also define the way the legend will be written using the **Mode** parameter :

- LEGEND_VERTICAL that will stack vertically the series.
- LEGEND_HORIZONTAL that will stack horizontally the series.

### Sample script

Code sample

```
/* Rounded legend box */
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$Size = $myPicture->getLegendSize(array("Style"=>LEGEND_ROUND,"Mode"=>LEGEND_HORIZONTAL));

print_r($Size);
```

The script will returns an associative array :

```
Array
(
   [Width] => 123
   [Height] => 23
)
```

# getAngle- Determine the angle of a line

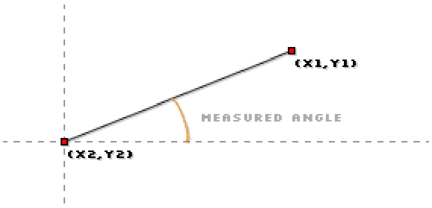
This function returns the angle in degrees made by the specified line and the horizontal axis. It is used internally for some trigonometric computing.

## Calling this function

```
getAngle($X1,$Y1,$X2,$Y2);
```

Where :

- X1,Y1 is the start coordinate of the line.
- X2,Y2 is the end coordinate of the line.



## Sample script

Code sample

```
/* Return the angle made by this line and the horizontal axis */
echo $myPicture->getAngle(10,10,200,200);
```

This will print 45.

# getWidth- Return the width of your pChart object

This function returns the width in pixels of your Pchart object. It is used internally by drawing functions.

## Calling this function

```
getWidth();
```

No parameters are required for this function.

## Sample script

getWidth() - Return the width of your pChart object

Code sample

```
/* Create your pChart object */
$myPicture = new pImage(700,230);
```

```
/* Return the angle made by this line and the horizontal axis */
echo $myPicture->getWidth();
```

This will print 700.

# getLength- Determine the length of a line

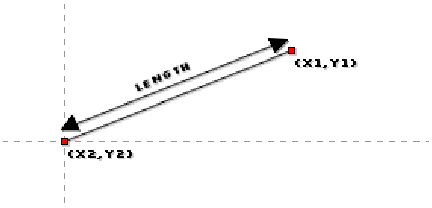
This function returns the length in pixel between two specified points. It is used internally for some trigonometric computing.

## Calling this function

```
getLength($X1,$Y1,$X2,$Y2);
```

Where :

- X1,Y1 is the start coordinate of the line.
- X2,Y2 is the end coordinate of the line.



## Sample script

Code sample

```
/* Return the angle made by this line and the horizontal axis */
echo $myPicture->getLength(10,10,200,200);
```

This will print 268.

# Dataset functions

setSerieDrawable
setXAxisUnit
setAxisDisplay
drawAll
setAbscissa
setSerieDescription

# normalize - Normalize your data series

This function allow you to normalize your data series by providing a 0-value range. This function is usually called when we want to normalize the dataset to a 100% scale. The points that are computed will replace your data series, each data row sum will be equal to the value provided.

Be careful

This function will modify your datasets in a non reversive way.

## Calling this function

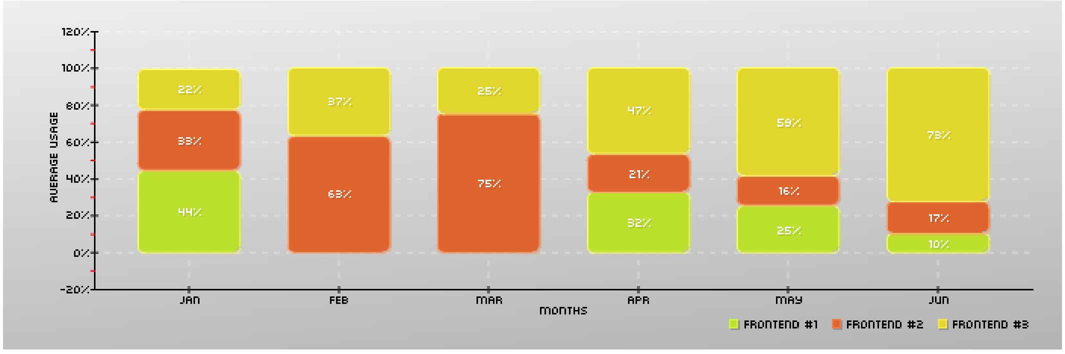
```
normalize($NormalizationFactor=100,$UnitChange=NULL,$Round=0);
```

Where :

- NormalizationFactor is the value used for the normalization process. Usually 100.
- UnitChange, if provided will override the axis unit. Set it to "%" if you use a factor 100.
- Round is the number of decimal value to keep.

All parameters are optional

## Sample script



Code sample

```
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pImage.class.php");

$MyData = new pData();
$MyData->addPoints(array(-4,VOID,VOID,12,8,3),"Frontend #1");
$MyData->addPoints(array(3,12,15,8,5,-5),"Frontend #2");
$MyData->addPoints(array(2,7,5,18,19,22),"Frontend #3");
$MyData->setAxisName(0,"Average Usage");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

$MyData->normalize(100,"%");

$myPicture = new pImage(700,230,$MyData);

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,"EndB"=>180,"Alpha"=>100));
```

```
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_HORIZONTAL,array("StartR"=>240,"StartG"=>240,"StartB"=>240,"EndR"=>180,"EndG"=>180,

"EndB"=>180,"Alpha"=>20));

 $myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));

 $myPicture->setGraphArea(60,20,680,190);
 $myPicture->drawScale(array("DrawSubTicks"=>TRUE,"Mode"=>SCALE_MODE_ADDALL));
 $myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
 $myPicture->drawStackedBarChart(array("DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO,"Rounded"=>TRUE,"Surrounding"=>60));
 $myPicture->setShadow(FALSE);

 $myPicture->drawLegend(480,210,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

 $myPicture->Render("normalize.png");
```

# setAxisXY - Affect an axis as a X or Y member

This function allows you to associate an axis to the X or Y direction. In almost all charts you don‘t need to call this function, it is only usefull while doing X versus Y charts.

Information

By default all new series are binded to the X direction.

## Calling this function

```
setAxisXY($AxisID,$Identity);
```

Where :

- AxisID is the ID of the axis we want to edit.
- Identity is the direction associated to the axis, it can be AXIS_X or AXIS_Y.

## Sample script

Code sample

```
/* Create the pDatа object */
$MyData = new pData();

/* Create dumb data */
$MyData->addPoints(array(2,4,7,9,10),"X Values");
$MyData->addPoints(array(4,3,1,9,4),"Y Values");

/* Associate the series to the axis */
$MyData->setSerieOnAxis("X Values",0); // This one isn't needed as this is the default serie
$MyData->setSerieOnAxis("Y Values",1);

/* Bind the axis to the X & Y directions */
$MyData->setAxisXY(0,AXIS_X);
$MyData->setAxisXY(1,AXIS_Y);
```

# setAxisUnit - Define the unit of one Y Axis

This function allows you to set the unit associated to one Y Axis. This unit will be displayed when the axis display mode is set to **AXIS_FORMAT_UNIT**. Values on this axis are automatically computed by the scaling algorithm.

Information

By default all new series are binded to the default axis with an ID of 0. You can bind a serie to another axis using the setSerieOnAxis function.

## Calling this function

```
setAxisUnit($AxisID,$Unit);
```

Where :

- AxisID is the ID of the axis we want to edit.
- Unit is the unit of this axis.

## Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* The default axis unit will be celsius degrees */
$MyData->setAxisUnit(0,"°C");
```

# getData - Extract your pData contents

This function allow you to export the contents of your pData object in an array format. It can be used for debugging purpose assuming that you understand the pData array concept. You can learn more on it here.

## Calling this function

```
getData();
```

There is no required parameters.

## Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(1,2,3,4),"My Serie 1");

/* This will dump the pData structure contents */
print_r($MyData->getData());
```

This will display something like :

```
Array
(
    [XAxisDisplay] => 680001
    [XAxisFormat] =>
    [XAxisName] =>
    [XAxisUnit] =>
    [Abscissa] =>
    [Axis] => Array
        (
            [0] => Array
                (
                    [Display] => 680001
                    [Position] => 681002
                )

        )

    [Series] => Array
        (
            [My Serie 1] => Array
                (
                    [Description] => My Serie 1
                    [isDrawable] => 1
                    [Picture] =>
                    [Max] => 4
                    [Min] => 1
                    [Axis] => 0
                    [Color] => Array
                        (
                            [R] => 188
                            [G] => 224
                            [b] => 46
                        )

                    [Data] => Array
                        (
                            [0] => 1
                            [1] => 2
                            [2] => 3
                            [3] => 4
                        )

                )

        )

    [Min] => 1
    [Max] => 4
)
```

# setXAxisName - Define the name of the X Axis

This function allows you to set the name of the X Axis. This name will be displayed when drawing the chart area under the abscissa axis. Values on this axis can be defined with the setAbscissa function.

## Calling this function

```
setXAxisName($Name);
```

Where :

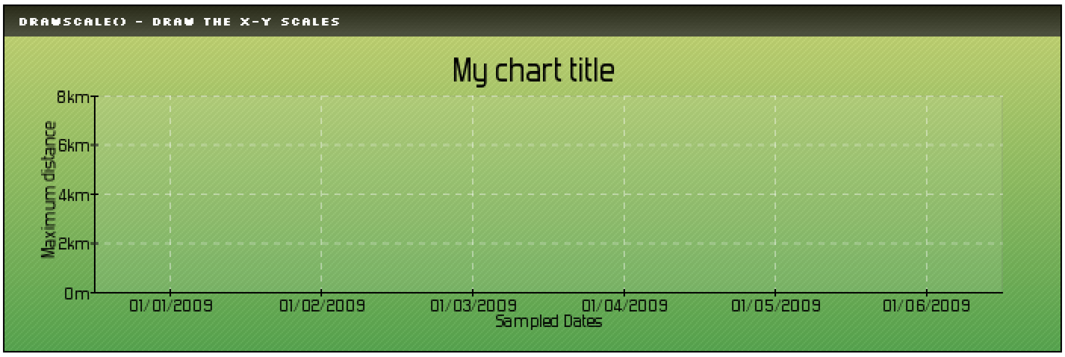
- Name is the name that will be put on the X axis.

## Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* The abscissa axis will be named "Temperatures" */
$MyData->setXAxisName("Temperatures");
```

### Sample script #2



Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData = new pData();
$MyData->addPoints(array(1700,2500,7800,4500,3150),"Distance");
$MyData->setAxisName(0,"Maximum distance");
$MyData->setAxisUnit(0,"m");
$MyData->setAxisDisplay(0,AXIS_FORMAT_METRIC);

/* Create the X serie */
$MyData->addPoints(array(1230768000,1233446400,1235865600,1238544000,1241136000,1243814400),"Timestamp");
$MyData->setSerieDescription("Timestamp","Sampled Dates");
$MyData->setAbscissa("Timestamp");
$MyData->setXAxisDisplay(AXIS_FORMAT_DATE);

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(60,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(60,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
$myPicture->drawScale();
```

# getSerieCount - Retrieve value at specified index in given serie

This function will return you the value at the specified index of the given serie.

Information

The index are starting from 0.

### Calling this function

```
getValueAt($Serie,$Index=0);
```

Where :

- Serie is the name of the serie.
- Index is the index of the value to retrieve.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Add some points to "My serie 1" */
$MyData->addPoints(array(1,2,3,4),"My serie 1");
```

```
/* Display the value at index #1 in "My serie 1" */
echo $MyData->getValueAt("My serie 1",1);
```

This will display 2.

# setXAxisDisplay - Define the way to show values on absissa Axis

This function allows you to set the kind of data to display on the X Axis. Depending on the type choosen, data will be computed in different ways.

Information

AXIS_FORMAT_DEFAULT is set as the default abscissa display mode, this allow you to put either text or numeric data into the abscissa data serie.

## Calling this function

```
setXAxisDisplay($Mode,$Format=NULL);
```

Where :

- Mode is the kind of data that will be displayed.
- Format is the way to display in the associated mode.

Valid **Format** values are :

- AXIS_FORMAT_DEFAULT to display raw data.
- AXIS_FORMAT_TIME to convert to time (HH:MM:SS).
- AXIS_FORMAT_DATE to convert to date (D/M/Y).
- AXIS_FORMAT_METRIC to convert to metric values (1k).
- AXIS_FORMAT_CURRENCY to convert to curencies (1.200€).

The way data is displayed can be modified using the **Format** parameter. Valid formats are :

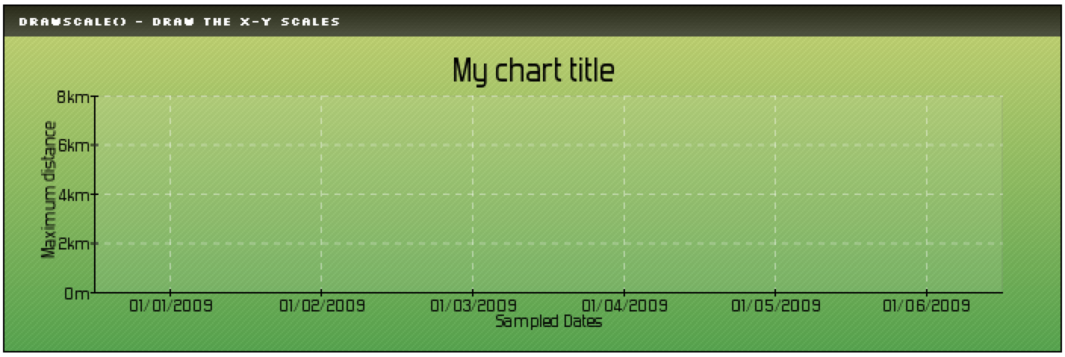
- For AXIS_FORMAT_TIME values you may provide a PHP date format. (like "H:i:s")
- For AXIS_FORMAT_DATE values you may provide a PHP date format. (like "d/m/y")
- For AXIS_FORMAT_CURENCY values you may provide a currency. (like $ or €)

## Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* The X axis will use time with a custom display mask */
$MyData->setXAxisDisplay(AXIS_FORMAT_TIME,"H:i");
```

## Sample script #2



Code sample

```
$MyData = new pData();
```

```
/* Prepare some nice data & axis config */
$MyData = new pData();
$MyData->addPoints(array(1700,2500,7800,4500,3150),"Distance");
$MyData->setAxisName(0,"Maximum distance");
$MyData->setAxisUnit(0,"m");
$MyData->setAxisDisplay(0,AXIS_FORMAT_METRIC);

/* Create the X serie */
$MyData->addPoints(array(1230768000,1233446400,1235865600,1238544000,1241136000,1243814400),"Timestamp");
$MyData->setSerieDescription("Timestamp","Sampled Dates");
$MyData->setAbscissa("Timestamp");
$MyData->setXAxisDisplay(AXIS_FORMAT_DATE);

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(60,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(60,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
$myPicture->drawScale();
```

# getMax - Return the biggest value of a given serie

This function will return you the biggest value contained in the given serie.

## Calling this function

```
getMax($Serie);
```

Where :

- Serie is the name of the serie.

## Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Add a some points to "My serie1" */
$MyData->addPoints(array(1,7,8,2,6,4),"My serie 1");

/* Display TRUE as our pData object now contains data */
echo $MyData->getMax("My serie 1");
```

This will display 8.

# setAxisPosition - Define the position of one Y Axis

This function allows you to set position (left or right) of an Y Axis. Values on this axis are automatically computed by the scaling algorithm.

Information

By default all new series are binded to the default axis with an ID of 0. You can bind a serie to another axis using the setSerieOnAxis function.

## Calling this function

```
setAxisPosition($AxisID,$Position);
```

Where :

- AxisID is the ID of the axis we want to name.

- Position is the side where the axis will be drawn.

Valid **Position** values are :

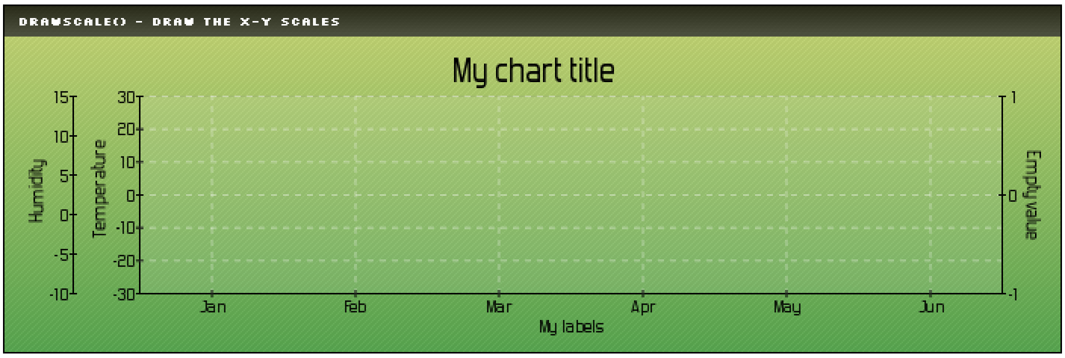
- AXIS_POSITION_LEFT default positioning on the left side of the chart.
- AXIS_POSITION_RIGHT to put the axis on the right side of the chart.

## Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* The default ordinate axis will be on the left side*/
$MyData->setAxisPosition(0,AXIS_FORMAT_METRIC);

/* The second ordinate axis will be on the right side */
$MyData->setAxisPosition(1,AXIS_POSITION_RIGHT);
```

## Sample script #2



Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData->addPoints(array(24,-25,26,25,25),"Temperature");
$MyData->addPoints(array(1,2,VOID,9,10),"Humidity 1");
$MyData->addPoints(array(1,VOID,7,-9,0),"Humidity 2");
$MyData->addPoints(array(-1,-1,-1,-1,-1),"Humidity 3");
$MyData->addPoints(array(0,0,0,0,0),"Vide");
$MyData->setSerieOnAxis("Temperature",0);
$MyData->setSerieOnAxis("Humidity 1",1);
$MyData->setSerieOnAxis("Humidity 2",1);
$MyData->setSerieOnAxis("Humidity 3",1);
$MyData->setSerieOnAxis("Vide",2);
$MyData->setAxisPosition(2,AXIS_POSITION_RIGHT);
$MyData->setAxisName(0,"Temperature");
$MyData->setAxisName(1,"Humidity");
$MyData->setAxisName(2,"Empty value");

/* Bind a data serie to the X axis */
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","My labels");
$MyData->setAbscissa("Labels");

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(90,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(90,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
```

```
$myPicture->drawScale(array("DrawYLines"=>array(0)));
```

# getSerieCount - Number of values contained in the given serie

This function will return you the number of values contained in the given serie.

## Calling this function

```
getSerieCount($Serie);
```

Where :

- Serie is the name of the serie.

## Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Add some points to "My serie 1" */
$MyData->addPoints(array(1,2,3,4),"My serie 1");

/* Display the number of values in "My serie 1" */
echo $MyData->getSerieCount("My serie 1");
```

This will display 4.

# getSerieAverage - Return the x[SUP]th[/SUP] percentile of the given serie

This function will return you the x[SUP]th[/SUP]percentile of the given serie.

## Calling this function

```
getSeriePercentile($Serie="Serie1",$Percentil=95);
```

Where :

- Serie is the name of the serie.
- Percentil is the percentil required.

## Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Add some points to "My serie 1" */
$MyData->addPoints(array(1,2,3,4),"My serie 1");

/* Display the 95th percentil of "My serie 1" */
echo $MyData->getSeriePercentile("My serie 1");
```

This will display 4.

# getSerieAverage - Return the average value of the given serie

This function will return you the average value of the given serie.

### Calling this function

```
getSerieAverage($Serie);
```

Where :

- Serie is the name of the serie.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Add some points to "My serie 1" */
$MyData->addPoints(array(1,2,3,4),"My serie 1");

/* Display the average value of "My serie 1" */
echo $MyData->getSerieAverage("My serie 1");
```

This will display 2.5.

## setAxisColor - Set the color of a specific axis

This function allows you to specify the color of a specific axis. If you specify a color for an axis, this will overide the default colors you can pass to the drawScale function. Parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

### Calling this function

```
setAxisColor($AxisID,$Format="");
```

Where :

- AxisID is the ID of the axis you want to customize.
- Format is an array containing the drawing parameters of the arrow.

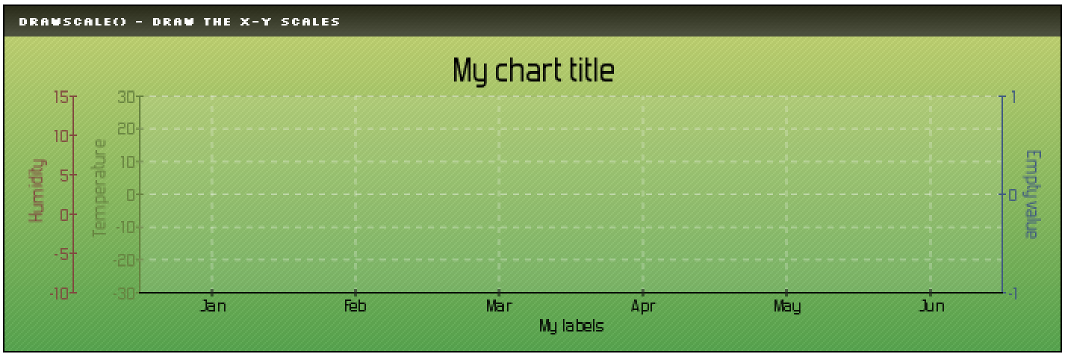
[ERR]This function will overide the axis color, tick color and font color of your axis.[/ERR]

### Customisation array - Tune up your axis!

It is possible to customize the color of the axis by playing with this array.

- Axis color can be set with R, G, B.

### Sample script



Code sample

```
/* Create your pData object */
$MyData = new pData();
$MyData->addPoints(array(24,-25,26,25,25),"Temperature");
$MyData->addPoints(array(1,2,VOID,9,10),"Humidity 1");
```

```
$MyData->addPoints(array(1,VOID,7,-9,0),"Humidity 2");
$MyData->addPoints(array(-1,-1,-1,-1,-1),"Humidity 3");
$MyData->addPoints(array(0,0,0,0,0),"Vide");
$MyData->setSerieOnAxis("Temperature",0);
$MyData->setSerieOnAxis("Humidity 1",1);
$MyData->setSerieOnAxis("Humidity 2",1);
$MyData->setSerieOnAxis("Humidity 3",1);
$MyData->setSerieOnAxis("Vide",2);
$MyData->setAxisPosition(2,AXIS_POSITION_RIGHT);
$MyData->setAxisName(0,"Temperature");
$MyData->setAxisName(1,"Humidity");
$MyData->setAxisName(2,"Empty value");

/* Associate a color to each axis */
$MyData->setAxisColor(0,array("R"=>102,"G"=>129,"B"=>63));
$MyData->setAxisColor(1,array("R"=>129,"G"=>72,"B"=>63));
$MyData->setAxisColor(2,array("R"=>63,"G"=>89,"B"=>129));

/* Create a labelled X axis */
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","My labels");
$MyData->setAbscissa("Labels");
```

# setSerieDrawable- Add a description to one serie

You can use this function to enable / disable the "Drawable" flag of a serie. When you‘ll call a charting function, only "Drawable" series will be represented.

Information

When creating a serie it will automatically be marked as drawable.

## Calling this function

```
setSerieDrawable($Serie,$Drawable=TRUE);
```

Where :

- Serie is the name of the serie that will be used to label the abscissa axis.
- Drawable is either TRUE or FALSE if you want to display or not the data serie.

## Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(24,25,26,25,25),"My Serie 1");
$MyData->addPoints(array(10,20,53,45,40),"My Serie 2");

/* Disable the "Drawable" flag for "My Serie 1" */
$MyData->setSerieDrawable("My Serie 1",FALSE);
```

Only "My Serie 2" will be drawn by the charting functions.

# setXAxisUnit - Define the unit to use on the X Axis

This function allows you to set the unit associated to the X Axis. This unit will be displayed when the axis display mode is set to AXIS_FORMAT_UNIT.

## Calling this function

```
setXAxisUnit($Unit);
```

Where :

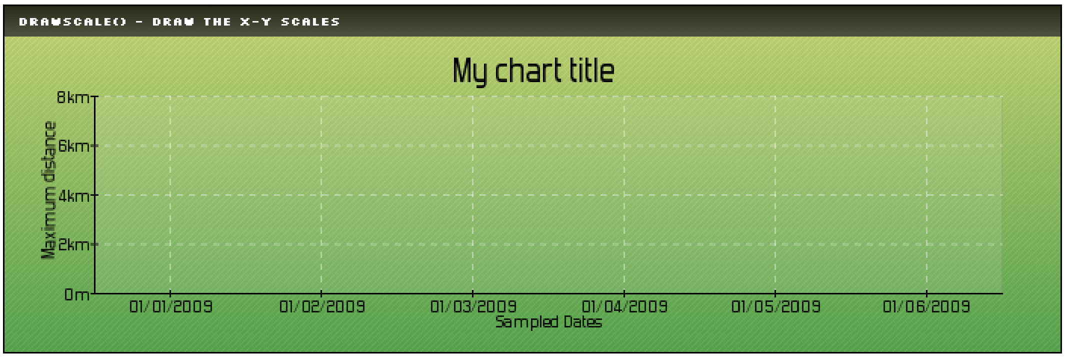
- Unit is the kind of data that will be displayed.

## Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Units appended to the axis values will be "m/s" */
$MyData->setXAxisUnit("m/s");
```

## Sample script #2



Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData = new pData();
$MyData->addPoints(array(1700,2500,7800,4500,3150),"Distance");
$MyData->setAxisName(0,"Maximum distance");
$MyData->setAxisUnit(0,"m");
$MyData->setAxisDisplay(0,AXIS_FORMAT_METRIC);

/* Create the X serie */
$MyData->addPoints(array(1230768000,1233446400,1235865600,1238544000,1241136000,1243814400),"Timestamp");
$MyData->setSerieDescription("Timestamp","Sampled Dates");
$MyData->setAbscissa("Timestamp");
$MyData->setXAxisDisplay(AXIS_FORMAT_DATE);

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(60,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(60,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
$myPicture->drawScale();
```

# setAxisDisplay - Define the way to show values on one Y Axis

This function allows you to set the kind of data to display on the Y Axis. Depending on the type choosen, data will be computed in different ways. Values on this axis are automatically computed by the scaling algorithm.

Information

By default all new series are binded to the default axis with an ID of 0. You can bind a serie to another axis using the setSerieOnAxis function.

## Calling this function

```
setAxisDisplay($AxisID,$Mode,$Format=NULL);
```

Where :

- AxisID is the ID of the axis we want to name.
- Mode is the kind of data that will be displayed.
- Format is the way to display in the associated mode.

Valid **Mode** possible values are :

- AXIS_FORMAT_DEFAULT to display raw data.
- AXIS_FORMAT_TIME to convert to time (HH:MM:SS).
- AXIS_FORMAT_DATE to convert to date (D/M/Y).
- AXIS_FORMAT_METRIC to convert to metric values (1k).
- AXIS_FORMAT_CURRENCY to convert to curencies (1.200€).

Depending on the choosen Mode, you can specify a custom format using the **Format** parameter. Valid formats are :

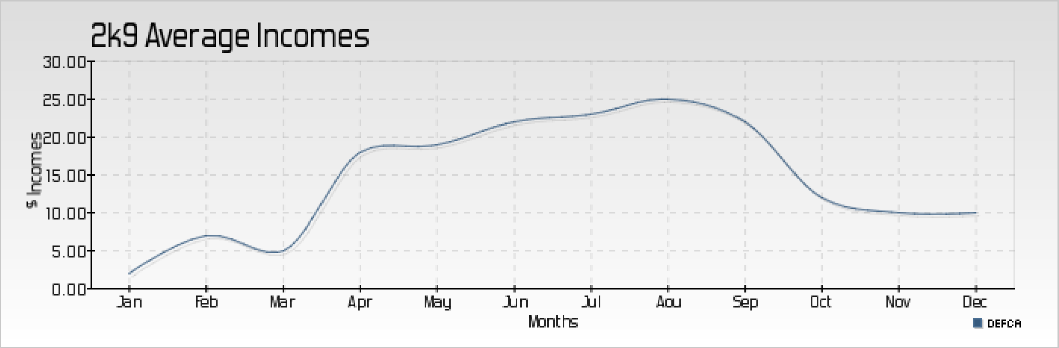
- For AXIS_FORMAT_TIME values you may provide a PHP date format. (like "H:i:s")
- For AXIS_FORMAT_DATE values you may provide a PHP date format. (like "d/m/y")
- For AXIS_FORMAT_CURENCY values you may provide a currency. (like $ or €)

## Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* The default ordinate axis will use metric conversion of the values */
$MyData->setAxisDisplay(0,AXIS_FORMAT_METRIC);

/* The second ordinate axis will use time with a custom display mask */
$MyData->setAxisDisplay(1,AXIS_FORMAT_TIME,"H:i");
```

## Sample script #2



Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData->addPoints(array(2,7,5,18,19,22,23,25,22,12,10,10),"DEFCA");
$MyData->setAxisName(0,"$ Incomes");
$MyData->setAxisDisplay(0,AXIS_FORMAT_CURRENCY);
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun","Jul","Aou","Sep","Oct","Nov","Dec"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");
$MyData->setPalette("Data",array("R"=>55,"G"=>91,"B"=>127));
```

```
/* Define the graph area and do some makeup */
$myPicture = new pImage(700,230,$MyData);

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,array("StartR"=>220,"StartG"=>220,"StartB"=>220,"EndR"=>255,"EndG"=>255,"En

dB"=>255,"Alpha"=>100));

$myPicture->drawRectangle(0,0,699,229,array("R"=>100,"G"=>100,"B"=>100));
 $myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>11));
$myPicture->drawText(60,35,"2k9 Average Incomes",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMLEFT));
$myPicture->setGraphArea(60,40,670,190);
$myPicture->drawFilledRectangle(60,40,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
 $myPicture->drawScale(array("GridR"=>180,"GridG"=>180,"GridB"=>180));

/* Build the chart */
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawSplineChart();
$myPicture->setShadow(FALSE);

/* Write the legend */
$myPicture->drawLegend(643,210,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));
```

# drawAll - Mark all series as drawable

This function will mark all the data series that you've populated as drawable. When you'll call a chart drawing function, all data series will be taken.

Information

By default, when you create a serie by adding some data in it, it is automatically marked as drawable. If a serie is defined as abscissa labels, it can&lsquo;t be marked as drawable.

## Calling this function

```
drawAll();
```

There is no required parameters.

## Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(1,2,3,4),"My Serie 1");
$MyData->addPoints(array(4,3,2,1),"My Serie 2");

/* Mark "My Serie 2" as not drawable */
$MyData->setSerieDrawable("My Serie 2",FALSE);

/* Mark "My Serie 1" and "My Serie 2" as drawable */
$MyData->drawAll();
```

Self explicit no?

# setAbscissa - Define the abscissa axis labels serie

This function allow you to associate a data series to the abscissa labels of you chart. This function can be used in a lot of circumstances like for displaying dates, hours, or pure text labels on the abscissa axis.

Information

When selecting a serie as Abscissa axis labels, this serie is marked as "non drawable"

## Calling this function

```
setAbscissa($Serie);
```

Where :

- Serie is the name of the serie that will be used to label the abscissa axis.

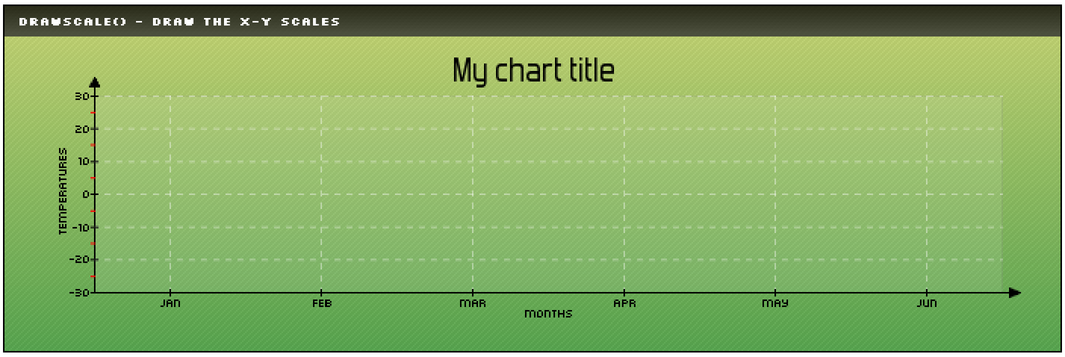
## Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(1,2,3,4),"My Serie 1");
$MyData->addPoints(array("Jan","Feb","Mar","Apr"),"My Serie 2");

/* Use "My Serie 2" as abscissa axis labels */
$MyData->setAbscissa("My Serie 2");
```

Labels "Jan","Feb","Mar","Apr" will be used on the abscissa axis.

## Sample script #2



Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData->addPoints(array(24,-25,26,25,25),"Temperature");
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(60,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(60,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));
```

# setSeriePicture - Associate a picture to a data serie

This function allows you to a link a picture (most of the time an icon) to a data serie. This image will be used in the chart legend.

Information

This function output isn&lsquo;t implemented yet.

## Calling this function

```
setSeriePicture($Serie,$Picture);
```

Where :

- Serie is the name of the serie that will be used to label the abscissa axis.
- Picture is the path to the serie picture.

## Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(24,25,26,25,25),"My Serie 1");

/* Associate the picture "pictures/icon.png" to "My Serie 1" */
$MyData->setSeriePicture("My Serie 1","pictures/icon.png");
```

This will associate the picture "pictures/icon.png" to "My Serie 1"

# setSerieWeight - Draw series with specified weight

This function allows you to draw your series with the specified weight instead of 1px default one.

Information

Currently the weight parameter is only used on standard line charts.

## Calling this function

```
setSerieTicks($Serie,$Weight);
```

Where :

- Serie is the name of the serie that will be used to label the abscissa axis.
- Weight is the weight of the lines. Setting this value to 0 will draw solid lines.

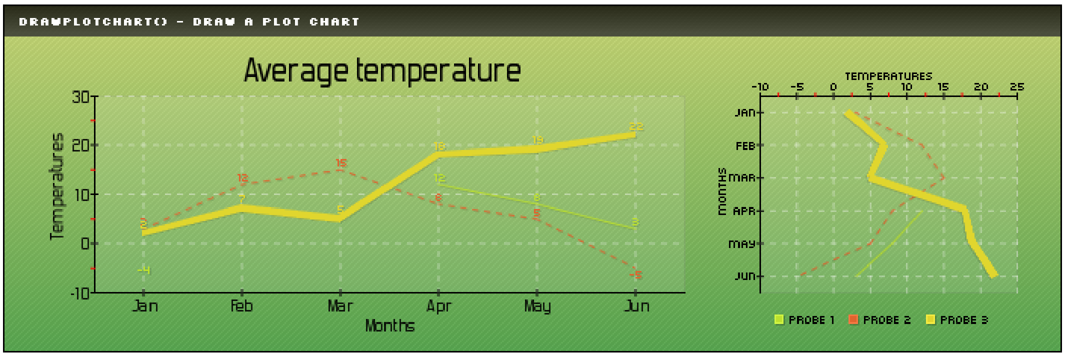
## Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(24,25,26,25,25),"My Serie 1");

/* Set a weight of 4 to "My Serie 1" */
$MyData->setSerieWeight("My Serie 1",4);
```

"My Serie 1" will be draw with a width of 4 px.

## Sample script #2



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(-4,VOID,VOID,12,8,3),"Probe 1");
$MyData->addPoints(array(3,12,15,8,5,-5),"Probe 2");
$MyData->addPoints(array(2,7,5,18,19,22),"Probe 3");
$MyData->setSerieTicks("Probe 2",4);
$MyData->setSerieWeight("Probe 3",2);
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);



$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>80));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));


/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawPlotChart() - draw a plot chart",array("R"=>255,"G"=>255,"B"=>255));

/* Write the chart title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));
$myPicture->drawText(250,55,"Average temperature",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
```

```
/* Draw the scale and the 1st chart */
$myPicture->setGraphArea(60,60,450,190);
$myPicture->drawFilledRectangle(60,60,450,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawLineChart(array("DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO));
$myPicture->setShadow(FALSE);

/* Draw the scale and the 2nd chart */
$myPicture->setGraphArea(500,60,670,190);
$myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>-1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawLineChart();
$myPicture->setShadow(FALSE);

/* Write the chart legend */
$myPicture->drawLegend(510,205,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawLineChart.png");
```

# setSerieTicks - Draw series with dotted lines

This function allows you to draw your series with ticks instead of solid lines.

## Calling this function

```
setSerieTicks($Serie,$Width);
```

Where :

- Serie is the name of the serie that will be used to label the abscissa axis.
- Width is the width of the ticks. Setting this value to 0 will draw solid lines.

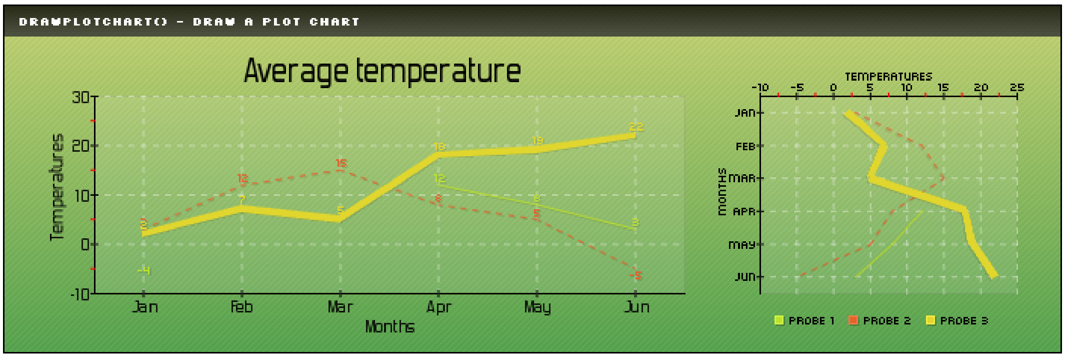
## Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(24,25,26,25,25),"My Serie 1");

/* Use ticks while drawing "My Serie 1" */
$MyData->setSerieTicks("My Serie 1",4);
```

"My Serie 1" will be draw with dotted lines.

## Sample script #2



Code sample

```
/* Build a dataset */
$MyData = new pData();
$MyData->addPoints(array(-4,VOID,VOID,12,8,3),"Probe 1");
$MyData->addPoints(array(3,12,15,8,5,-5),"Probe 2");
$MyData->addPoints(array(2,7,5,18,19,22),"Probe 3");
$MyData->setSerieTicks("Probe 2",4);
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Create the 1st chart*/
$myPicture->setGraphArea(60,60,450,190);
$myPicture->drawFilledRectangle(60,60,450,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawLineChart(array("DisplayValues"=>TRUE,"DisplayColor"=>DISPLAY_AUTO));
$myPicture->setShadow(FALSE);

/* Create the 2nd chart */
$myPicture->setGraphArea(500,60,670,190);
$myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"DrawSubTicks"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
$myPicture->drawLineChart();
$myPicture->setShadow(FALSE);

/* Write the legend*/
$myPicture->drawLegend(510,205,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));
```

# importFromCSV - Load data from a CSV (or similar) file

This function allow you to populate your data series by retrieving data from a CSV (or any kind of characters separated text files).It is possible to set extra parameters with the **$Option** array. To learn more about this please read the Format array guide.

Information

Column numbering is starting from 0.

## Calling this function

```
importFromCSV($FileName,$Option="");
```

Where :

- FileName is the path to the file to load (can be local or remote).
- Option is an array containing some extra parameters.

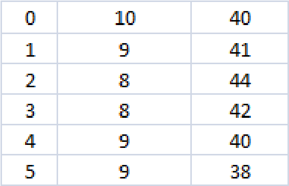
## Customisation array - Extra parameters

It is possible to give extra parameters to the CSV engine :

- The delimiter can be set with Delimiter.
- If your file contains column headers then set GotHeader to TRUE.
- If you want to skip columns, provide them through the SkipColumns array.
- If the CSV does not contains an header, serie names will be based on the DefaultSerieName start string.

## Sample script #1

Assuming we have the following CSV file (delimited by comma) :



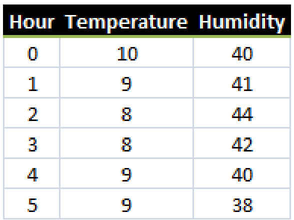
| | | |
|---|---|---|
| 0 | 10 | 40 |
| 1 | 9 | 41 |
| 2 | 8 | 44 |
| 3 | 8 | 42 |
| 4 | 9 | 40 |
| 5 | 9 | 38 |

Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");

/* Create the pData object with some random values*/
$MyData = new pData();

/* Import the data from a CSV file */
$MyData->importFromCSV("resources/dataset.txt");
```

## Sample script #2

Assuming we have the following CSV file (delimited by comma) with an header:



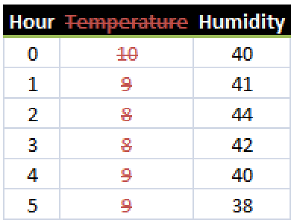
| Hour | Temperature | Humidity |
|---|---|---|
| 0 | 10 | 40 |
| 1 | 9 | 41 |
| 2 | 8 | 44 |
| 3 | 8 | 42 |
| 4 | 9 | 40 |
| 5 | 9 | 38 |

Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");

/* Create the pData object with some random values*/
$MyData = new pData();

/* Import the data from a CSV file */
$MyData->importFromCSV("resources/dataset.txt",array("GotHeader"=>TRUE));
```

## Sample script #3

Assuming we have the following CSV file (delimited by comma) with an header where we want to skip column 1 (Temperature) :



| Hour | ~~Temperature~~ | Humidity |
|---|---|---|
| 0 | ~~10~~ | 40 |
| 1 | ~~9~~ | 41 |
| 2 | ~~8~~ | 44 |
| 3 | ~~8~~ | 42 |
| 4 | ~~9~~ | 40 |
| 5 | ~~9~~ | 38 |

Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");

/* Create the pData object with some random values*/
$MyData = new pData();

/* Import the data from a CSV file */
$MyData->importFromCSV("resources/dataset.txt",array("GotHeader"=>TRUE,"SkipColumns"=>array(1)));
```

# getValues - Extract serie values

This function allow you to extract the data contained in the specified serie in an array format.

## Calling this function

```
getValues($Serie);
```

Where :

- Serie is the name of the serie.

## Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(1,2,3,4),"My Serie 1");

/* This will dump the values contained in "My Serie 1" */
print_r($MyData->getValues("My Serie 1");
```

This will display :

```
Array
Array
(
   [0] => 1
   [1] => 2
   [2] => 3
   [3] => 4
)
```

# getMin - Return the smallest value of a given serie

This function will return you the smallest value contained in the given serie.

## Calling this function

```
getMin($Serie);
```

Where :

- Serie is the name of the serie.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Add a some points to "My serie1" */
$MyData->addPoints(array(1,7,8,2,6,4),"My serie 1");

/* Display TRUE as our pData object now contains data */
echo $MyData->getMin("My serie 1");
```

This will display 1.

## setSerieOnAxis - Bind a serie to one axis

This function allows you to bind a data serie to one ordinate axis. You can create as many axis as you want. If the axis ID you are specifying does not exists it will be created with it's default parameters. You can then name it, place it and specify how to display data on it.

It is important to create different axis to dispatch data series of different natures. You can't scale the same way temperatures (°C) and humidity values (%). Creating two axis, one for each kind of data will help you to display the link between this two entities on the same chart without stretching the scales.

Information

By default all new series are binded to the default axis with an ID of 0. You can bind a serie to another axis using the setSerieOnAxis function.

### Calling this function

```
setSerieOnAxis($SerieID,$AxisID);
```

Where :

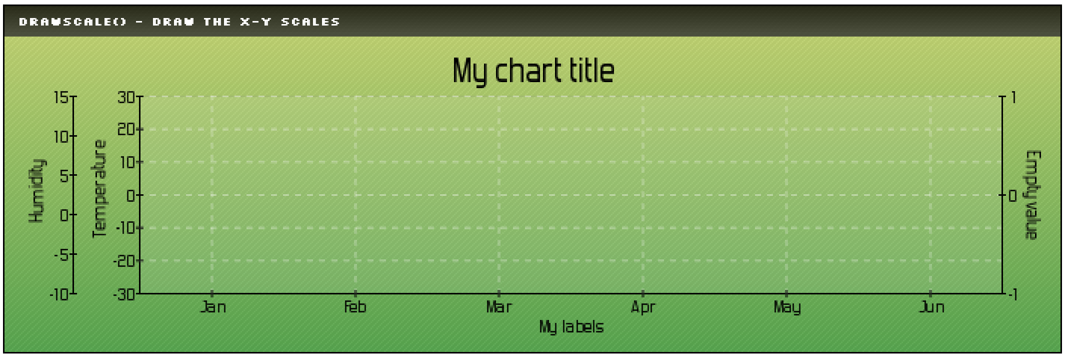
- SerieID is the name of the serie you want to bind to another axis.
- AxisID is the ID of the axis we want to name.

### Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Create some dummy data */
$myData->addPoints(array(10,11,14,13),"Temperature");
$myData->addPoints(array(70,70,69,68),"Humidity");

/* Dispatch the two data series on dedicated axis */
$MyData->setSerieOnAxis("Temperature",0); // This one isn't needed as this is the default serie
$MyData->setSerieOnAxis("Humidity",1);
```

### Sample script #2



Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData->addPoints(array(24,-25,26,25,25),"Temperature");
$MyData->addPoints(array(1,2,VOID,9,10),"Humidity 1");
$MyData->addPoints(array(1,VOID,7,-9,0),"Humidity 2");
$MyData->addPoints(array(-1,-1,-1,-1,-1),"Humidity 3");
$MyData->addPoints(array(0,0,0,0,0),"Vide");
$MyData->setSerieOnAxis("Temperature",0);
$MyData->setSerieOnAxis("Humidity 1",1);
$MyData->setSerieOnAxis("Humidity 2",1);
$MyData->setSerieOnAxis("Humidity 3",1);
$MyData->setSerieOnAxis("Vide",2);
$MyData->setAxisPosition(2,AXIS_POSITION_RIGHT);
$MyData->setAxisName(0,"Temperature");
$MyData->setAxisName(1,"Humidity");
$MyData->setAxisName(2,"Empty value");

/* Bind a data serie to the X axis */
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","My labels");
$MyData->setAbscissa("Labels");

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(90,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(90,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
$myPicture->drawScale(array("DrawYLines"=>array(0)));
```

# setAxisName - Define the name of one Y Axis

This function allows you to set the name of one Y Axis. This name will be displayed when drawing the chart area. Values on this axis are automatically computed by the scaling algorithm.

Information

By default all new series are binded to the default axis with an ID of 0. You can bind a serie to another axis using the setSerieOnAxis function.

## Calling this function

```
setAxisName($AxisID,$Name);
```

Where :

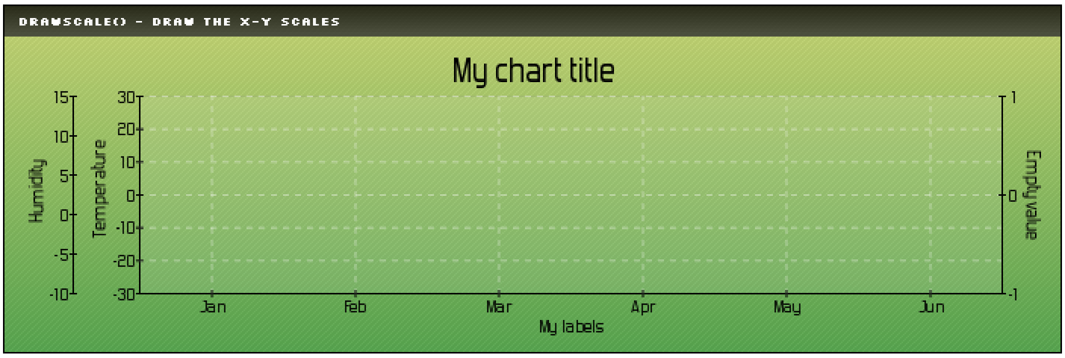
- AxisID is the ID of the axis we want to name.
- Name is the name that will be put on the axis.

### Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* The default axis will be named "Samples" */
$MyData->setAxisName(0,"Samples");
```

### Sample script #2



Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData->addPoints(array(24,-25,26,25,25),"Temperature");
$MyData->addPoints(array(1,2,VOID,9,10),"Humidity 1");
$MyData->addPoints(array(1,VOID,7,-9,0),"Humidity 2");
$MyData->addPoints(array(-1,-1,-1,-1,-1),"Humidity 3");
$MyData->addPoints(array(0,0,0,0,0),"Vide");
$MyData->setSerieOnAxis("Temperature",0);
$MyData->setSerieOnAxis("Humidity 1",1);
$MyData->setSerieOnAxis("Humidity 2",1);
$MyData->setSerieOnAxis("Humidity 3",1);
$MyData->setSerieOnAxis("Vide",2);
$MyData->setAxisPosition(2,AXIS_POSITION_RIGHT);
$MyData->setAxisName(0,"Temperature");
$MyData->setAxisName(1,"Humidity");
$MyData->setAxisName(2,"Empty value");

/* Bind a data serie to the X axis */
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","My labels");
$MyData->setAbscissa("Labels");

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(90,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(90,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
$myPicture->drawScale(array("DrawYLines"=>array(0)));
```

## addRandomValues - Create a randomized serie

This function allow you to populate your data series with random values. You can manage multiple data series in a single pData object. It is possible to set extra parameters with the **$Option** array. To learn more about this please read the Format array guide.

## Calling this function

```
addRandomValues($SerieName="Serie1",$Option="");
```

Where :

- SerieName is the name of the serie where you want to add random points.
- Option is an array containing some extra parameters.

If **SerieName** isn't provided, the points will be added to *Serie1*.

## Customisation array - Extra parameters

It is possible to give extra parameters to the randomizer engine. If nothing is provided, 20 integers between 0 and 100 will be added to the selected serie.

- The number of values to add can be set with Values.
- The min value can be set with Min.
- The max value can be set with Max.
- If you want float values, you can set withFloat to TRUE.

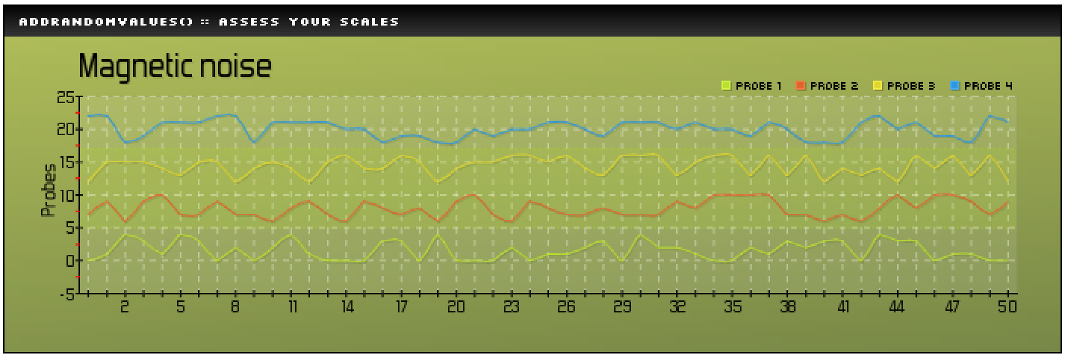
## Sample script #1

Code sample

```
/* Add a 20 random integers between 0-100 in Serie1 */
$myData->addRandomValues();

/* Add a 10 random integers between 0-100 in Serie2 */
$randomOptions = array("Values"=>10);
$myData->addRandomValues("Serie2",$randomOptions);

/* Add a 100 random floats between 10-20 in Serie3 */
$randomOptions = array("Values"=>100,"Min"=>10,"Max"=>20,"withFloat"=>TRUE);
$myData->addRandomValues("Serie3",$randomOptions);
```

This will populate Serie1, Serie2, Serie3 with random values.

## Sample script #2



Code sample

```
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pImage.class.php");

$MyData = new pData();
$MyData->addRandomValues("Probe 1",array("Values"=>50,"Min"=>0,"Max"=>4));
$MyData->addRandomValues("Probe 2",array("Values"=>50,"Min"=>6,"Max"=>10));
$MyData->addRandomValues("Probe 3",array("Values"=>50,"Min"=>12,"Max"=>16));
$MyData->addRandomValues("Probe 4",array("Values"=>50,"Min"=>18,"Max"=>22));
$MyData->setAxisName(0,"Probes");
```

```
 $myPicture = new pImage(700,230,$MyData);

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,array("StartR"=>180,"StartG"=>193,"StartB"=>91,"EndR"=>120,"EndG"=>137,"EndB"=>72,"Alpha"=>100));

$myPicture->drawGradientArea(0,0,700,230,DIRECTION_HORIZONTAL,array("StartR"=>180,"StartG"=>193,"StartB"=>91,"EndR"=>120,"EndG"=>137,"EndB"=>72,"Alpha"=>20));

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>100));
 $myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

 $myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
 $myPicture->drawText(10,13,"addRandomValues() :: assess your scales",array("R"=>255,"G"=>255,"B"=>255));

 $myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>11));
 $myPicture->setGraphArea(50,60,670,190);
 $myPicture->drawFilledRectangle(50,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
 $myPicture->drawScale(array("LabelSkip"=>3,"DrawSubTicks"=>TRUE));

 $myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));
 $myPicture->drawText(50,52,"Magnetic noise",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMLEFT));

 $myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
 $myPicture->drawSplineChart();
 $myPicture->setShadow(FALSE);

 $myPicture->drawLegend(475,50,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

 $myPicture->Render("addrandomvalues.png");
```

# removeSerie - Remove a data serie

This function allow you to remove a data serie from your pData object.

## Calling this function

```
removeSerie($Serie);
```

Where :

- Serie is the name of the serie.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(1,2,3,4),"My Serie 1");
$MyData->addPoints(array(4,3,2,1),"My Serie 2");

/* Remove "My Serie 1" as we don't needed it anymore */
$MyData->removeSerie("My Serie 1");
```

This will remove a data serie.

## containsData - Check data consistency

You can use this function to validate that you have valid data to be drawn. This function is mainly used internally to check data consistency. You probably will not need to use it.

### Calling this function

```
containsData();
```

There is no required parameters.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Display FALSE as our pData object is empty */
echo $MyData->containsData();

/* Add a single point to Serie1 */
$MyData->addPoints(7);

/* Display TRUE as our pData object now contains data */
echo $MyData ->containsData();
```

This function can help you to validate your dataset.

## setScatterSerie - Create a scatter serie

This function allows you to create a scatter serie using the 1st parameter as X coordinates and 2nd as Y ones. By default a scatter serie with an ID of 0 will be created, if you want to create another scatter serie, specify its unique ID with the ID parameter.

### Calling this function

```
setScatterSerie($SerieX,$SerieY,$ID=0)
```

Where :

- SerieX is the name of the serie that will be used for X axis positions.
- SerieY is the name of the serie that will be used for Y axis positions.
- ID is an optional parameter to define the scatter serie unique ID.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Create the two data series that will be used for the scatter coordinates */
$MyData->addPoints(array(1,2,3,4),"My Serie X");
$MyData->addPoints(array(4,3,2,1),"My Serie Y");

/* Associate the two series to create a scatter serie */
$MyData->setScatterSerie("My Serie X","My Serie Y");
```

This will create a scatter serie with the following points : (1,4), (2,3), (3,2), (4,1).

## setScatterSerieDescription - Define the description of a scatter serie

This function allows you to define the description of a scatter serie. This value will be written in the legend box.

### Calling this function

```
setScatterSerieDescription($ID,$Description)
```

Where :

- ID is the unique ID of the scatter serie to modify.
- Description is the description of the scatter serie.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Create the two data series that will be used for the scatter coordinates */
$MyData->addPoints(array(1,2,3,4),"My Serie X");
$MyData->addPoints(array(4,3,2,1),"My Serie Y");

/* Associate the two series to create a scatter serie */
$MyData->setScatterSerie("My Serie X","My Serie Y");

/* Re-define the scatter serie description */
$MyData->setScatterSerieDescription(0,"My scatter serie");
```

Scatter serie 0 will have the description set to "My scatter serie".

## setScatterSeriePicture - Define the associated picture of a scatter serie

This function allows you to define the associted picture of a scatter serie. This picture will be used in the legend box and by the plot charting functions.

### Calling this function

```
setScatterSeriePicture($ID,$Picture=NULL)
```

Where :

- ID is the unique ID of the scatter serie to modify.
- Picture is the path to the picture file (can be PNG, JPG or GIF).

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Create the two data series that will be used for the scatter coordinates */
$MyData->addPoints(array(1,2,3,4),"My Serie X");
$MyData->addPoints(array(4,3,2,1),"My Serie Y");

/* Associate the two series to create a scatter serie */
$MyData->setScatterSerie("My Serie X","My Serie Y");

/* Re-define the scatter serie picture */
$MyData->setScatterSeriePicture(0,"icons/world.png");
```

Scatter serie 0 will have the description set to "My scatter serie".

## setScatterSerieDrawable - Define if a scatter serie will be drawn

This function allows you to define if a scatter serie will be drawn when calling the legend or any charting function.

### Calling this function

```
setScatterSerieDrawable($ID ,$Drawable=TRUE)
```

Where :

- ID is the unique ID of the scatter serie to modify.
- Drawable is a boolean value.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Create the two data series that will be used for the scatter coordinates */
$MyData->addPoints(array(1,2,3,4),"My Serie X");
$MyData->addPoints(array(4,3,2,1),"My Serie Y");

/* Associate the two series to create a scatter serie */
$MyData->setScatterSerie("My Serie X","My Serie Y");

/* Mark the scatter serie as non-drawable */
$MyData->setScatterSerieDrawable(0,FALSE);
```

Scatter serie 0 will not be drawn by the charting functions.

## setScatterSerieTicks - Define the ticks parameter of a scatter serie

This function allows you to define the ticks width parameter of a scatter serie. This will be reflected in the legend box and by the line and spline charting functions.

### Calling this function

```
setScatterSerieTicks($ID,$Width=0)
```

Where :

- ID is the unique ID of the scatter serie to modify.
- Width is the width of the ticks.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Create the two data series that will be used for the scatter coordinates */
$MyData->addPoints(array(1,2,3,4),"My Serie X");
$MyData->addPoints(array(4,3,2,1),"My Serie Y");

/* Associate the two series to create a scatter serie */
$MyData->setScatterSerie("My Serie X","My Serie Y");

/* Draw the scatter serie 0 with ticks */
$MyData->setScatterSerieTicks(0,4);
```

Scatter serie 0 will be drawn with ticks

## setScatterSerieWeight - Define the weight parameter of a scatter serie

This function allows you to define the weight parameter of a scatter serie. This will be reflected in the legend box and by the line and spline charting functions.

### Calling this function

```
setScatterSerieWeight($ID,$Weight=0)
```

Where :

- ID is the unique ID of the scatter serie to modify.
- Weight is the weight of the ticks.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Create the two data series that will be used for the scatter coordinates */
$MyData->addPoints(array(1,2,3,4),"My Serie X");
$MyData->addPoints(array(4,3,2,1),"My Serie Y");

/* Associate the two series to create a scatter serie */
$MyData->setScatterSerie("My Serie X","My Serie Y");

/* Draw the scatter serie 0 with a weight of 2px */
$MyData->setScatterSerieTicks(0,2);
```

Scatter serie 0 will be drawn with a weight of 2px.

## setScatterSerieColor - Define the color of a scatter serie

This function allows you to redefine the color of the given scatter serie. It is possible to set extra parameters with the **$Format** array. To learn more about this please read the Format array guide.

### Calling this function

```
setScatterSerieColor($ID,$Format)
```

Where :

- ID is the unique ID of the scatter serie to modify.
- Format is an optional parameter array containing the new color.

### Customisation array - Extra parameters

It is possible to give extra parameters :

- The color can be set with  R, G, B.
- The alpha transparency factor can be set with Alpha.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Create the two data series that will be used for the scatter coordinates */
$MyData->addPoints(array(1,2,3,4),"My Serie X");
$MyData->addPoints(array(4,3,2,1),"My Serie Y");

/* Associate the two series to create a scatter serie */
$MyData->setScatterSerie("My Serie X","My Serie Y");

/* Re-define the scatter serie color */
$MyData->setScatterSerieColor(0,array("R"=>255,"G"=>255,"B"=>255));
```

Scatter serie 0 will be drawn in white (255,255,255).

## addPoints - Build your data series

This function allow you to populate your data series by either providing single points or an array of points. You can manage multiple data series in a single pData object.

Information

Missing points can be replaced by the VOID keyword.

If you add points to a serie that isn‘t existing yet, it will be binded to the axis #0, to bind it to another axis, use the setSerieOnAxis function. You can create an unlimited number of axis in pChart 2.0.

### Calling this function

```
addPoints($Values,$SerieName="Serie1");
```

Where :

- Value is either one value or an array of values.
- SerieName is the name of the serie were you want to add the point(s).

If **SerieName** isn‘t provided, the points will be added to *Serie1*.

### Sample script

Code sample

```
/* Add a single point to Serie1 */
$myData->addPoints(7);

/* Add an array of points to Serie2 */
$myData->addPoints(array(2,3,4,VOID,5),"Serie2");
```

This will create two data series, 1[SUP]st[/SUP] one will take the default name of **Serie1** and will contains only one data point. The second one explicitely called **Serie2** will contains 5 points including a missing value represented by the *VOID* keyword.

## reverseSerie - Reverse data serie values

This function allow you to reverse the content of the specified data serie from your pData object.

### Calling this function

```
reverseSerie($Serie);
```

Where :

- Serie is the name of the serie.

### Sample script

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(1,2,3,4),"My Serie 1");

/* Reverse "My Serie 1" contents */
$MyData->reverseSerie("My Serie 1");
```

This will reverse the content of "My Serie 1". The data array will now contains :

```
Array
(
    [0] => 4
    [1] => 3
    [2] => 2
    [3] => 1
)
```

## setSerieDescription- Add a description to one serie

You can use this function to set a description to one serie. This attribute is important when you want to add a legend to you chart, this way instead of writting the serie name that may be obscur to the user, you can use an user friendly description

Information

When creating a serie the description is set to the serie name.

### Calling this function

```
setSerieDescription($Serie,$Description);
```

Where :

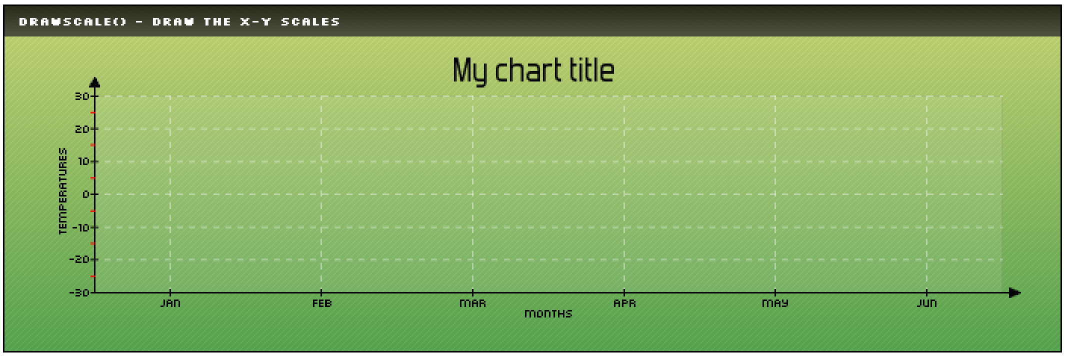
- Serie is the name of the serie that will be used to label the abscissa axis.
- Description is the description associated to the serie.

### Sample script #1

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate some data */
$MyData->addPoints(array(24,25,26,25,25),"My Serie 1");

/* Set the description of "My Serie 1" */
$MyData->setSerieDescription("My Serie 1","Temperature");
```

## Sample script #2



Code sample

```
$MyData = new pData();

/* Prepare some nice data & axis config */
$MyData->addPoints(array(24,-25,26,25,25),"Temperature");
$MyData->setAxisName(0,"Temperatures");
$MyData->addPoints(array("Jan","Feb","Mar","Apr","May","Jun"),"Labels");
$MyData->setSerieDescription("Labels","Months");
$MyData->setAbscissa("Labels");

/* Define the graph area and do some makeup */
$myPicture->setGraphArea(60,60,660,190);
$myPicture->drawText(350,55,"My chart title",array("FontSize"=>20,"Align"=>TEXT_ALIGN_BOTTOMMIDDLE));
$myPicture->drawFilledRectangle(60,60,660,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));

/* Compute and draw the scale */
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE));
```

# Barcode 39 functions

# pBarcode39 - Draw a code 39 barcode

This function allows you to embed a code 39 barcode on your pictures.

Code 39 (also known as "USS Code 39", "Code 3/9", "Code 3 of 9", "USD-3", "Alpha39", "Type 39") is a barcode symbology that can encode uppercase letters (A through Z), digits (0 through 9) and a handful of special characters like the $ sign. The barcode itself does not contain a check digit (in contrast to Code 128), but it can be considered self-checking by some, on the grounds that a single erroneously interpreted bar cannot generate another valid character. Possibly the most serious drawback of Code 39 is its low data density: It requires more space to encode data in Code 39 than, for example, in Code 128. This means that very small goods cannot be labeled with a Code 39 based barcode. However, Code 39 is still widely used and can be decoded with virtually any barcode reader.

The Code 39 (also known as 3 of 9 bar code) is a variable length, discrete, alphanumeric bar code. Its character set contains 43 meaningful characters: 0 - 9, A-Z, -, ., $, /, +, %, and space. Each character is composed of nine elements: five bars and four spaces. Three of the nine elements are wide (binary value 1), and six elements are narrow (binary value 0). An additional common character (*) is used for both start and stop delimiters.

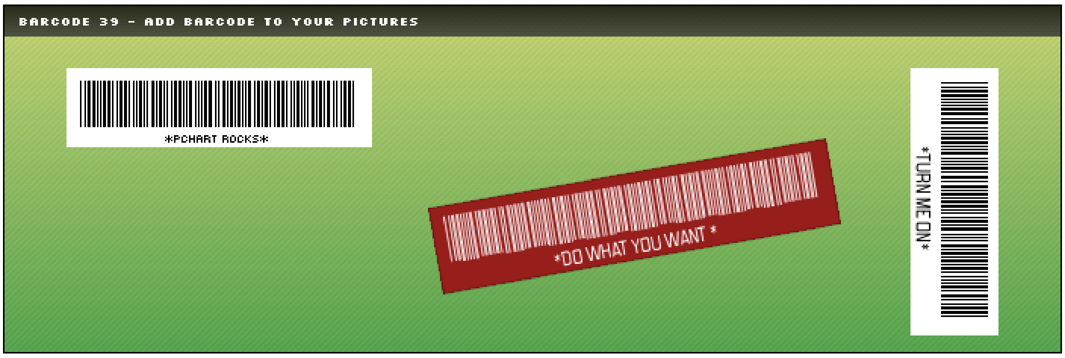
## Calling this function

```
pBarcode39($BasePath="",$EnableMOD43=FALSE);
```

Where :

- $BasePath specify the top level path where the data folder stands.
- $EnableMOD43 specify if we‘ll have a modulo 43 checksum to the barcode (default is FALSE).

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pDraw.class.php");
include("../class/pBarcode39.class.php");
include("../class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(700,230);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,

"EndB"=>50,"Alpha"=>80));


/* Draw the picture border */
```

```
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"Barcode 39 - Add barcode to your pictures",array("R"=>255,"G"=>255,"B"=>255));

/* Create the barcode 39 object */
$Barcode = new pBarcode39("../");

/* Draw a simple barcode */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$Settings = array("ShowLegend"=>TRUE,"DrawArea"=>TRUE);
$Barcode->draw($myPicture,"pChart Rocks!",50,50,$Settings);

/* Draw a rotated barcode */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>12));
$Settings = array("ShowLegend"=>TRUE,"DrawArea"=>TRUE,"Angle"=>90);
$Barcode->draw($myPicture,"Turn me on",650,50,$Settings);

/* Draw a rotated barcode */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>12));

$Settings = array("R"=>255,"G"=>255,"B"=>255,"AreaR"=>150,"AreaG"=>30,"AreaB"=>27,"ShowLegend"=>TRUE,

"DrawArea"=>TRUE,"Angle"=>350,"AreaBorderR"=>70,"AreaBorderG"=>20,"AreaBorderB"=>20);

$Barcode->draw($myPicture,"Do what you want !",290,140,$Settings);

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.barcode39.png");
```

This will draw three barcodes.

# draw - Draw a code 39 barcode

This function allows you to draw a code 39 barcode on your pictures. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
draw($pChartObject,$TextString,$X,$Y,$Format="");
```

Where :

- $pChartObject is the reference to the pChart object where the barcode will be drawn.
- $TextString is the text to be encoded.
- X,Y are the coordinate of the top left point.
- Format is an array containing the drawing parameters of the pixel.

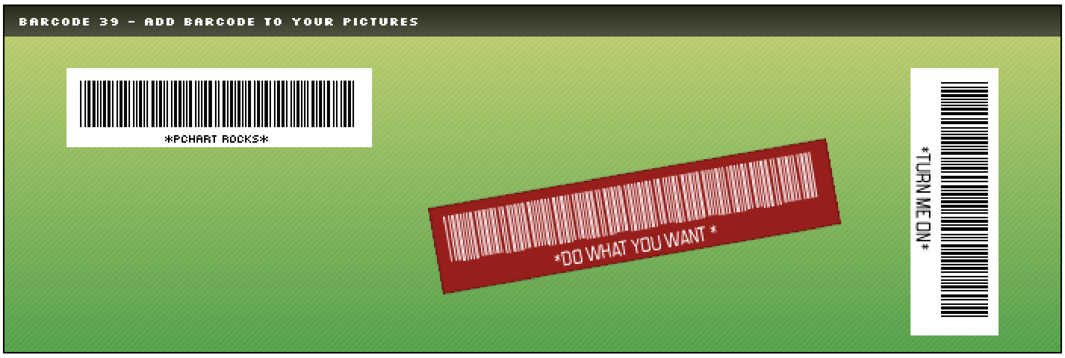
## Customisation array - Tune up your barcode!

It is possible to customize the barcode rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the barcode will be drawn black with no transparency, no border and no subtitle.

- The barcode color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.
- You can define the barcode height with Height. (default is 30)
- You can define the barcode angle with Angle.
- If you want to display the legend in clear text, set ShowLegend to TRUE.
- You can define the legend offset with Angle. (default is 5)
- You can draw a surrounding area by setting DrawArea to TRUE.
- The area filling color can be set with R, G, B.

- The area border color can be set with AreaBorderR, AreaBorderG, AreaBorderB.
- You can add the MOD43 security character setting EnableMOD43 to TRUE.

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pDraw.class.php");
include("../class/pBarcode39.class.php");
include("../class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(700,230);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,

"EndB"=>50,"Alpha"=>80));


/* Draw the picture border */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"Barcode 39 - Add barcode to your pictures",array("R"=>255,"G"=>255,"B"=>255));

/* Create the barcode 39 object */
$Barcode = new pBarcode39("../");

/* Draw a simple barcode */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$Settings = array("ShowLegend"=>TRUE,"DrawArea"=>TRUE);
$Barcode->draw($myPicture,"pChart Rocks!",50,50,$Settings);

/* Draw a rotated barcode */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>12));
$Settings = array("ShowLegend"=>TRUE,"DrawArea"=>TRUE,"Angle"=>90);
$Barcode->draw($myPicture,"Turn me on",650,50,$Settings);

/* Draw a rotated barcode */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>12));

$Settings = array("R"=>255,"G"=>255,"B"=>255,"AreaR"=>150,"AreaG"=>30,"AreaB"=>27,"ShowLegend"=>TRUE,
"DrawArea"=>TRUE,"Angle"=>350,"AreaBorderR"=>70,"AreaBorderG"=>20,"AreaBorderB"=>20);
$Barcode->draw($myPicture,"Do what you want !",290,140,$Settings);

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.barcode39.png");
```

This will draw three barcodes.

# getSize - Return the projected size of a code 39 barcode

This function allows you to estimate the target size of a code 39 barcode. You‘ll have to give it the same **$Format** array than the one you‘ll use when calling the draw method in order to have an accurate projection of the barcode size.

This function is particulary useful when you want to create a picture containing only a barcode (to embed it in a more complex document like PDF file).

[ERR]If you plan to display the barcode legend in plain text (ShowLegend parameter), you‘ll need to specify the font height manually (FontSize parameter) as when you‘ll call this function the pChart object will not be instantiated yet.[/ERR]

## Calling this function

```
getSize($TextString,$Format="")
```

Where :

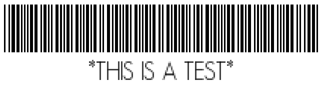
- $TextString is the text to be encoded.
- Format is an array containing the drawing parameters of the pixel.

## Customisation array - Tune up your barcode!

It is possible to customize the barcode rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the barcode will be drawn black with no transparency, no border and no subtitle.

- You can define the barcode height with Height. (default is 30)
- You can define the barcode angle with Angle.
- If you want to display the legend in clear text, set ShowLegend to TRUE.
- You can define the legend offset with LegendOffset. (default is 5)
- You can draw a surrounding area by setting DrawArea to TRUE.
- You can specify the font height to use with FontSize.

## Sample script



Code sample

```
 /* pChart library inclusions */
include("class/pDraw.class.php");
include("class/pBarcode39.class.php");
include("class/pImage.class.php");

/* Create the barcode 39 object */
$Barcode = new pBarcode39();

/* String to be written on the barcode */
$String = "This is a test";

/* Retrieve the barcode projected size */
$Settings = array("ShowLegend"=>TRUE,"DrawArea"=>TRUE);
$Size = $Barcode->getSize($String,$Settings);
```

```
/* Create the pChart object */
$myPicture = new pImage($Size["Width"],$Size["Height"]);

/* Render the barcode */
$Barcode->draw($myPicture,$String,10,10,$Settings);

/* Render the picture */
$myPicture->Render("singlebarcode39.png");
```

This will draw a single barcode.

# Barcode 128 functions

# pBarcode128 - Draw a code 128 barcode

This function allows you to embed a code 128 barcode on your pictures.

Code 128 is a very high density alphanumeric bar code. The symbol can be as long as necessary to store the encoded data. It is designed to encode all 128 ASCII characters, and will use the least amount of space for data of 6 characters or more of any 1-D symbology. Each data character encoded in a Code 128 symbol is made up of 11 black or white modules. The stop character, however, is made up of 13 modules. Three bars and three spaces are formed out of these 11 modules. Bar and spaces can vary between 1 and 4 modules wide.

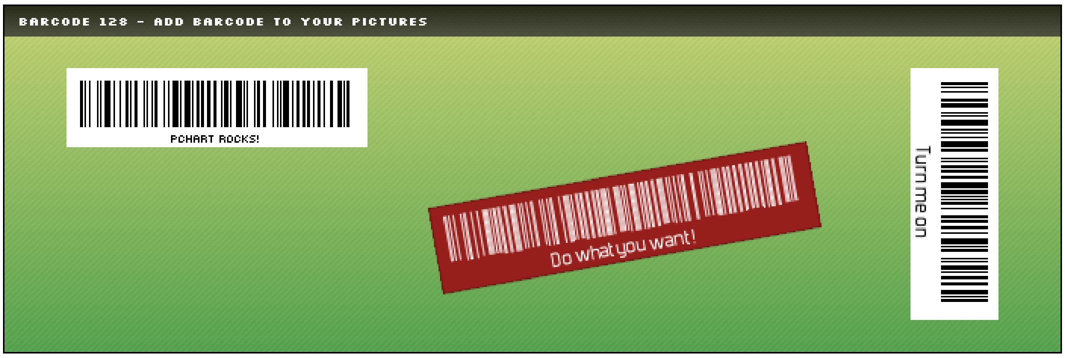
The symbol includes a quiet zone (10 x-dimensions), a start character, the encoded data, a check character, the stop character, and a trailing quiet zone (10 x-dimensions). For optimum hand-scanning with a contact reader, the quiet zone should be at least 0.25 inches.

## Calling this function

```
pBarcode128($BasePath="");
```

- $BasePath specify the top level path where the data folder stands.

## Sample script



Code sample

```
/* pChart library inclusions */
include("class/pDraw.class.php");
include("class/pBarcode128.class.php");
include("class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(700,230);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);



$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>80));

/* Draw the top bar */

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,
"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>100));

$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));
$myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"Barcode 128 - Add barcode to your pictures",array("R"=>255,"G"=>255,"B"=>255));

/* Create the barcode 128 object */
$Barcode = new pBarcode128();

/* Draw a simple barcode */
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$Settings = array("ShowLegend"=>TRUE,"DrawArea"=>TRUE);
$Barcode->draw($myPicture,"pChart Rocks!",50,50,$Settings);

/* Draw a rotated barcode */
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>12));
$Settings = array("ShowLegend"=>TRUE,"DrawArea"=>TRUE,"Angle"=>90);
$Barcode->draw($myPicture,"Turn me on",650,50,$Settings);

/* Draw a rotated barcode */
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>12));

$Settings = array("R"=>255,"G"=>255,"B"=>255,"AreaR"=>150,"AreaG"=>30,"AreaB"=>27,"ShowLegend"=>TRUE,"DrawArea"=>TRUE,

"Angle"=>350,"AreaBorderR"=>70,"AreaBorderG"=>20,"AreaBorderB"=>20);

$Barcode->draw($myPicture,"Do what you want !",290,140,$Settings);

/* Render the picture */
$myPicture->Render("drawbarcode.png");
```

This will draw three barcodes.

# draw - Draw a code 128 barcode

This function allows you to draw a code 128 barcode on your pictures. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide.

Code 128 is a very high-density barcode symbology. It is used for alphanumeric or numeric-only barcodes. It can encode all 128 characters of ASCII and is widely used in the industry to track packages.

## Calling this function

```
encode128($pChartObject,$TextString,$X,$Y,$Format="");
```

Where :

- $pChartObject is the reference to the pChart object where the barcode will be drawn.
- $TextString is the text to be encoded.
- X,Y are the coordinate of the top left point.
- Format is an array containing the drawing parameters of the pixel.

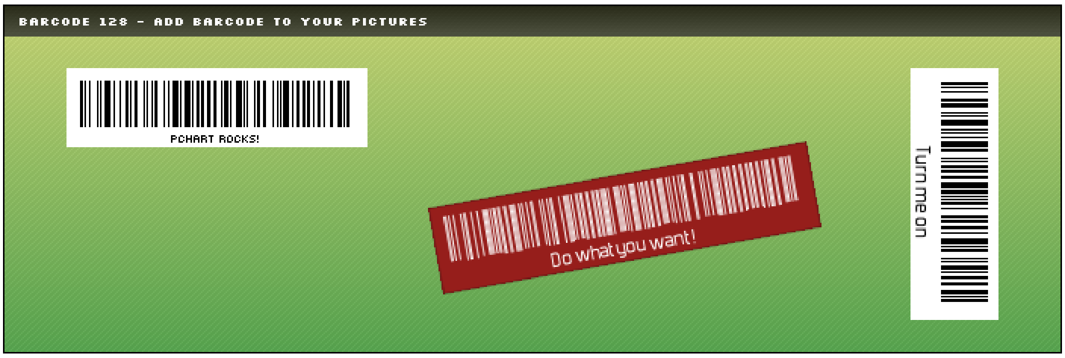
## Customisation array - Tune up your barcode!

It is possible to customize the barcode rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the barcode will be drawn black with no transparency, no border and no subtitle.

- The barcode color can be set with R, G, B.
- The alpha transparency factor can be set with Alpha.
- You can define the barcode height with Height. (default is 30)
- You can define the barcode angle with Angle.
- If you want to display the legend in clear text, set ShowLegend to TRUE.
- You can define the legend offset with Angle. (default is 5)

- You can draw a surrounding area by setting DrawArea to TRUE.
- The area filling color can be set with R, G, B.
- The area border color can be set with AreaBorderR, AreaBorderG, AreaBorderB.

## Sample script



Code sample

```
/* pChart library inclusions */
include("class/pDraw.class.php");
include("class/pBarcode128.class.php");
include("class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(700,230);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>80));

/* Draw the top bar */
$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,
"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>100));
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));
$myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"Barcode 128 - Add barcode to your pictures",array("R"=>255,"G"=>255,"B"=>255));

/* Create the barcode 128 object */
$Barcode = new pBarcode128();

/* Draw a simple barcode */
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$Settings = array("ShowLegend"=>TRUE,"DrawArea"=>TRUE);
$Barcode->draw($myPicture,"pChart Rocks!",50,50,$Settings);

/* Draw a rotated barcode */
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>12));
```

```
 $Settings = array("ShowLegend"=>TRUE,"DrawArea"=>TRUE,"Angle"=>90);
 $Barcode->draw($myPicture,"Turn me on",650,50,$Settings);

 /* Draw a rotated barcode */
 $myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>12));
 $Settings = array("R"=>255,"G"=>255,"B"=>255,"AreaR"=>150,"AreaG"=>30,"AreaB"=>27,"ShowLegend"=>TRUE,

"DrawArea"=>TRUE,"Angle"=>350,"AreaBorderR"=>70,"AreaBorderG"=>20,"AreaBorderB"=>20);
 $Barcode->draw($myPicture,"Do what you want !",290,140,$Settings);

 /* Render the picture */
 $myPicture->Render("drawbarcode.png");
```

This will draw three barcodes.

# getSize - Return the projected size of a code 128 barcode

This function allows you to estimate the target size of a code 128 barcode. You‘ll have to give it the same **$Format** array than the one you‘ll use when calling the draw method in order to have an accurate projection of the barcode size.

This function is particulary useful when you want to create a picture containing only a barcode (to embed it in a more complex document like PDF file).

[ERR]If you plan to display the barcode legend in plain text (ShowLegend parameter), you‘ll need to specify the font height manually (FontSize parameter) as when you‘ll call this function the pChart object will not be instantiated yet.[/ERR]

## Calling this function

```
getSize($TextString,$Format="")
```

Where :

- $TextString is the text to be encoded.
- Format is an array containing the drawing parameters of the pixel.

## Customisation array - Tune up your barcode!

It is possible to customize the barcode rendering by playing with this array. Providing a detailled configuration is not mandatory, by default the barcode will be drawn black with no transparency, no border and no subtitle.

- You can define the barcode height with Height. (default is 30)
- You can define the barcode angle with Angle.
- If you want to display the legend in clear text, set ShowLegend to TRUE.
- You can define the legend offset with LegendOffset. (default is 5)
- You can draw a surrounding area by setting DrawArea to TRUE.
- You can specify the font height to use with FontSize.

## Sample script

Code sample

```
  /* pChart library inclusions */
 include("class/pDraw.class.php");
 include("class/pBarcode128.class.php");
 include("class/pImage.class.php");

 /* Create the barcode 128 object */
 $Barcode = new pBarcode128();
```

```
/* String to be written on the barcode */
$String = "This is a test";

/* Retrieve the barcode projected size */
$Settings = array("ShowLegend"=>TRUE,"DrawArea"=>TRUE);
$Size = $Barcode->getSize($String,$Settings);

/* Create the pChart object */
$myPicture = new pImage($Size["Width"],$Size["Height"]);

/* Render the barcode */
$Barcode->draw($myPicture,$String,10,10,$Settings);

/* Render the picture */
$myPicture->Render("singlebarcode128.png");
```

This will draw a single barcode.

# Spring charts functions

# pSpring - Draw spring graphs

This function allows you to create Spring Graphs.

Force-based or force-directed algorithms are a class of algorithms for drawing graphs in an aesthetically pleasing way. Their purpose is to position the nodes of a graph in two dimensional or three dimensional space (not supported yet) so that all the edges are of more or less equal length and there are as few crossing edges as possible.

The force-directed algorithms achieve this by assigning forces amongst the set of edges and the set of nodes; the most straightforward method is to assign forces as if the edges were springs (see Hooke's law) and the nodes were electrically charged particles (see Coulomb's law). The entire graph is then simulated as if it were a physical system. The forces are applied to the nodes, pulling them closer together or pushing them further apart. This is repeated iteratively until the system comes to an equilibrium state; i.e., their relative positions do not change anymore from one iteration to the next. At that moment, the graph is drawn. The physical interpretation of this equilibrium state is that all the forces are in mechanical equilibrium.

Be careful

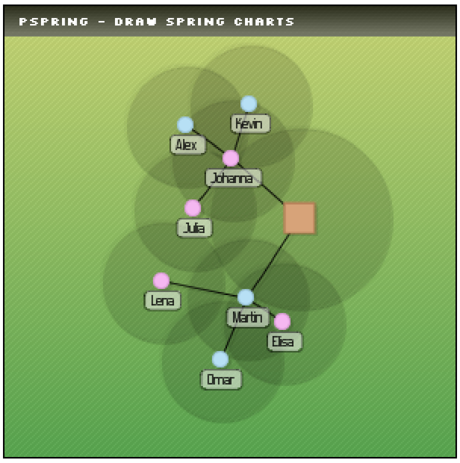
This function will create a different layout everytime it will be called because of the initial placement algorithm that use a random generator.

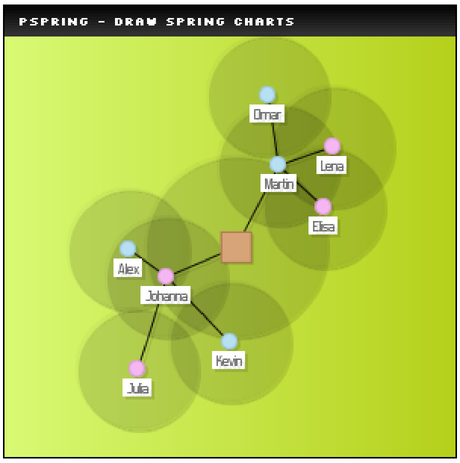
## Calling this function

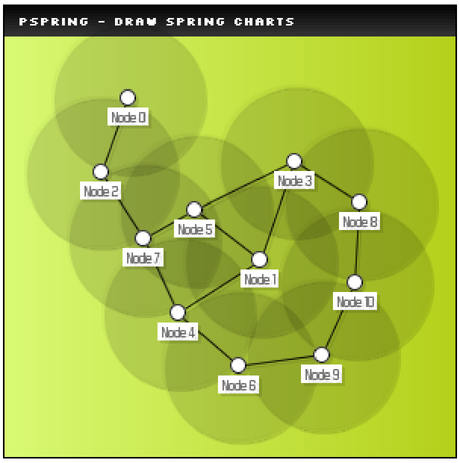
```
pSpring();
```

There is no parameters to give while creating this object.

## Sample script







Code sample

```
/* pChart library inclusions */
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pSpring.class.php");
include("class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(300,300);

/* Create the background */
$myPicture->drawGradientArea(0,0,300,300,DIRECTION_HORIZONTAL,array("StartR"=>217,"StartG"=>250,"StartB"=>116,"EndR"=>181,"EndG"=>209,
"EndB"=>27,"Alpha"=>100));
$myPicture->drawGradientArea(0,0,300,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>100));

$myPicture->drawRectangle(0,0,299,299,array("R"=>0,"G"=>0,"B"=>0));
$myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pSpring - Draw spring charts",array("R"=>255,"G"=>255,"B"=>255));

/* Define the charting area & stuff */
$myPicture->setGraphArea(20,20,280,280);
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>9,"R"=>80,"G"=>80,"B"=>80));
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Create the pSpring object */
$SpringChart = new pSpring();

/* Create some nodes and connections */
$SpringChart->addNode(0,array("Shape"=>NODE_SHAPE_SQUARE,"FreeZone"=>60,"Size"=>20,"NodeType"=>NODE_TYPE_CENTRAL));
$SpringChart->addNode(1,array("Connections"=>"0"));
$SpringChart->addNode(2,array("Connections"=>"0"));
$SpringChart->addNode(3,array("Shape"=>NODE_SHAPE_TRIANGLE,"Connections"=>"1"));
$SpringChart->addNode(4,array("Shape"=>NODE_SHAPE_TRIANGLE,"Connections"=>"1"));
$SpringChart->addNode(5,array("Shape"=>NODE_SHAPE_TRIANGLE,"Connections"=>"1"));
```

```
$SpringChart->addNode(6,array("Connections"=>"2"));
$SpringChart->addNode(7,array("Connections"=>"2"));
$SpringChart->addNode(8,array("Connections"=>"2"));

/* Define the nodes color */
$SpringChart->setNodesColor(0,array("R"=>215,"G"=>163,"B"=>121,"BorderR"=>166,"BorderG"=>115,"BorderB"=>74));
$SpringChart->setNodesColor(array(1,2),array("R"=>150,"G"=>215,"B"=>121,"Surrounding"=>-30));
$SpringChart->setNodesColor(array(3,4,5),array("R"=>216,"G"=>166,"B"=>14,"Surrounding"=>-30));
$SpringChart->setNodesColor(array(6,7,8),array("R"=>179,"G"=>121,"B"=>215,"Surrounding"=>-30));

/* Customize some relations */
$SpringChart->linkProperties(0,1,array("R"=>255,"G"=>0,"B"=>0,"Ticks"=>2));
$SpringChart->linkProperties(0,2,array("R"=>255,"G"=>0,"B"=>0,"Ticks"=>2));

/* Render the spring chart */
$Result = $SpringChart->drawSpring($myPicture);
print_r($Result);

/* Render the picture */
$myPicture->Render("drawspring.png");
```

This will create a **drawspring.png** file and print the algorithm statistics.

```
C:WebLibrary>php -q test-spring.php
Array
(
    [Pass] => 1
    [Conflicts] => 0
)
```

# linkProperties - Defines the properties of a link between two nodes

This function allows you to specify the properties of a link between two nodes. You can specify the link color, ticks and text. Parameters are given trough a **$Settings** array. To learn more about this please read the Format array guide.

## Calling this function

```
linkProperties($FromNode,$ToNode,$Settings);
```

Where :

- FromNode is the ID of the starting node.
- ToNode is the ID of the ending node.

Information

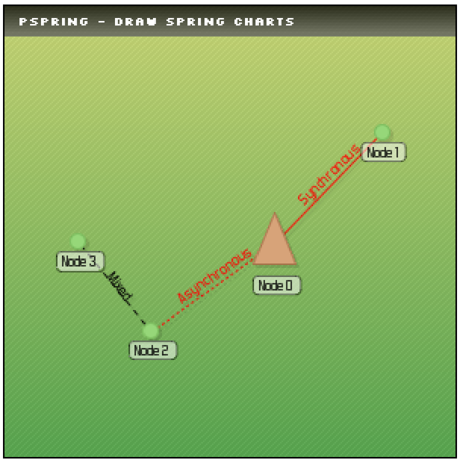
You just have to specify one way of the relation : A -> B or B -> A.

## Settings array - Tune up your link!

It is possible to customize the color, ticks and text of the link with this array.

- Link color can be set with R, G, B.
- Ticks width can be set with Ticks.
- Link text can be set with Name.

## Sample script



Code sample

```
/* pChart library inclusions */
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pSpring.class&.phpquot;);
include("class/pImage.class");

/* Create the pChart object */
$myPicture = new pImage(300,300);

/* Create the background */
$myPicture->drawGradientArea(0,0,300,300,DIRECTION_HORIZONTAL,array("StartR"=>217,"StartG"=>250,"StartB"=>116,"EndR"=>181,"EndG"=>209,
"EndB"=>27,"Alpha"=>100));
$myPicture->drawGradientArea(0,0,300,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>100));
$myPicture->drawRectangle(0,0,299,299,array("R"=>0,"G"=>0,"B"=>0));
$myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pSpring - Draw spring charts",array("R"=>255,"G"=>255,"B"=>255));

/* Define the charting area & stuff */
$myPicture->setGraphArea(20,20,280,280);
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>9,"R"=>80,"G"=>80,"B"=>80));
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Create the pSpring object */
$SpringChart = new pSpring();

/* Define the default free zone size */
$SpringChart->setNodeDefaults(array("FreeZone"=>100));

/* Create some nodes & relations */
$SpringChart->addNode(0,array("Shape"=>NODE_SHAPE_TRIANGLE,"FreeZone"=>60,"Size"=>20,"NodeType"=>NODE_TYPE_CENTRAL));
$SpringChart->addNode(1,array("Connections"=>"0"));
$SpringChart->addNode(2,array("Connections"=>"0"));
```

```
$SpringChart->addNode(3,array("Connections"=>"2"));

/* Define the nodes color */
$SpringChart->setNodesColor(0,array("R"=>215,"G"=>163,"B"=>121,"BorderR"=>166,"BorderG"=>115,"BorderB"=>74));
$SpringChart->setNodesColor(array(1,2,3),array("R"=>150,"G"=>215,"B"=>121,"Surrounding"=>-30));

/* Customize some relations */
$SpringChart->linkProperties(0,1,array("Name"=>"Synchronous","R"=>255,"G"=>0,"B"=>0));
$SpringChart->linkProperties(0,2,array("Name"=>"Asynchronous","R"=>255,"G"=>0,"B"=>0,"Ticks"=>2));
$SpringChart->linkProperties(3,2,array("Name"=>"Mixed","Ticks"=>4));

/* Render the spring chart */
$Result = $SpringChart->drawSpring($myPicture);
print_r($Result);

/* Render the picture */
$myPicture->Render("drawspring4.png");
```

# setNodeDefaults - Defines the default properties of the nodes

This function allows you to specify the default properties of the nodes. Calling this function before defining the nodes allows you to apply default settings like color, shape,.. to all of them. Parameters are given trough a **$Settings** array. To learn more about this please read the Format array guide.

## Calling this function

```
setNodeDefaults ($Settings="");
```

*No parameters required.*

## Settings array - Tune up your nodes!

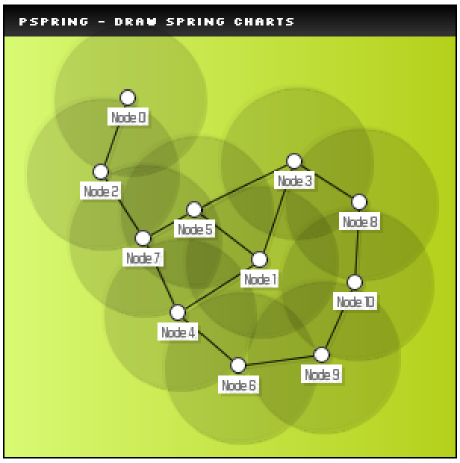
It is possible to specify the following default settings with this array.

- Node color can be set with R, G, B.
- Node border color can be set with BorderR, BorderG, BorderB.
- Surrounding color can be set with Surrounding.
- Label background color can be set with BackgroundR,BackgroundRG,BackgroundB.
- Shape can be set with Shape. (see below)
- Size can be set with Shape.
- Free zone can be set with FreeZone.

Following shapes types are allowed :

- NODE_SHAPE_CIRCLE, this will  draw circular nodes.
- NODE_SHAPE_SQUARE, this will  draw box nodes.
- NODE_SHAPE_TRIANGLE, this will  draw triangular nodes.

## Sample script



Code sample

```
/* pChart library inclusions */
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pSpring.class.php");
include("class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(300,300);

/* Background customization */
$myPicture->drawGradientArea(0,0,300,300,DIRECTION_HORIZONTAL,array("StartR"=>217,"StartG"=>250,"StartB"=>116,"EndR"=>181,"EndG"=>209,
"EndB"=>27,"Alpha"=>100));
$myPicture->drawGradientArea(0,0,300,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>100));
$myPicture->drawRectangle(0,0,299,299,array("R"=>0,"G"=>0,"B"=>0));
$myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pSpring - Draw spring charts",array("R"=>255,"G"=>255,"B"=>255));

/* Prepare the graph area */
$myPicture->setGraphArea(20,20,280,280);
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>9,"R"=>80,"G"=>80,"B"=>80));
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Create the pSpring object */
$SpringChart = new pSpring();

/* Set the nodes default settings */
$SpringChart->setNodeDefaults(array("FreeZone"=>50));

/* Build random nodes & connections */
for($i=0;$i<=10;$i++)
 {
  $Connections = "";
  for($j=0;$j<=rand(0,1);$j++)
   { $Connections[] = rand(0,10); }

  $SpringChart->addNode($i,array("Name"=>"Node ".$i,"Connections"=>$Connections));
 }

/* Compute and draw the Spring Graph */
$Result = $SpringChart->drawSpring($myPicture,array("DrawQuietZone"=>TRUE));

/* Display the statistics */
print_r($Result);

/* Render the picture */
$myPicture->Render("drawspring3.png");
```

# addNode - Add a node

This function allows you to add a new node and its relations with the other nodes. You can definin the node properties like color, shape,.. to all of them. Parameters are given trough a **$Settings** array. To learn more about this please read the Format array guide.

## Calling this function

```
addNode($NodeID,$Settings="")
```

- $NodeID will be the unique ID of the node. We recommend to use numbers.

## Settings array - Tune up your node!

It is possible to specify the following settings with this array.

- Node name can be set with Name.
- Node color can be set with R, G, B.
- Node border color can be set with BorderR, BorderG, BorderB.
- Surrounding color can be set with Surrounding.
- Label background color can be set with BackgroundR,BackgroundRG,BackgroundB.
- Shape can be set with Shape. (see below)
- Size can be set with Shape.
- Free zone can be set with FreeZone.
- Nodes connections must be given in Connections.

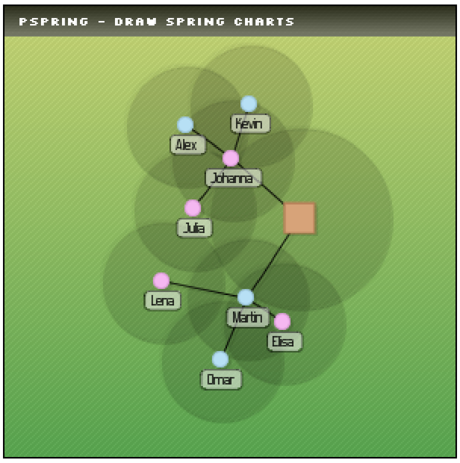
Information

The connections can be given as an array of IDs if they are multiple or with a single ID if there is online one.

Following shapes types are allowed :

- NODE_SHAPE_CIRCLE, this will  draw circular nodes.
- NODE_SHAPE_SQUARE, this will  draw box nodes.
- NODE_SHAPE_TRIANGLE, this will  draw triangular nodes.

## Sample script



Code sample

```
/* pChart library inclusions */
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pSpring.class.php");
include("class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(300,300);

/* Create the background */
$myPicture->drawGradientArea(0,0,300,300,DIRECTION_HORIZONTAL,array("StartR"=>217,"StartG"=>250,"StartB"=>116,"EndR"=>181,"EndG"=>209,
"EndB"=>27,"Alpha"=>100));
$myPicture->drawGradientArea(0,0,300,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>100));

$myPicture->drawRectangle(0,0,299,299,array("R"=>0,"G"=>0,"B"=>0));
$myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pSpring - Draw spring charts",array("R"=>255,"G"=>255,"B"=>255));

/* Define the charting area & stuff */
$myPicture->setGraphArea(20,20,280,280);
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>9,"R"=>80,"G"=>80,"B"=>80));
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Create the pSpring object */
$SpringChart = new pSpring();

/* Create some nodes & relations */
$SpringChart->addNode(0,array("Shape"=>NODE_SHAPE_SQUARE,"FreeZone"=>60,"Size"=>20,"NodeType"=>NODE_TYPE_CENTRAL));
$SpringChart->addNode(1,array("Connections"=>"0"));
$SpringChart->addNode(2,array("Connections"=>"0"));
$SpringChart->addNode(3,array("Shape"=>NODE_SHAPE_TRIANGLE,"Connections"=>"1"));
$SpringChart->addNode(4,array("Shape"=>NODE_SHAPE_TRIANGLE,"Connections"=>"1"));
$SpringChart->addNode(5,array("Shape"=>NODE_SHAPE_TRIANGLE,"Connections"=>"1"));
```

```
$SpringChart->addNode(6,array("Connections"=>"2"));
$SpringChart->addNode(7,array("Connections"=>"2"));
$SpringChart->addNode(8,array("Connections"=>"2"));

/* Define the nodes color */
$SpringChart->setNodesColor(0,array("R"=>215,"G"=>163,"B"=>121,"BorderR"=>166,"BorderG"=>115,"BorderB"=>74));
$SpringChart->setNodesColor(array(1,2),array("R"=>150,"G"=>215,"B"=>121,"Surrounding"=>-30));
$SpringChart->setNodesColor(array(3,4,5),array("R"=>216,"G"=>166,"B"=>14,"Surrounding"=>-30));
$SpringChart->setNodesColor(array(6,7,8),array("R"=>179,"G"=>121,"B"=>215,"Surrounding"=>-30));

/* Customize some relations */
$SpringChart->linkProperties(0,1,array("R"=>255,"G"=>0,"B"=>0,"Ticks"=>2));
$SpringChart->linkProperties(0,2,array("R"=>255,"G"=>0,"B"=>0,"Ticks"=>2));

/* Render the spring chart */
$Result = $SpringChart->drawSpring($myPicture);
print_r($Result);

/* Render the picture */
$myPicture->Render("drawspring.png");
```

# setNodesColor - Bulk change the node colors

This function allows you to set the color of a list of given nodes. You&lsquo;ll have to call this function when the nodes are already created. Parameters are given trough a **$Settings** array. To learn more about this please read the Format array guide.

## Calling this function

```
setNodesColor($Nodes,$Settings="");
```

Where :

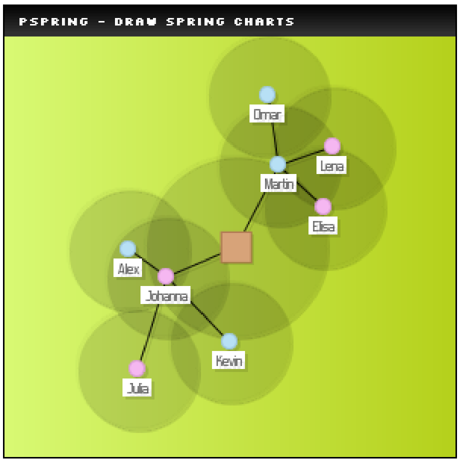
- Nodes is an array of nodes IDs or a single node ID.

## Settings array - Tune up your node!

It is possible to customize the color, ticks and text of the link with this array.

- Node(s) color can be set with R, G, B.
- Node(s) border color can be set with BorderR, BorderG, BorderB.
- Node(s) surrounding color can be set with Surrounding.

## Sample script



Code sample

```
/* pChart library inclusions */
```

```
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pSpring.class.php");
include("class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(300,300);
$myPicture->drawGradientArea(0,0,300,300,DIRECTION_HORIZONTAL,array("StartR"=>217,"StartG"=>250,"StartB"=>116,"EndR"=>181,"EndG"=>209,"EndB"=>27,"Alpha"=>100));
$myPicture->drawGradientArea(0,0,300,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>100));
$myPicture->drawRectangle(0,0,299,299,array("R"=>0,"G"=>0,"B"=>0));
$myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pSpring - Draw spring charts",array("R"=>255,"G"=>255,"B"=>255));

/* Draw the background */
$myPicture->setGraphArea(20,20,280,280);
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>9,"R"=>80,"G"=>80,"B"=>80));
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Create the pSpring object */
$SpringChart = new pSpring();

/* Create some nodes and relations */
$SpringChart->addNode("0",array("Name"=>"","Shape"=>NODE_SHAPE_SQUARE,"FreeZone"=>60,"Size"=>20,"NodeType"=>NODE_TYPE_CENTRAL));
$SpringChart->addNode("1",array("Name"=>"Johanna","Connections"=>array("0")));
$SpringChart->addNode("2",array("Name"=>"Martin","Connections"=>array("0")));
$SpringChart->addNode("3",array("Name"=>"Kevin","Connections"=>array("1")));
$SpringChart->addNode("4",array("Name"=>"Alex","Connections"=>array("1")));
$SpringChart->addNode("5",array("Name"=>"Julia","Connections"=>array("1")));
$SpringChart->addNode("6",array("Name"=>"Lena","Connections"=>array("2")));
$SpringChart->addNode("7",array("Name"=>"Elisa","Connections"=>array("2")));
$SpringChart->addNode("8",array("Name"=>"Omar","Connections"=>array("2")));

/* Define the nodes color */
$SpringChart->setNodesColor(array(0),array("R"=>215,"G"=>163,"B"=>121,"BorderR"=>166,"BorderG"=>115,"BorderB"=>74));
$SpringChart->setNodesColor(array(1,5,6,7),array("R"=>245,"G"=>183,"B"=>241,"Surrounding"=>-30));
$SpringChart->setNodesColor(array(2,3,4,8),array("R"=>183,"G"=>224,"B"=>245,"Surrounding"=>-30));

/* Draw the Sprint graph */
$Result = $SpringChart->drawSpring($myPicture,array("DrawQuietZone"=>TRUE));
```

```
/* Display the statistics */
print_r($Result);

/* Render the picture */
$myPicture->Render("drawspring2.png");
```

# dumpNodes - Retrieve nodes config

This function allows you to retrieve the current node config. This can be used for debugging purpose.

## Calling this function

```
dumpNodes();
```

*No parameters required*

## Sample script

Code sample

```
/* pChart library inclusions */
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pSpring.class.php");
include("class/pImage.class.php");

/* Create the pChart object */
$myPicture = new pImage(300,300);

$myPicture->drawGradientArea(0,0,300,300,DIRECTION_HORIZONTAL,array("StartR"=>217,"StartG"=>250,"StartB"=>116,"EndR"=>181,"EndG"=>209,
"EndB"=>27,"Alpha"=>100));

$myPicture->drawGradientArea(0,0,300,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>100));

$myPicture->drawRectangle(0,0,299,299,array("R"=>0,"G"=>0,"B"=>0));
$myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pSpring - Draw spring charts",array("R"=>255,"G"=>255,"B"=>255));

/* Draw the background */
$myPicture->setGraphArea(20,20,280,280);
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>9,"R"=>80,"G"=>80,"B"=>80));
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Create the pSpring object */
$SpringChart = new pSpring();

/* Create some nodes and relations */
```

```
$SpringChart->addNode("0",array("Name"=>"","Shape"=>NODE_SHAPE_SQUARE,"FreeZone"=>60,"Size"=>20,"NodeType"=>NODE_TYPE_CENTRA
L));

 $SpringChart->addNode("1",array("Name"=>"Johanna","Connections"=>array("0")));
 $SpringChart->addNode("2",array("Name"=>"Martin","Connections"=>array("0")));
 $SpringChart->addNode("3",array("Name"=>"Kevin","Connections"=>array("1")));
 $SpringChart->addNode("4",array("Name"=>"Alex","Connections"=>array("1")));
 $SpringChart->addNode("5",array("Name"=>"Julia","Connections"=>array("1")));
 $SpringChart->addNode("6",array("Name"=>"Lena","Connections"=>array("2")));
 $SpringChart->addNode("7",array("Name"=>"Elisa","Connections"=>array("2")));
 $SpringChart->addNode("8",array("Name"=>"Omar","Connections"=>array("2")));

 /* Define the nodes color */
 $SpringChart->setNodesColor(array(0),array("R"=>215,"G"=>163,"B"=>121,"BorderR"=>166,"BorderG"=>115,"BorderB"=>74));
 $SpringChart->setNodesColor(array(1,5,6,7),array("R"=>245,"G"=>183,"B"=>241,"Surrounding"=>-30));
 $SpringChart->setNodesColor(array(2,3,4,8),array("R"=>183,"G"=>224,"B"=>245,"Surrounding"=>-30));

 /* Dump the nodes config */
 print_r($SpringChart->dumpNodes());
```

This will display something like :

```
Array
(
    [0] => Array
        (
            [R] => 215
            [G] => 163
            [&#66;] => 121
            [BorderR] => 166
            [BorderG] => 115
            [BorderB] => 74
            [BackgroundR] => 255
            [BackgroundG] => 255
            [BackgroundB] => 255
            [Name] =>
            [Force] => 1
            [Type] => 690002
            [Size] => 20
            [Shape] => 690013
            [FreeZone] => 60
            [Connections] => Array
                (
                    [0] => 1
                    [1] => 2
                )

            [X] => 160.63714209637
            [Y] => 159.46856112912
        )
(...)
```

# drawSpring - Draw a Spring graph

This function allows will draw a spring graph. Parameters are given trough a **$Settings** array. To learn more about this please read the Format array guide.

Information

Based on the number of nodes, the amount of connection and the graph area size, it may be not possible to create a graph without links overlap. As the spring algorithm is an almost linear one, the initial placement formula (defined with the "Algorithm" parameter) will do the difference in the final layout.

## Calling this function

```
drawSpring($Object,$Settings="");
```

Where :

- Object is the reference to a pImage object.

## Settings array - Tune up your link!

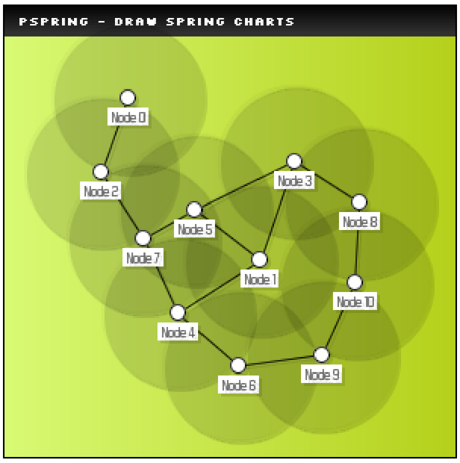
It is possible to customize the rendering with this array.

- You can specify the initial placement algorithm with Algorithm. (default is ALGORITHM_WEIGHTED)
- The number of pass can be adjusted with Pass. (default is 50)
- The number of retries can be set with Retries. (default is 10)
- You can specify the text padding with TextPadding. (default is 4)
- You can specify the attraction force with MagneticForceA. (default is 1.5)
- You can specify the repulsion force with MagneticForceR. (default is 2)
- You can draw the force vectors setting DrawVectors to TRUE.
- You can draw the quiet zonesDrawQuietZone to TRUE.

Following algorithms are allowed :

- ALGORITHM_RANDOM, nodes will be placed randomly.
- ALGORITHM_WEIGHTED, node will be placed depending of their number of connections.
- ALGORITHM_CIRCULAR, based on the weighted algorithm, this will limit the number of crossing links.

## Sample script



Code sample

```
/* pChart library inclusions */
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pSpring.class.php");
include("class/pImage.class.php");
```

```
/* Create the pChart object */
$myPicture = new pImage(300,300);

/* Background customization */
$myPicture->drawGradientArea(0,0,300,300,DIRECTION_HORIZONTAL,array("StartR"=>217,"StartG"=>250,"StartB"=>116,"EndR"=>181,"EndG"=>209,
"EndB"=>27,"Alpha"=>100));
$myPicture->drawGradientArea(0,0,300,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>100));
$myPicture->drawRectangle(0,0,299,299,array("R"=>0,"G"=>0,"B"=>0));
$myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pSpring - Draw spring charts",array("R"=>255,"G"=>255,"B"=>255));

/* Prepare the graph area */
$myPicture->setGraphArea(20,20,280,280);
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>9,"R"=>80,"G"=>80,"B"=>80));
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Create the pSpring object */
$SpringChart = new pSpring();

/* Set the nodes default settings */
$SpringChart->setNodeDefaults(array("FreeZone"=>50));

/* Build random nodes & connections */
for($i=0;$i<=10;$i++)
 {
  $Connections = "";
  for($j=0;$j<=rand(0,1);$j++)
   { $Connections[] = rand(0,10); }

  $SpringChart->addNode($i,array("Name"=>"Node ".$i,"Connections"=>$Connections));
 }

/* Compute and draw the Spring Graph */
$Result = $SpringChart->drawSpring($myPicture,array("DrawQuietZone"=>TRUE));

/* Display the statistics */
print_r($Result);

/* Render the picture */
$myPicture->Render("drawspring3.png");
```

# pCache functions

# pCache - Reduce your CPU time

This class implements a simple caching method for your charts. You can use it to reduce the web server CPU time, speed up the web transaction and improve the user experience while people will browse your pages.

Information

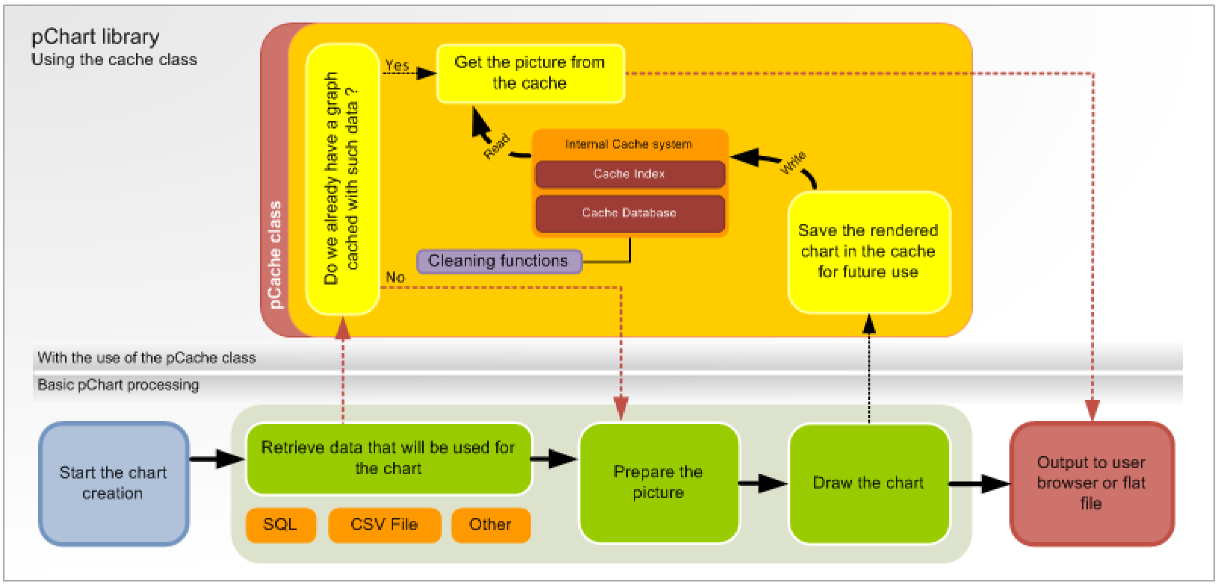
Caching can only work efficiently if the users of your web site are displaying the same picture many time, in the case that you are displaying real-time data with a small refresh time, caching will be useless.

The pCache class makes use of two flat files : **index.db** that contains for each cached entries :

- The unique hash. (or key)
- Theposition of the object in the raw database.
- The size of the object.
- The date it has been rendered.
- The number of hits for this object.

..and **cache.db** that contains the raw pictures. The web server must be able to get full access to this files. If you also want to make use of the cache removal function, the best is to create a cache folder and grant full access to the web server user on it.

The basic cache workflow is described bellow :


pChart library
Using the cache class
pCache class
Do we already have a graph cached with such data ?
Yes
Get the picture from the cache
Read
Internal Cache system
Cache Index
Cache Database
Write
Save the rendered chart in the cache for future use
Cleaning functions
No
With the use of the pCache class
Basic pChart processing
Start the chart creation
Retrieve data that will be used for the chart
SQL
CSV File
Other
Prepare the picture
Draw the chart
Output to user browser or flat file

Using this class will require that you change a bit your way of coding. Cache can be tested even before creating the pDraw object. If you use it wisely this will drasticaly lower the CPU & memory use of your web server.

## Calling this function

```
pCache($Settings="");
```

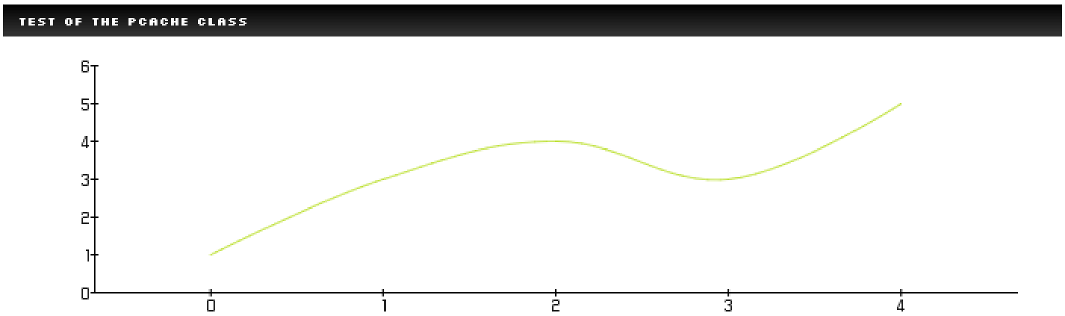
- Settings is an optional array containing the cache configuration parameters.

## Cache settings - Override default settings!

The Settings array allow you to redefine the path to the cache folder and the name of the database files. By default, the class will assume the existence of a "cache" folder at the same level of your calling script. The database index file will be called **index.db** and the database solid file will be called **cache.db**.

- The name of the cache folder can be changed with CacheFolder.
- The name of the index file can be changed with CacheIndex.
- The name of the database solid file can be changed with CacheDB

## Sample script



Code sample

```
/* Include all the classes */
include("class/pDraw.class.php");
include("class/pImage.class.php");
include("class/pData.class.php");
include("class/pCache.class.php");

/* Create your dataset object */
$myData = new pData();


/* Add data in your dataset */
$myData->addPoints(array(1,3,4,3,5));

/* Create the cache object */
$myCache = new pCache();

/* Compute the hash linked to the chart data */
$ChartHash = $myCache->getHash($myData);

/* Test if we got this hash in our cache already */
if ( $myCache->isInCache($ChartHash))
 {
  /* If we have it, get the picture from the cache! */
  $myCache->saveFromCache($ChartHash,"cache.png");
 }
else
 {
  /* Create a pChart object and associate your dataset */
  $myPicture = new pImage(700,230,$myData);

  /* Choose a nice font */
  $myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>11));

  /* Define the boundaries of the graph area */
  $myPicture->setGraphArea(60,40,670,190);

  /* Draw the scale, keep everything automatic */
  $myPicture->drawScale();

  /* Draw the scale, keep everything automatic */
  $myPicture->drawSplineChart();

  /* Do some cosmetics */
$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>100));
  $myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
  $myPicture->drawText(10,13,"Test of the pCache class",array("R"=>255,"G"=>255,"B"=>255));

  /* Push the rendered picture to the cache */
  $myCache->writeToCache($ChartHash,$myPicture);

  /* Render the picture */
  $myPicture->Render("cache.png");
 }
```

The first time that this code will be executed, the picture will be rendered and pushed in the cache, the next times, it will be retrieved from the cache directly saving CPU and execution time.

# writeToCache - Push an object in the cache

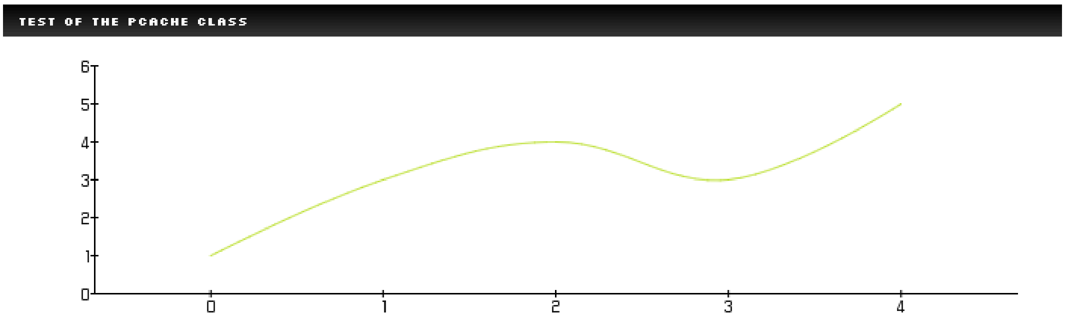
This function allows you to push a rendered picture (object) in the static cache. Caching can greatly enhance the user experience of the people browsing your web application by lowering the web server CPU / memory usage and speed up the page rendering time. Every cached object must have an unique ID, see the getHash function for more information.

## Calling this function

```
writeToCache($ID,$pChartObject);
```

- ID is the unique ID affected to this object.
- pChartObject is the pChart object containing the picture you want to cache.

## Sample script



Code sample

```
/* Include all the classes */
include("class/pDraw.class.php");
include("class/pImage.class.php");
include("class/pData.class.php");
include("class/pCache.class.php");

/* Create your dataset object */
$myData = new pData();


/* Add data in your dataset */
$myData->addPoints(array(1,3,4,3,5));

/* Create the cache object */
$myCache = new pCache();
```

```
/* Compute the hash linked to the chart data */
$ChartHash = $myCache->getHash($myData);

/* Test if we got this hash in our cache already */
if ( $myCache->isInCache($ChartHash))
 {
  /* If we have it, get the picture from the cache! */
  $myCache->saveFromCache($ChartHash,"cache.png");
 }
else
 {
  /* Create a pChart object and associate your dataset */
  $myPicture = new pImage(700,230,$myData);

  /* Choose a nice font */
  $myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>11));

  /* Define the boundaries of the graph area */
  $myPicture->setGraphArea(60,40,670,190);

  /* Draw the scale, keep everything automatic */
  $myPicture->drawScale();

  /* Draw the scale, keep everything automatic */
  $myPicture->drawSplineChart();

  /* Do some cosmetics */



$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"


Alpha"=>100));

  $myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
  $myPicture->drawText(10,13,"Test of the pCache class",array("R"=>255,"G"=>255,"B"=>255));

  /* Push the rendered picture to the cache */
  $myCache->writeToCache($ChartHash,$myPicture);

  /* Render the picture */
  $myPicture->Render("cache.png");
 }
```

The first time that this code will be executed, the picture will be rendered and pushed in the cache, the next times, it will be retrieved from the cache directly saving CPU and execution time.

## removeOlderThan - Remove object based on their age

This function allows you to remove objects from the cache based on the date where they‘ve been generated. It can be used to flush all object that have been generated more than 24 hours ago if your application got dailly changes on the graphs layouts.

### Calling this function

```
removeOlderThan($Expiry)
```

- Expiry is an amount of seconds. Objects pushed in the cache with an age greated of this value will be removed.

### Sample script

This will remove objects pushed more than one day ago.

Code sample

```
/* Include all the classes */
include("class/pCache.class.php");

/* Create the cache object */
$myCache = new pCache();

/* Remove objects older than one day */
$myCache->removeOlderThan(60*60*24);
```

## remove - Remove one object from the cache

This function allows you to remove one object of the cache based on its unique ID. As removing an object from the cache require a whole rebuild of the cache index and cache database this action will take an amount of time linked to the number of objects contained in the cache.

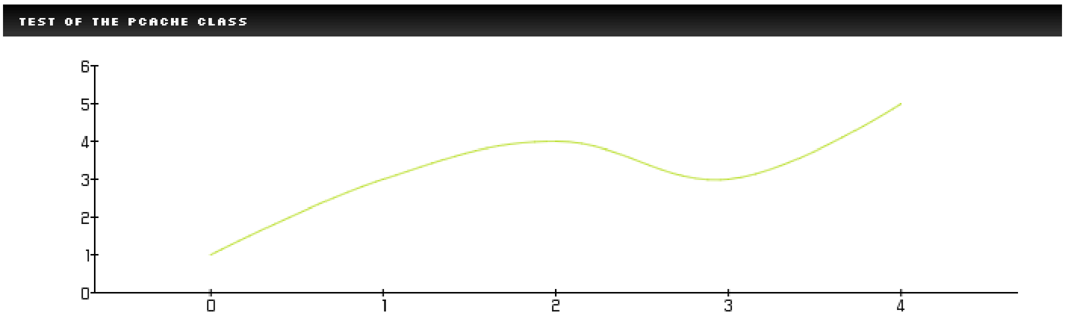
### Calling this function

```
remove($ID)
```

- ID is the unique ID associated to the cache object. This ID can be manually defined or hash-based.

### Sample script

If the object is already in the database, it will be re-generated and pushed back to the cache :



Code sample

```
/* Include all the classes */
include("class/pDraw.class.php");
include("class/pImage.class.php");
include("class/pData.class.php");
include("class/pCache.class.php");

/* Create your dataset object */
$myData = new pData();


/* Add data in your dataset */
$myData->addPoints(array(1,3,4,3,5));

/* Create the cache object */
$myCache = new pCache();

/* Compute the hash linked to the chart data */
$ChartHash = $myCache->getHash($myData);

/* Test if we got this hash in our cache already and remove it */
```

```
if ( $myCache->isInCache($ChartHash))  {  $myCache->remove($ChartHash);  }

/* Create a pChart object and associate your dataset */
$myPicture = new pImage(700,230,$myData);

/* Choose a nice font */
$myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>11));

/* Define the boundaries of the graph area */
$myPicture->setGraphArea(60,40,670,190);

/* Draw the scale, keep everything automatic */
$myPicture->drawScale();

/* Draw the scale, keep everything automatic */
$myPicture->drawSplineChart();

/* Do some cosmetics */
$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>100));
$myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"Test of the pCache class",array("R"=>255,"G"=>255,"B"=>255));

/* Push the rendered picture to the cache */
$myCache->writeToCache($ChartHash,$myPicture);

/* Render the picture */
$myPicture->Render("cache.png");
```

# isInCache - Test if an object is already cached

This function allows you to check if an object is already in the cache. It can also returns detailled informations about the cached entry if any.

## Calling this function

```
isInCache($ID,$Verbose=FALSE)
```

- ID is the unique ID of the cached object.
- Verbose if set to yes, this will returns some extended informations about the cached entry.

If **Verbose** is kept to FALSE, this function will either returns TRUE if the object is cached or FALSE if it's not present in the cahe. If **Verbose** is set to TRUE, this function will return an array containing :

- DBPos : The position of the picture in the solid database.
- PicSize : The size of the picture.
- GeneratedTS : The timestamp when the object has been pushed to the cache.
- Hits : The number of times the object has been pulled off the cache.

## Sample script

Code sample

```
/* Include all the classes */
include("class/pData.class.php");
include("class/pCache.class.php");
```

```
/* Create your dataset object */
$myData = new pData();


/* Add data in your dataset */
$myData->addPoints(array(1,3,4,3,5));

/* Create the cache object */
$myCache = new pCache();

/* Compute the hash linked to the chart data */
$ID = $myCache->getHash($myData);

/* Display the informations about the cached entry */
print_r($myCache->isInCache($ID,TRUE));
```

This will returns an array :

```
Array
(
   [DBPos] => 0
   [PicSize] => 7202
   [GeneratedTS] => 1282652176
   [Hits] => 4
)
```

## saveFromCache - Pick up from the cache and write it to a file

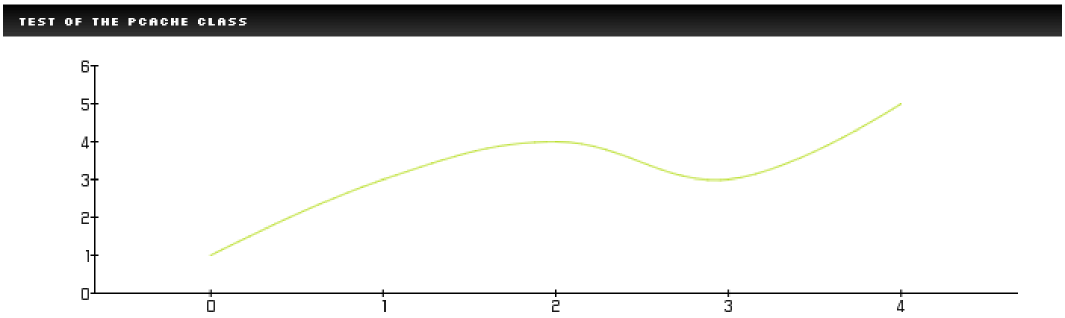
This function allows you to retrieve a cached object and write it as a file on the web server. Basically it will works the same way than where you‘ll call the **render()** function from the pChart class.

### Calling this function

```
saveFromCache($ID,$Destination);
```

- ID is the unique ID affected to this object.
- $Destination is the path to a file where the picture will be saved.

### Sample script



Code sample

```
/* Include all the classes */
include("class/pDraw.class.php");
include("class/pImage.class.php");
include("class/pData.class.php");
include("class/pCache.class.php");

/* Create your dataset object */
$myData = new pData();
```

```
/* Add data in your dataset */
$myData->addPoints(array(1,3,4,3,5));

/* Create the cache object */
$myCache = new pCache();

/* Compute the hash linked to the chart data */
$ChartHash = $myCache->getHash($myData);

/* Test if we got this hash in our cache already */
if ( $myCache->isInCache($ChartHash))
 {
  /* If we have it, get the picture from the cache! */
  $myCache->saveFromCache($ChartHash,"cache.png");
 }
else
 {
  /* Create a pChart object and associate your dataset */
  $myPicture = new pImage(700,230,$myData);

  /* Choose a nice font */
  $myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>11));

  /* Define the boundaries of the graph area */
  $myPicture->setGraphArea(60,40,670,190);

  /* Draw the scale, keep everything automatic */
  $myPicture->drawScale();

  /* Draw the scale, keep everything automatic */
  $myPicture->drawSplineChart();

  /* Do some cosmetics */

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>100));

  $myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
  $myPicture->drawText(10,13,"Test of the pCache class",array("R"=>255,"G"=>255,"B"=>255));

  /* Push the rendered picture to the cache */
  $myCache->writeToCache($ChartHash,$myPicture);

  /* Render the picture */
  $myPicture->render("cache.png");
 }
```

The first time that this code will be executed, the picture will be written in a local file (cache.png) and pushed in the cache, the next times, it will be retrieved from the cache directly saving CPU and execution time.

## strokeFromCache - Pick up from the cache and send to the browser

This function allows you to retrieve a cached object and send it directly to the user web browser. Basically it will works the same way than where you'll call the **stroke()** function from the pChart class.

Information

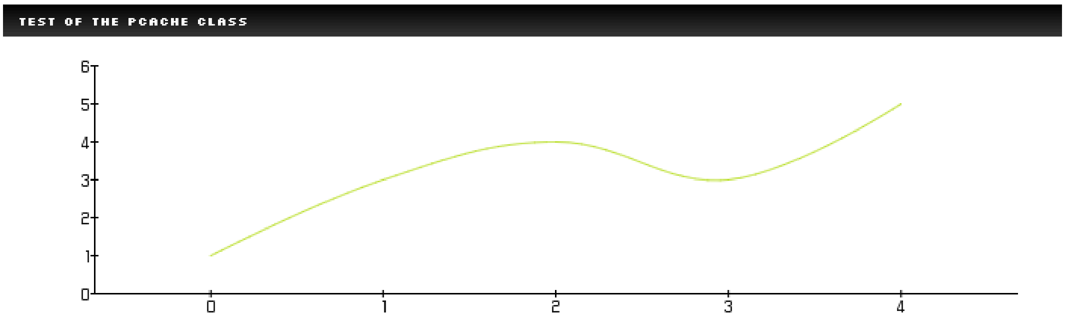
This function will automatically send the Content-type header to the user web browser.

## Calling this function

```
strokeFromCache($ID);
```

- ID is the unique ID affected to this object.

## Sample script



Code sample

```
/* Include all the classes */
include("class/pDraw.class.php");
include("class/pImage.class.php");
include("class/pData.class.php");
include("class/pCache.class.php");

/* Create your dataset object */
$myData = new pData();


/* Add data in your dataset */
$myData->addPoints(array(1,3,4,3,5));

/* Create the cache object */
$myCache = new pCache();

/* Compute the hash linked to the chart data */
$ChartHash = $myCache->getHash($myData);

/* Test if we got this hash in our cache already */
if ( $myCache->isInCache($ChartHash))
 {
  /* If we have it, get the picture from the cache! */
  $myCache->strokeFromCache($ChartHash);
 }
else
 {
  /* Create a pChart object and associate your dataset */
  $myPicture = new pImage(700,230,$myData);

  /* Choose a nice font */
  $myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>11));

  /* Define the boundaries of the graph area */
  $myPicture->setGraphArea(60,40,670,190);

  /* Draw the scale, keep everything automatic */
  $myPicture->drawScale();

  /* Draw the scale, keep everything automatic */
  $myPicture->drawSplineChart();
  /* Do some cosmetics */



$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>100));

  $myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
  $myPicture->drawText(10,13,"Test of the pCache class",array("R"=>255,"G"=>255,"B"=>255));

  /* Push the rendered picture to the cache */
  $myCache->writeToCache($ChartHash,$myPicture);

  /* Render the picture */
  $myPicture->stroke();
 }
```

The first time that this code will be executed, the picture will be sent to the user browser and pushed in the cache, the next times, it will be retrieved from the cache directly saving CPU and execution time.

# getHash - Returns the hash of a pData object

This function allows you to calculate the (nearly) unique hash of a pData object. This hash can be used to match a picture with an unique ID in the cache database. Speaking with figures, you have only one chance on 2339099464253592691 tries to find two pData objects with the same ID, we can safely rely on it to identify different charts. Moreover, you can use the **Marker** parameter to specify which chart template you‘re using, so same data on different charts layout will get different IDs.

## Calling this function

```
getHash($Data,$Marker="")
```

- $Data is a pData object.
- $Marker is an optional parameter allowing you to create different ID even if the pData objet is the same.

## Sample script

This will display the unique ID of the given pData object.

Code sample

```
/* Include all the classes */
include("class/pData.class.php");
include("class/pCache.class.php");

/* Create your dataset object */
$myData = new pData();


/* Add data in your dataset */
$myData->addPoints(array(1,3,4,3,5));

/* Create the cache object */
$myCache = new pCache();

/* Compute the hash linked to the chart data */
echo $myCache->getHash($myData);
```

# flush - Clear the cache contents

This function allows you to flush the contents of the cache database. You can use it to do a mass cleanup of all the cached object.

## Calling this function

```
flush()
```

There is no parameters.

## Sample script

This will flush the whole cache contents.

Code sample

```
/* Include all the classes */
include("class/pCache.class.php");

/* Create the cache object */
$myCache = new pCache();

/* Flush the cache contents*/
$myCache->flush();
```

# Pie charts class

# setSliceColor - define the color of a slice

Because the pie charts are using a variant of the standard pData object structure (all values are given through one serie), it's not possible to change a slice color playing with the standard palette options (that can change only series color). All the color parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
setSliceColor($SliceID,$Format="");
```

Where :

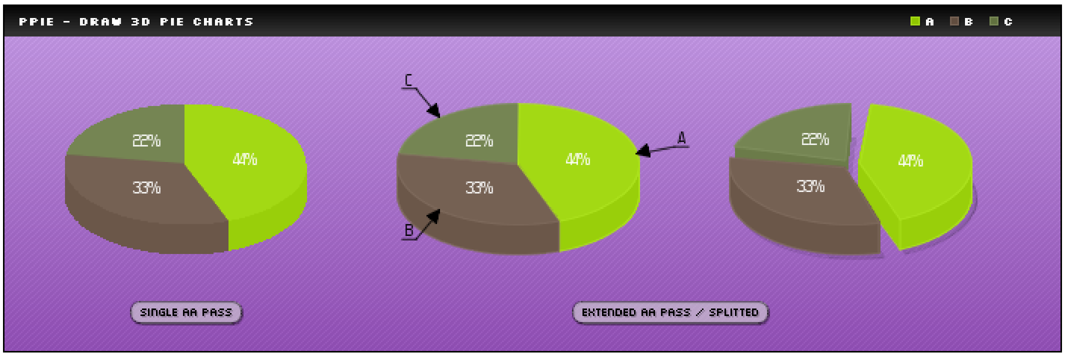
- SliceID is the ID of the slice you want to change the color (numbering start at 0)
- Format is an array containing the color parameters you want to apply.

## Customisation array - Tune up your color!

It is possible to customize the slice rendering by playing with this array. Providing a detailled configuration is not mandatory. You'll see below a representation of all the customization possible :

- The new color can be set with R,G,B.
- The transparency factor can be set with Alpha.

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pPie.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(40,30,20),"ScoreA");
$MyData->setSerieDescription("ScoreA","Application A");

/* Define the absissa serie */
$MyData->addPoints(array("A","B","C"),"Labels");
$MyData->setAbscissa("Labels");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw a solid background */
$Settings = array("R"=>173, "G"=>152, "B"=>217, "Dash"=>1, "DashR"=>193, "DashG"=>172, "DashB"=>237);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Draw a gradient overlay */
$Settings = array("StartR"=>209, "StartG"=>150, "StartB"=>231, "EndR"=>111, "EndG"=>3, "EndB"=>138, "Alpha"=>50);
```

```
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>100));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pPie - Draw 3D pie charts",array("R"=>255,"G"=>255,"B"=>255));

/* Set the default font properties */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>10,"R"=>80,"G"=>80,"B"=>80));

/* Create the pPie object */
$PieChart = new pPie($myPicture,$MyData);

/* Define the slice color */
$PieChart->setSliceColor(0,array("R"=>143,"G"=>197,"B"=>0));
$PieChart->setSliceColor(1,array("R"=>97,"G"=>177,"B"=>63));
$PieChart->setSliceColor(2,array("R"=>97,"G"=>113,"B"=>63));

/* Draw a simple pie chart */
$PieChart->draw3DPie(140,125,array("SecondPass"=>FALSE));

/* Draw an AA pie chart */
$PieChart->draw3DPie(340,125,array("DrawLabels"=>TRUE,"Border"=>TRUE));

/* Enable shadow computing */
$myPicture->setShadow(TRUE,array("X"=>3,"Y"=>3,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Draw a splitted pie chart */
$PieChart->draw3DPie(540,125,array("DataGapAngle"=>10,"DataGapRadius"=>6,"Border"=>TRUE));

/* Write the legend */
$myPicture->setFontProperties(array("FontName"=>"../fonts/MankSans.ttf","FontSize"=>11));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));
$myPicture->drawText(140,200,"Single AA pass",array("R"=>0,"G"=>0,"B"=>0,"Align"=>TEXT_ALIGN_TOPMIDDLE));
$myPicture->drawText(440,200,"Extended AA pass / Splitted",array("R"=>0,"G"=>0,"B"=>0,"Align"=>TEXT_ALIGN_TOPMIDDLE));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.draw3DPie.png");
```

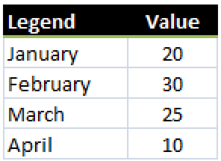
## Pie charts basics

pChart can render both 2D and 3D pie charts. Data are provided through the pData class with some limitation because only one data serie may be rendered on this kind of charts.

- Only the 1st data serie that is marked as Drawable will be used.
- An abscissa serie must be set, it will be used for labelling the data.

## Exemple

Let‘s assume that we got the data represented in the table bellow :

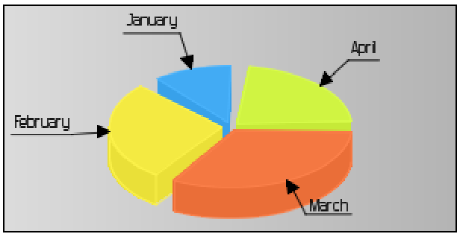


| Legend | Value |
|---|---|
| January | 20 |
| February | 30 |
| March | 25 |
| April | 10 |

This will lead to the following declaration with the pData object structure :

Code sample

```
/* pData object creation */
$MyData = new pData();

/* Data definition */
$MyData->addPoints(array(20,30,25,10),"Value");

/* Labels definition */
$MyData->addPoints(array("January","February","March","April"),"Legend");
$MyData->setAbscissa("Legend");
```



## Full example code

Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pPie.class.php");
include("../class/pImage.class.php");

/* pData object creation */
$MyData = new pData();


/* Data definition */
$MyData->addPoints(array(20,30,25,10),"Value");


/* Labels definition */
$MyData->addPoints(array("January","February","March","April"),"Legend");
$MyData->setAbscissa("Legend");

/* Create the pChart object */
$myPicture = new pImage(300,150,$MyData);

/* Draw a gradient background */




$myPicture->drawGradientArea(0,0,300,300,DIRECTION_HORIZONTAL,array("StartR"=>220,"StartG"=>220,"StartB"=>220,"EndR"=>180,"EndG"=>180,
"EndB"=>180,"Alpha"=>100));


/* Add a border to the picture */
$myPicture->drawRectangle(0,0,299,149,array("R"=>0,"G"=>0,"B"=>0));

/* Create the pPie object */
$PieChart = new pPie($myPicture,$MyData);

/* Enable shadow computing */
$myPicture->setShadow(FALSE);

/* Set the default font properties */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>10,"R"=>80,"G"=>80,"B"=>80));

/* Draw a splitted pie chart */
$PieChart->draw3DPie(150,100,array("Radius"=>80,"DrawLabels"=>TRUE,"DataGapAngle"=>10,"DataGapRadius"=>6,"Border"=>TRUE));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pie.png");
```

# drawPieLegend - Write the legend of your pie charts

This function will write the legend of your pie charts based on the defined abscissa series set with the setAbscissa function. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
drawPieLegend($X,$Y,$Format="");
```

Where :

- X and Y are the coordinates where will be drawn the legend.
- Format is an array containing the drawing parameters of the arrow.

## Customisation array - Tune up your legend!

It is possible to customize the way your legend will be rendered by playing with this array. Providing a detailled configuration is not mandatory, by default the legend will be drawn in a soft grey box with curvy corners.

- You can specify the font file that will be used with FontName.
- You can specify the font size with FontSize.
- You can specify the font color with FontR,FontG,FontB.
- You can specify the size of the colored boxes with BoxSize.
- You can specify the inner margins with Margin.
- You can specify the background color with R,G,B.
- You can specify the background alpha with Alpha.
- You can specify the border color using BorderR,BorderG,BorderB.
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.

You can choose the style that will be applied to you legend box using the **Style** parameter :

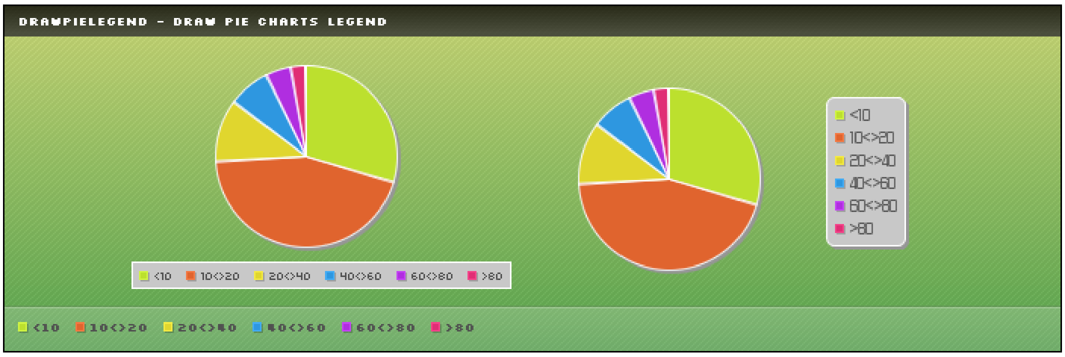
- LEGEND_NOBORDER no borders will be drawn around the legend.
- LEGEND_BOX a rectangle will be drawn around the legend.
- LEGEND_ROUND a rounded rectangle will be drawn around the legend.

You can also define the way the legend will be written using the **Mode** parameter :

- LEGEND_VERTICAL that will stack vertically the series.

- LEGEND_HORIZONTAL that will stack horizontally the series.

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pPie.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(40,60,15,10,6,4),"ScoreA");
$MyData->setSerieDescription("ScoreA","Application A");

/* Define the absissa serie */
$MyData->addPoints(array("<10","10<>20","20<>40","40<>60","60<>80",">80"),"Labels");
$MyData->setAbscissa("Labels");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>80));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawPieLegend - Draw pie charts legend",array("R"=>255,"G"=>255,"B"=>255));

/* Set the default font properties */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>10,"R"=>80,"G"=>80,"B"=>80));

/* Enable shadow computing */
```

```
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>150,"G"=>150,"B"=>150,"Alpha"=>100));

/* Create the pPie object */
$PieChart = new pPie($myPicture,$MyData);

/* Draw two AA pie chart */
$PieChart->draw2DPie(200,100,array("Border"=>TRUE));
$PieChart->draw2DPie(440,115,array("Border"=>TRUE));

/* Write down the legend next to the 2nd chart*/
$PieChart->drawPieLegend(550,70);

/* Write a legend box under the 1st chart */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
$PieChart->drawPieLegend(90,176,array("Style"=>LEGEND_BOX,"Mode"=>LEGEND_HORIZONTAL));

/* Write the bottom legend box */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));

$myPicture->drawGradientArea(1,200,698,228,DIRECTION_VERTICAL,array("StartR"=>247,"StartG"=>247,"StartB"=>247,"EndR"=>217,"EndG"=>217,"
EndB"=>217,"Alpha"=>20));

$myPicture->drawLine(1,199,698,199,array("R"=>100,"G"=>100,"B"=>100,"Alpha"=>20));
$myPicture->drawLine(1,200,698,200,array("R"=>255,"G"=>255,"B"=>255,"Alpha"=>20));
$PieChart->drawPieLegend(10,210,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture to a file */
$myPicture->Render("example.drawPieLegend.png");
```

# draw2DPie - Draw 2D pie charts

This function allows you to draw a 2D pie chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
draw2DPie($X,$Y,$Format);
```

Where :

- X,Y is the center coordinate of the pie. (Radius can be adjusted in the Format array)
- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

It is possible to customize the pie chart rendering by playing with this array. Providing a detailled configuration is not mandatory. You'll see below a representation of all the customization possible :

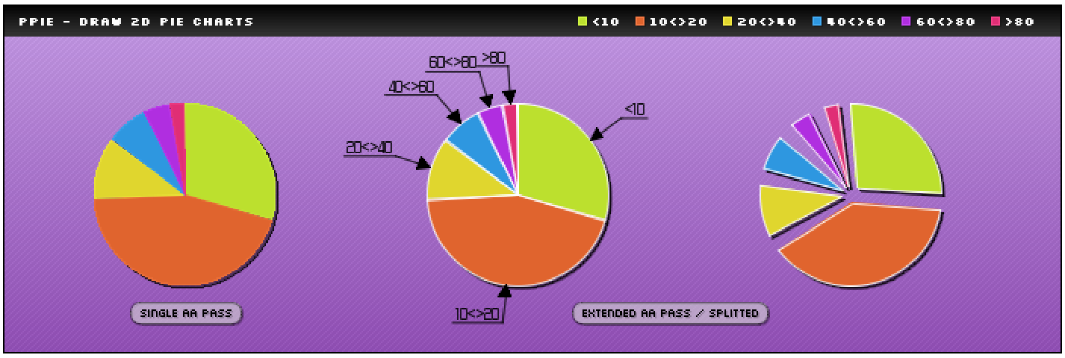
- You can specify if the radius of the pie chart with Radius.
- You can specify the angle in degree between two values with DataGapAngle.
- If your specify a gap, you can set the inner radius withDataGapRadius.
- You can enable a 2nd AA pass setting SecondPass to TRUE.
- You can draw a border around the pie parts setting Border to TRUE.
- You can set the border color with BorderR,BorderG,BorderB.
- You can write the data labels setting DrawLabels to TRUE.

- The labels colors can be set with LabelR,LabelG,LabelB,LabelAlpha.

The color of the labels can be defined statically or dynamically setting **LabelColor** to :

- PIE_LABEL_COLOR_MANUAL to manually provide a single color (LabelR,...)
- PIE_LABEL_COLOR_AUTO to re-use the color of the data series.

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pPie.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(40,60,15,10,6,4),"ScoreA");
$MyData->setSerieDescription("ScoreA","Application A");

/* Define the absissa serie */
$MyData->addPoints(array("<10","10<>20","20<>40","40<>60","60<>80",">80"),"Labels");
$MyData->setAbscissa("Labels");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw a solid background */
$Settings = array("R"=>173, "G"=>152, "B"=>217, "Dash"=>1, "DashR"=>193, "DashG"=>172, "DashB"=>237);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Draw a gradient overlay */
$Settings = array("StartR"=>209, "StartG"=>150, "StartB"=>231, "EndR"=>111, "EndG"=>3, "EndB"=>138, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>100));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pPie - Draw 2D pie charts",array("R"=>255,"G"=>255,"B"=>255));
```

```
/* Set the default font properties */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>10,"R"=>80,"G"=>80,"B"=>80));

/* Enable shadow computing */
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>150,"G"=>150,"B"=>150,"Alpha"=>100));

/* Create the pPie object */
$PieChart = new pPie($myPicture,$MyData);

/* Draw a simple pie chart */
$PieChart->draw2DPie(140,125,array("SecondPass"=>FALSE));

/* Draw an AA pie chart */
$PieChart->draw2DPie(340,125,array("DrawLabels"=>TRUE,"Border"=>TRUE));

/* Draw a splitted pie chart */
$PieChart->draw2DPie($myPicture,$MyData,540,125,array("DataGapAngle"=>10,"DataGapRadius"=>6,"Border"=>TRUE,"BorderR"=>255,"BorderG"=>255,"BorderB"=>255));

/* Write the legend */
$myPicture->setFontProperties(array("FontName"=>"../fonts/MankSans.ttf","FontSize"=>11));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));
$myPicture->drawText(140,200,"Single AA pass",array("R"=>0,"G"=>0,"B"=>0,"Align"=>TEXT_ALIGN_TOPMIDDLE));
$myPicture->drawText(540,200,"Extended AA pass / Splitted",array("R"=>0,"G"=>0,"B"=>0,"Align"=>TEXT_ALIGN_TOPMIDDLE));

/* Render the picture to a file */
$myPicture->Render("example.draw2DPie.png");
```

# draw3DPie - Draw 3D pie charts

This function allows you to draw a 3D pie chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

Information

In exploded 3D pie charts some visual artifacts may appears, this will be fixed in the next releases.

## Calling this function

```
draw3DPie($X,$Y,$Format);
```

Where :

- X,Y is the center coordinate of the pie. (Radius can be adjusted in the Format array)
- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

It is possible to customize the pie chart rendering by playing with this array. Providing a detailled configuration is not mandatory. You‘ll see below a representation of all the customization possible :

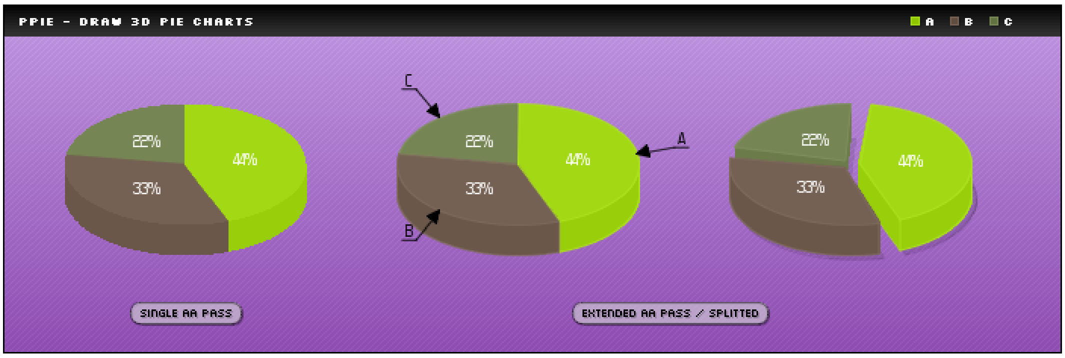
- You can specify if the radius of the pie chart with Radius.
- You can specify the skew factor with SkewFactor (>0.5 <=1).
- You can specify the height of each sloces with SliceHeight.
- You can specify the angle in degree between two values with DataGapAngle.

- If your specify a gap, you can set the inner radius withDataGapRadius.
- You can enable a 2nd AA pass setting SecondPass to TRUE.
- You can write the data labels setting DrawLabels to TRUE.
- The labels colors can be set with LabelR,LabelG,LabelB,LabelAlpha.

The color of the labels can be defined statically or dynamically setting **LabelColor** to :

- PIE_LABEL_COLOR_MANUAL to manually provide a single color (LabelR,...)
- PIE_LABEL_COLOR_AUTO to re-use the color of the data series.

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class&.phpquot;);
include("../class/pPie.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(40,30,20),"ScoreA");
$MyData->setSerieDescription("ScoreA","Application A");

/* Define the absissa serie */
$MyData->addPoints(array("A","B","C"),"Labels");
$MyData->setAbscissa("Labels");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw a solid background */
$Settings = array("R"=>173, "G"=>152, "B"=>217, "Dash"=>1, "DashR"=>193, "DashG"=>172, "DashB"=>237);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Draw a gradient overlay */
$Settings = array("StartR"=>209, "StartG"=>150, "StartB"=>231, "EndR"=>111, "EndG"=>3, "EndB"=>138, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>100));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));
```

```
/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pPie - Draw 3D pie charts",array("R"=>255,"G"=>255,"B"=>255));

/* Set the default font properties */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>10,"R"=>80,"G"=>80,"B"=>80));

/* Create the pPie object */
$PieChart = new pPie($myPicture,$MyData);

/* Define the slice color */
$PieChart->setSliceColor(0,array("R"=>143,"G"=>197,"B"=>0));
$PieChart->setSliceColor(1,array("R"=>97,"G"=>177,"B"=>63));
$PieChart->setSliceColor(2,array("R"=>97,"G"=>113,"B"=>63));

/* Draw a simple pie chart */
$PieChart->draw3DPie(140,125,array("SecondPass"=>FALSE));

/* Draw an AA pie chart */
$PieChart->draw3DPie(340,125,array("DrawLabels"=>TRUE,"Border"=>TRUE));

/* Enable shadow computing */
$myPicture->setShadow(TRUE,array("X"=>3,"Y"=>3,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Draw a splitted pie chart */
$PieChart->draw3DPie(540,125,array("DataGapAngle"=>10,"DataGapRadius"=>6,"Border"=>TRUE));

/* Write the legend */
$myPicture->setFontProperties(array("FontName"=>"../fonts/MankSans.ttf","FontSize"=>11));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));
$myPicture->drawText(140,200,"Single AA pass",array("R"=>0,"G"=>0,"B"=>0,"Align"=>TEXT_ALIGN_TOPMIDDLE));
$myPicture->drawText(440,200,"Extended AA pass / Splitted",array("R"=>0,"G"=>0,"B"=>0,"Align"=>TEXT_ALIGN_TOPMIDDLE));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.draw3DPie.png");
```

# Radar & Polar charts

# drawRadar - Draw radar charts

This function allows you to draw a radar chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide. Radar charts have been completly relooked since pChart 1.x to take full advantage of the new anti-aliasing, transparency & shadow engine. Best is to keep the scale computing to automatic (SEGMENT_HEIGHT_AUTO) to have the most eye candy rendering.

## Calling this function

```
drawRadar($Object,$Values,$Format);
```

Where :

- Object is reference to a pImage object.
- Values is reference to a pData object.
- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

It is possible to customize the radar chart rendering by playing with this array. Providing a detailled configuration is not mandatory. You‘ll see below a representation of all the customization possible :

- You can specify if the axis color with AxisR, AxisG, AxisB, AxisAlpha, .
- You can specify the axis rotation in degree using AxisRotation, .
- You can draw ticks outside of the radard setting DrawTicks to TRUE.
- You can define the ticks length with TicksLength (default is 2px)
- You can write the axis names setting DrawAxisValues to TRUE.
- You can specify the position of the axis names with LabelPos (see below).
- You can specify the label paddings with LabelPadding (default is 4px).
- You can draw a background setting DrawBackground to TRUE.
- You can specify the background color with BackgroundR, BackgroundG, BackgroundB, BackgroundAlpha.
- You can specify a background gradient with BackgroundGradient see the details below .
- You can define the radar layout with Layout see the details below .
- You can specify the height of one segment manually with SegmentHeight (default is SEGMENT_HEIGHT_AUTO for automatic computation)
- You can specify the number of segments manually with Segments (this may be overwritten if you use SEGMENT_HEIGHT_AUTO)
- You can write the segments values setting WriteLabels to TRUE.
- You can skip any number of labels playing with SkipLabels.
- You can center the label in the slice setting LabelMiddle to TRUE.
- You can specify if the segment values should have background setting LabelsBackground to TRUE.
- You can specify the segments values background color with LabelsBGR, LabelsBGG, LabelsBGB, LabelsBGAlpha.
- You can specify if you want to draw a circle for each data point setting DrawPoints to TRUE.
- You can specify the radius of the circles with PointRadius. (default is 4)
- You can specify a surrounding color around the circles with PointSurrounding. (default is -30)
- You can specify if you want to draw lines between each data point setting DrawLines to TRUE.
- You define if lines should loop to the 1st data point setting LineLoopStart to TRUE.
- You can specify if you want to draw polygons with your data points setting DrawPoly to TRUE.
- You can overide the polygons transparency with PolyAlpha.

The Layout (Layout) can be :

- RADAR_LAYOUT_STAR, this will draw star like axis.
- RADAR_LAYOUT_CIRCLE, this will draw concentric circles axis.

The labels positions (LabelPos) can be :

- RADAR_LABELS_ROTATED, this will written with an angle depending of the axis position.

- RADAR_LABELS_HORIZONTAL, labels will be written horizontally.

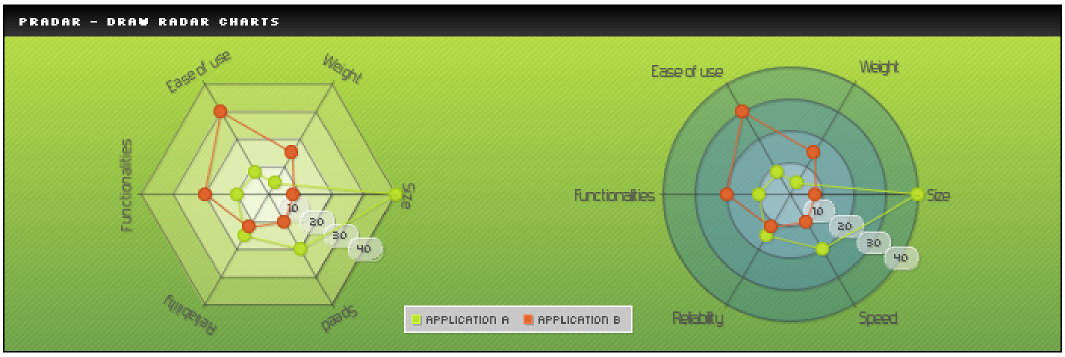
Gradients are defined with an array containing 8 values :

- The starting color defined by StartR, StartG, StartB, StartAlpha.
- The ending color defined by EndR, EndG, EndB, EndAlpha.

Code sample

```
$Gradient = array("StartR"=>255,"StartG"=>255,"StartB"=>255,"StartAlpha"=>50,"EndR"=>32,"EndG"=>109,"EndB"=>174,"EndAlpha"=>30)
```

## Sample script



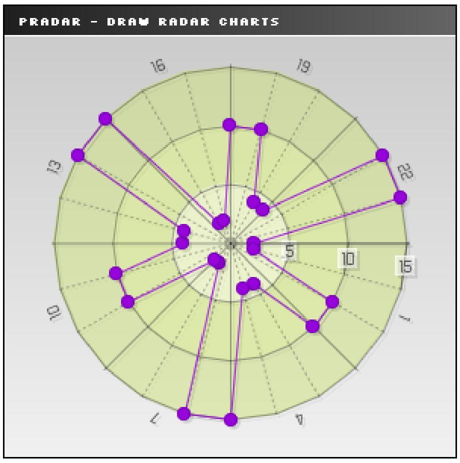
Code sample

```
 /* pChart library inclusions */
include("class/pData.class.php");
include("class/pDraw.class.php");
include("class/pRadar.class.php");
include("class/pImage.class.php");

/* Prepare some nice data & axis config */
$MyData = new pData();
$MyData->addPoints(array(40,20,15,10,8,4),"ScoreA");
$MyData->addPoints(array(8,10,12,20,30,15),"ScoreB");
$MyData->setSerieDescription("ScoreA","Application A");
$MyData->setSerieDescription("ScoreB","Application B");


/* Create the X serie */
$MyData->addPoints(array("Size","Speed","Reliability","Functionalities","Ease of use","Weight"),"Labels");
$MyData->setAbscissa("Labels");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw a solid background */
$Settings = array("R"=>179, "G"=>217, "B"=>91, "Dash"=>1, "DashR"=>199, "DashG"=>237, "DashB"=>111);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay some gradient areas */
$Settings = array("StartR"=>194, "StartG"=>231, "StartB"=>44, "EndR"=>43, "EndG"=>107, "EndB"=>58, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);



$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>100));
```

```
 /* Draw the border */
 $myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

 /* Write the title */
 $myPicture->setFontProperties(array("FontName"=>"fonts/Silkscreen.ttf","FontSize"=>6));
 $myPicture->drawText(10,13,"pRadar - Draw radar charts",array("R"=>255,"G"=>255,"B"=>255));

 /* Define general drawing parameters */
 $myPicture->setFontProperties(array("FontName"=>"fonts/Forgotte.ttf","FontSize"=>10,"R"=>80,"G"=>80,"B"=>80));
 $myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

 /* Create the radar object */
 $SplitChart = new pRadar();

 /* Draw the 1st radar chart */
 $myPicture->setGraphArea(10,25,340,225);

 $Options =


array("Layout"=>RADAR_LAYOUT_STAR,"BackgroundGradient"=>array("StartR"=>255,"StartG"=>255,"StartB"=>255,"StartAlpha"=>100,"EndR"=>207,"


EndG"=>227,"EndB"=>125,"EndAlpha"=>50));

 $SplitChart->drawRadar($myPicture,$MyData,$Options);

 /* Draw the 2nd radar chart */
 $myPicture->setGraphArea(350,25,690,225);

 $Options = array("Layout"=>RADAR_LAYOUT_CIRCLE, "LabelPos"=>RADAR_LABELS_HORIZONTAL,


"BackgroundGradient"=>array("StartR"=>255,"StartG"=>255,"StartB"=>255,"StartAlpha"=>50,"EndR"=>32,"EndG"=>109,"EndB"=>174,"EndAlpha"=>30))


;

 $SplitChart->drawRadar($myPicture,$MyData,$Options);

 /* Write down the legend */
 $myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
 $myPicture->drawLegend(270,205,array("Style"=>LEGEND_BOX,"Mode"=>LEGEND_HORIZONTAL));

 /* Render the picture */
 $myPicture->Render("drawradar.png");
```

## Sample script - Playing with labels

In this example we only print one label every 3 divisions (using the SkipLabels parameter). We‘ve also choosen to center the label inside each main slice parts (setting LabelMiddle to TRUE).



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pRadar.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(2,2,10,10,4,4,15,15,2,2,10,10,4,4,15,15,2,2,10,10,4,4,15,15),"ScoreA");
$MyData->setSerieDescription("ScoreA","Application A");
$MyData->setPalette("ScoreA",array("R"=>150,"G"=>5,"B"=>217));

/* Define the absissa serie */
$MyData->addPoints(array(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24),"Time");
$MyData->setAbscissa("Time");

/* Create the pChart object */
$myPicture = new pImage(300,300,$MyData);
$myPicture->drawGradientArea(0,0,300,300,DIRECTION_VERTICAL,array("StartR"=>200,"StartG"=>200,"StartB"=>200,

"EndR"=>240,"EndG"=>240,"EndB"=>240,"Alpha"=>100));

$myPicture->drawGradientArea(0,0,300,20,DIRECTION_HORIZONTAL,array("StartR"=>30,"StartG"=>30,"StartB"=>30,

"EndR"=>100,"EndG"=>100,"EndB"=>100,"Alpha"=>100));
$myPicture->drawLine(0,20,300,20,array("R"=>255,"G"=>255,"B"=>255));
$RectangleSettings = array("R"=>180,"G"=>180,"B"=>180,"Alpha"=>100);

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,299,299,array("R"=>0,"G"=>0,"B"=>0));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"pRadar - Draw radar charts",array("R"=>255,"G"=>255,"B"=>255));

/* Set the default font properties */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>10,"R"=>80,"G"=>80,"B"=>80));

/* Enable shadow computing */
$myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Create the pRadar object */
$SplitChart = new pRadar();

/* Draw a radar chart */
$myPicture->setGraphArea(10,25,290,290);
```

```
$Options = array("SkipLabels"=>3,"LabelMiddle"=>TRUE,"Layout"=>RADAR_LAYOUT_STAR,"BackgroundGradient"=>array("StartR"=>255,

"StartG"=>255,"StartB"=>255,"StartAlpha"=>100,"EndR"=>207,"EndG"=>227,"EndB"=>125, "EndAlpha"=>50));
$SplitChart->drawRadar($myPicture,$MyData,$Options);

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.radar.labels.png");
```

# drawPolar - Draw polar charts

This function allows you to draw a polar chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide. Polar charts were not existing in pChart 1.x. Best is to keep the scale computing to automatic (SEGMENT_HEIGHT_AUTO) to have the most eye candy rendering.

Information

Polar charts share many parameters with the radar charts. The main difference is that the X axis is fixed to 0-360, data is given using the common way with the pData class, X values of the associated points are given through the abscissa serie which is then mandatory.

## Calling this function

```
drawPolar($Object,$Values,$Format);
```

Where :

- Object is reference to a pImage object.
- Values is reference to a pData object.
- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

It is possible to customize the polar chart rendering by playing with this array. Providing a detailled configuration is not mandatory. You‘ll see below a representation of all the customization possible :

- You can specify if the axis color with AxisR, AxisG, AxisB, AxisAlpha, .
- You can specify the axis rotation in degree using AxisRotation.
- You can specify the step in degree between two axis lines with AxisSteps (default is 20°) .
- You can draw ticks outside of the radard setting DrawTicks to TRUE.
- You can define the ticks length with TicksLength (default is 2px)
- You can write the axis names setting DrawAxisValues to TRUE.
- You can specify the position of the axis names with LabelPos (see below).
- You can specify the label paddings with LabelPadding (default is 4px).
- You can draw a background setting DrawBackground to TRUE.
- You can specify the background color with BackgroundR, BackgroundG, BackgroundB, BackgroundAlpha.
- You can specify a background gradient with BackgroundGradient see the details below .
- You can define the radar layout with Layout see the details below .
- You can specify the height of one segment manually with SegmentHeight (default is SEGMENT_HEIGHT_AUTO for automatic computation)
- You can specify the number of segments manually with Segments (this may be overwritten if you use SEGMENT_HEIGHT_AUTO)
- You can write the segments values setting WriteLabels to TRUE.
- You can specify if the segment values should have background setting LabelsBackground to TRUE.
- You can specify the segments values background color with LabelsBGR, LabelsBGG, LabelsBGB, LabelsBGAlpha.
- You can specify if you want to draw a circle for each data point setting DrawPoints to TRUE.
- You can specify the radius of the circles with PointRadius. (default is 4)
- You can specify a surrounding color around the circles with PointSurrounding. (default is -30)
- You can specify if you want to draw lines between each data point setting DrawLines to TRUE.
- You define if lines should loop to the 1st data point setting LineLoopStart to TRUE.

- You can specify if you want to draw polygons with your data points setting DrawPoly to TRUE.
- You can overide the polygons transparency with PolyAlpha.

The Layout (Layout) can be :

- RADAR_LAYOUT_STAR, this will draw star like axis.
- RADAR_LAYOUT_CIRCLE, this will draw concentric circles axis.

The labels positions (LabelPos) can be :

- RADAR_LABELS_ROTATED, this will written with an angle depending of the axis position.
- RADAR_LABELS_HORIZONTAL, labels will be written horizontally.

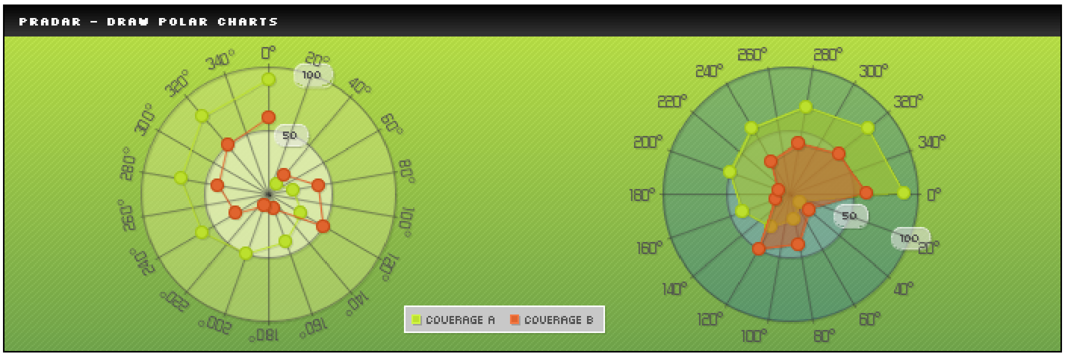
Gradients are defined with an array containing 8 values :

- The starting color defined by StartR, StartG, StartB, StartAlpha.
- The ending color defined by EndR, EndG, EndB, EndAlpha.

Code sample

```
$Gradient = array("StartR"=>255,"StartG"=>255,"StartB"=>255,"StartAlpha"=>50,"EndR"=>32,"EndG"=>109,"EndB"=>174,"EndAlpha"=>30)
```

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pRadar.class.php");
include("../class/pImage.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(10,20,30,40,50,60,70,80,90),"ScoreA");
$MyData->addPoints(array(20,40,50,12,10,30,40,50,60),"ScoreB");
$MyData->setSerieDescription("ScoreA","Coverage A");
$MyData->setSerieDescription("ScoreB","Coverage B");

/* Define the absissa serie */
$MyData->addPoints(array(40,80,120,160,200,240,280,320,360),"Coord");
$MyData->setAbscissa("Coord");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw a solid background */
$Settings = array("R"=>179, "G"=>217, "B"=>91, "Dash"=>1, "DashR"=>199, "DashG"=>237, "DashB"=>111);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay some gradient areas */
$Settings = array("StartR"=>194, "StartG"=>231, "StartB"=>44, "EndR"=>43, "EndG"=>107, "EndB"=>58, "Alpha"=>50);
```

```
 $myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);



$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>100));

 /* Add a border to the picture */
 $myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

 /* Write the picture title */
 $myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
 $myPicture->drawText(10,13,"pRadar - Draw polar charts",array("R"=>255,"G"=>255,"B"=>255));

 /* Set the default font properties */
 $myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>10,"R"=>80,"G"=>80,"B"=>80));

 /* Enable shadow computing */
 $myPicture->setShadow(TRUE,array("X"=>2,"Y"=>2,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

 /* Create the pRadar object */
 $SplitChart = new pRadar();

 /* Draw a polar chart */
 $myPicture->setGraphArea(10,25,340,225);

 $Options =

array("BackgroundGradient"=>array("StartR"=>255,"StartG"=>255,"StartB"=>255,"StartAlpha"=>100,"EndR"=>207,"EndG"=>227,"EndB"=>125,"EndAlph

a"=>50));

 $SplitChart->drawPolar($myPicture,$MyData,$Options);

 /* Draw a polar chart */
 $myPicture->setGraphArea(350,25,690,225);
 $Options = array("LabelPos"=>RADAR_LABELS_HORIZONTAL,"BackgroundGradient"=>array("StartR"=>255,"StartG"=>255,"StartB"=>255,

"StartAlpha"=>50,"EndR"=>32,"EndG"=>109,"EndB"=>174,"EndAlpha"=>30),"AxisRotation"=>0,"DrawPoly"=>TRUE, "PolyAlpha"=>50);
 $SplitChart->drawPolar($myPicture,$MyData,$Options);

 /* Write the chart legend */
 $myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));
 $myPicture->drawLegend(270,205,array("Style"=>LEGEND_BOX,"Mode"=>LEGEND_HORIZONTAL));

 /* Render the picture (choose the best way) */
 $myPicture->autoOutput("pictures/example.polar.png");
```

# Stock charts

# Stock charts basics

*Require pChart 2.0.11*

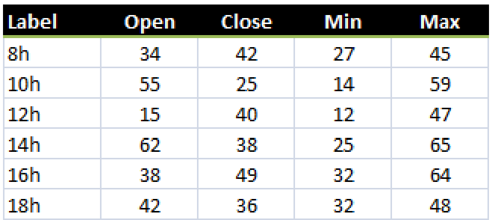
The pStock extension allows you to create stock charts. As one data point is characterized by 4 values, you‘ll have to use your datasets in a different way than when you‘re drawing standard charts. Each data points is defined by the following states :

- The Max value as a the 1st boundary.
- The Min value as a 2nd boundary.
- The Open value.
- The Close value.

All this values have to be listed in dedicated datasets. You can also use an extra dataset to put your X axis labels.

## Exemple

Let‘s assume that we got the data represented in the table bellow :



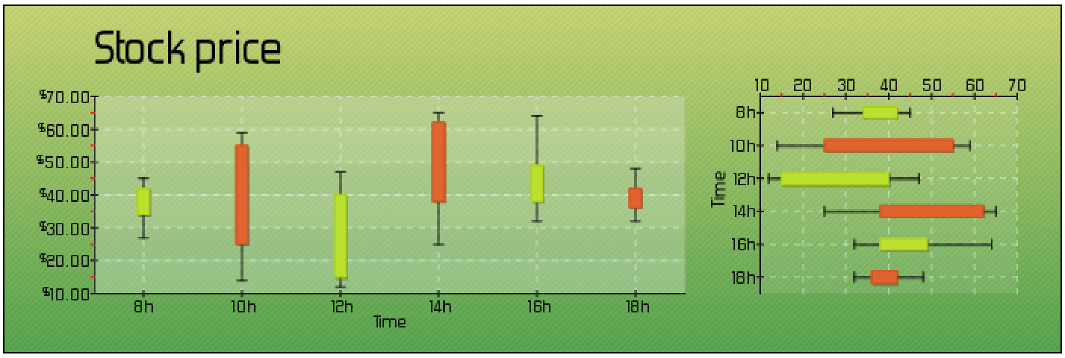
| Label | Open | Close | Min | Max |
|---|---|---|---|---|
| 8h | 34 | 42 | 27 | 45 |
| 10h | 55 | 25 | 14 | 59 |
| 12h | 15 | 40 | 12 | 47 |
| 14h | 62 | 38 | 25 | 65 |
| 16h | 38 | 49 | 32 | 64 |
| 18h | 42 | 36 | 32 | 48 |

This will lead to the following declaration with the pData object structure :

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Populate the data series */
$MyData->addPoints(array(34,55,15,62,38,42),"Open");
$MyData->addPoints(array(42,25,40,38,49,36),"Close");
$MyData->addPoints(array(27,14,12,25,32,32),"Min");
$MyData->addPoints(array(45,59,47,65,64,48),"Max");

/* Populate the X lables serie */
$MyData->addPoints(array("8h","10h","12h","14h","16h","18h"),"Time");
$MyData->setAbscissa("Time");
```

That can be charted this way :



# drawStockChart - Draw a stock chart

*Require pChart 2.0.11*

This function allows you to draw a stock chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
drawStockChart($Format);
```

Where :

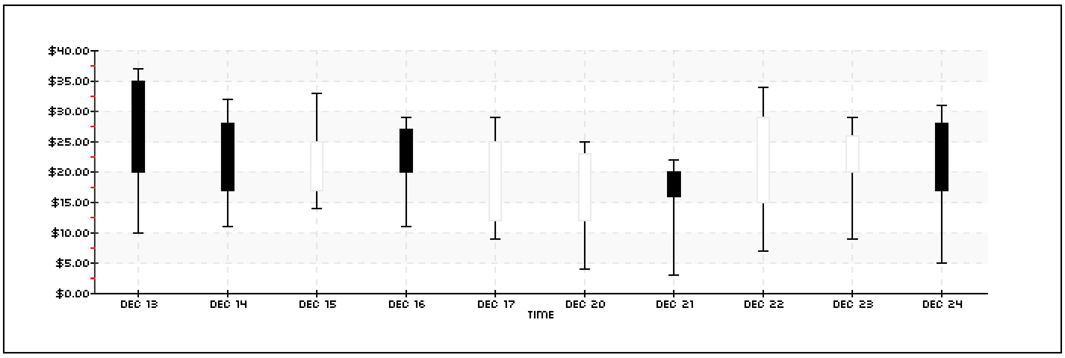
- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

It is possible to customize the stock chart rendering by playing with this array. Providing a detailled configuration is not mandatory. You'll see below a representation of all the customization possible :

- You can specify the name of the serie containing the "Open" value SerieOpen (default is Open).
- You can specify the name of the serie containing the "Close" value SerieClose (default is Close).
- You can specify the name of the serie containing the "Max" value SerieMax (default is Max).
- You can specify the name of the serie containing the "Min" value SerieMin (default is Min).
- You can specify the width of the vertical segment with LineWidth.
- You can specify the color of the vertical segment with LineR, LineG, LineB, LineAlpha.
- You can specify the extremity segments width with ExtremityWidth.
- You can specify the extremity segments length with ExtremityLength.
- You can specify the color of the extremity segments with ExtremityR, ExtremityG, ExtremityB, ExtremityAlpha.
- You can specify the width of the boxes with BoxWidth.
- You can specify the color of the positives boxes with BoxUpR, BoxUpG, BoxUpB, BoxUpAlpha, .
- You can specify a surrounding factor for the positives boxes borders BoxUpSurrounding.
- You can specify the color of the positives boxes border with BoxUpBorderR, BoxUpBorderG, BoxUpBorderB, BoxUpBorderAlpha .
- You can specify the color of the negatives boxes with BoxDownR, BoxDownG, BoxDownB, BoxDownAlpha, .
- You can specify a surrounding factor for the negatives boxes borders BoxDownSurrounding.
- You can specify the color of the negativesboxes border with BoxDownBorderR, BoxDownBorderG, BoxDownBorderB, BoxDownBorderAlpha.
- You can choose to enable shadow computing only on the boxes setting ShadowOnBoxesOnly to TRUE.

## Sample script #1

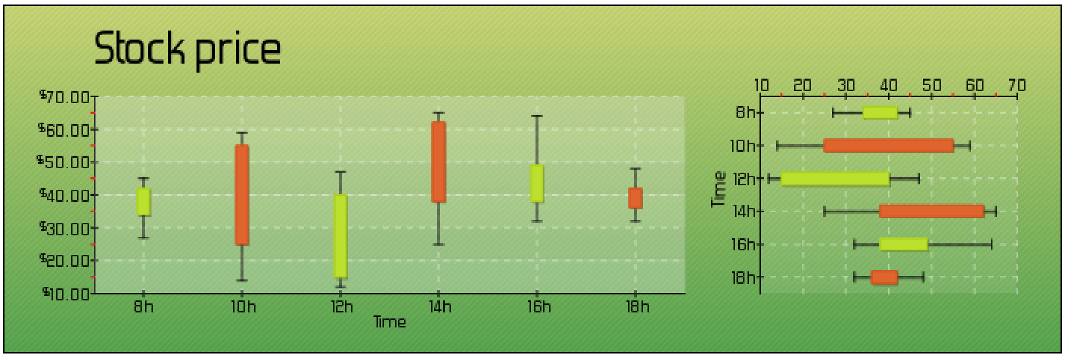


Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
include("../class/pStock.class.php");

/* Create and populate the pData object */
```

```
$MyData = new pData();
$MyData->addPoints(array(35,28,17,27,12,12,20,15,20,28),"Open");
$MyData->addPoints(array(20,17,25,20,25,23,16,29,26,17),"Close");
$MyData->addPoints(array(10,11,14,11,9,4,3,7,9,5),"Min");
$MyData->addPoints(array(37,32,33,29,29,25,22,34,29,31),"Max");
$MyData->setAxisDisplay(0,AXIS_FORMAT_CURRENCY,"$");

$MyData->addPoints(array("Dec 13","Dec 14","Dec 15","Dec 16","Dec 17", "Dec 20","Dec 21","Dec 22","Dec 23","Dec 24"),"Time");
$MyData->setAbscissa("Time");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Turn of AAliasing */
$myPicture->Antialias = FALSE;

/* Draw the border */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Set the default font settings */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

/* Define the chart area */
$myPicture->setGraphArea(60,30,650,190);

/* Draw the scale */
$scaleSettings = array("GridR"=>200,"GridG"=>200,"GridB"=>200,"DrawSubTicks"=>TRUE,"CycleBackground"=>TRUE);
$myPicture->drawScale($scaleSettings);

/* Create the pStock object */
$mystockChart = new pStock($myPicture,$MyData);

/* Draw the stock chart */
$stockSettings = array("BoxUpR"=>255,"BoxUpG"=>255,"BoxUpB"=>255,"BoxDownR"=>0,"BoxDownG"=>0,"BoxDownB"=>0);
$mystockChart->drawStockChart($stockSettings);

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawStockChart.simple.png");
```

## Sample script #2



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
include("../class/pStock.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(34,55,15,62,38,42),"Open");
```

```
$MyData->addPoints(array(42,25,40,38,49,36),"Close");
$MyData->addPoints(array(27,14,12,25,32,32),"Min");
$MyData->addPoints(array(45,59,47,65,64,48),"Max");
$MyData->setAxisDisplay(0,AXIS_FORMAT_CURRENCY,"$");
$MyData->addPoints(array("8h","10h","12h","14h","16h","18h"),"Time");
$MyData->setAbscissa("Time");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);

/* Draw the border */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Write the title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));
$myPicture->drawText(60,45,"Stock price",array("FontSize"=>28,"Align"=>TEXT_ALIGN_BOTTOMLEFT));

/* Draw the 1st scale */
$myPicture->setGraphArea(60,60,450,190);
$myPicture->drawFilledRectangle(60,60,450,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE,"CycleBackground"=>TRUE));

/* Draw the 1st stock chart */
$mystockChart = new pStock($myPicture,$MyData);
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>30));
$mystockChart->drawStockChart();

/* Reset the display mode because of the graph small size */
$MyData->setAxisDisplay(0,AXIS_FORMAT_DEFAULT);

/* Draw the 2nd scale */
$myPicture->setShadow(FALSE);
$myPicture->setGraphArea(500,60,670,190);
$myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"DrawSubTicks"=>TRUE));

/* Draw the 2nd stock chart */
$mystockChart = new pStock($myPicture,$MyData);
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>30));
$mystockChart->drawStockChart();

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawStockChart.png");
```

# Linear Bubble chart

# Linear bubble charts basics

*Require pChart 2.0.13*

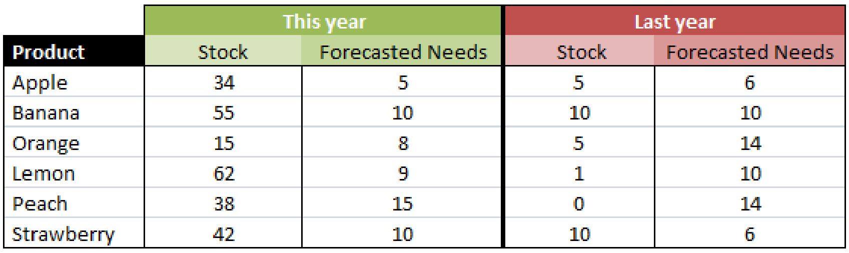
Linear bubble charts is a convenient way to draw data that got two parameters. In this approach, the 1st parameter is the Y axis value and the second one a value representing something link to this point. We can imagine the association "items in stock" / "items ordered" that would reflect the current status of your stock an it's forecast growth.

Information

This two values have to be listed in dedicated datasets.

## Exemple

Let's assume that we got the data represented in the table bellow :



| | This year | | Last year | |
|---|---|---|---|---|
| **Product** | Stock | Forecasted Needs | Stock | Forecasted Needs |
| Apple | 34 | 5 | 5 | 6 |
| Banana | 55 | 10 | 10 | 10 |
| Orange | 15 | 8 | 5 | 14 |
| Lemon | 62 | 9 | 1 | 10 |
| Peach | 38 | 15 | 0 | 14 |
| Strawberry | 42 | 10 | 10 | 6 |

This will lead to the following declaration with the pData object structure :

Code sample

```
/* Create the pData object */
$MyData = new pData();

/* Provide this year data */
$MyData->addPoints(array(34,55,15,62,38,42),"ThisYear");
$MyData->addPoints(array(5,10,8,9,15,10),"ThisYearNeeds");
$MyData->setSerieDescription("ThisYear","This year");

/* Provide last year data */
$MyData->addPoints(array(5,10,5,1,0,10),"LastYear");
$MyData->addPoints(array(6,10,14,10,14,6),"LastYearNeeds");
$MyData->setSerieDescription("LastYear","Last year");

/* Give a name to the X axis */
$MyData->setAxisName(0,"Current stock");

$MyData->addPoints(array("Apple","Banana","Orange","Lemon","Peach","Strawberry"),"Product");
$MyData->setAbscissa("Product");
```

You'll then have to update your scale before calling the drawScale() function :

Code sample

```
/* Scale up for the bubble chart */
$bubbleDataSeries   = array("thisYear","LastYear");
$bubbleWeightSeries = array("ThisYearNeeds","LastYearNeeds");
$myBubbleChart->bubbleScale($bubbleDataSeries,$bubbleWeightSeries);
```

Finally, when the scale is drawn, just call the drawBubbleChart() function with the same arrays :

Code sample

```
$myBubbleChart->drawBubbleChart($bubbleDataSeries,$bubbleWeightSeries);
```

# bubbleScale - Correct the chart scale for bubble charts

*Require pChart 2.0.13*

A the bubble chart add extra boundaries to your charts, the scale must be updated accordingly. Calling this function will fix the chart scale Min / Max so the bubble will never go outside the borders of the charting area.

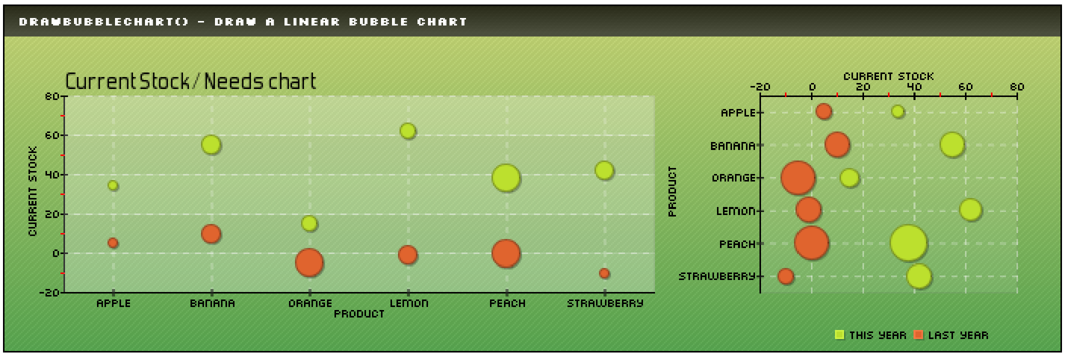
## Calling this function

```
bubbleScale($DataSeries,$WeightSeries);
```

Where :

- DataSeries is an array containing the name of the serie providing the Y values.
- WeightSeries is an array containing the name of the serie providing the Z values.

## Sample script #1



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class&.phpquot;);
include("../class/pBubble.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(34,55,15,62,38,42),"Probe1");
$MyData->addPoints(array(5,10,8,9,15,10),"Probe1Weight");
$MyData->addPoints(array(5,10,-5,-1,0,-10),"Probe2");
$MyData->addPoints(array(6,10,14,10,14,6),"Probe2Weight");
$MyData->setSerieDescription("Probe1","This year");
$MyData->setSerieDescription("Probe2","Last year");
$MyData->setAxisName(0,"Current stock");
$MyData->addPoints(array("Apple","Banana","Orange","Lemon","Peach","Strawberry"),"Product");
$MyData->setAbscissa("Product");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);
```

```
$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>80));

 /* Add a border to the picture */
 $myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));


 /* Write the picture title */
 $myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
 $myPicture->drawText(10,13,"drawBubbleChart() - draw a linear bubble chart",array("R"=>255,"G"=>255,"B"=>255));

 /* Write the title */
 $myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));
 $myPicture->drawText(40,55,"Current Stock / Needs chart",array("FontSize"=>14,"Align"=>TEXT_ALIGN_BOTTOMLEFT));

 /* Change the default font */
 $myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

 /* Create the Bubble chart object and scale up */
 $myBubbleChart = new pBubble($myPicture,$MyData);

 /* Scale up for the bubble chart */
 $bubbleDataSeries   = array("Probe1","Probe2");
 $bubbleWeightSeries = array("Probe1Weight","Probe2Weight");
 $myBubbleChart->bubbleScale($bubbleDataSeries,$bubbleWeightSeries);

 /* Draw the 1st chart */
 $myPicture->setGraphArea(40,60,430,190);
 $myPicture->drawFilledRectangle(40,60,430,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
 $myPicture->drawScale(array("DrawSubTicks"=>TRUE,"CycleBackground"=>TRUE));
 $myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>30));
 $myBubbleChart->drawBubbleChart($bubbleDataSeries,$bubbleWeightSeries);

 /* Draw the 2nd scale */
 $myPicture->setShadow(FALSE);
 $myPicture->setGraphArea(500,60,670,190);
 $myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
 $myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"DrawSubTicks"=>TRUE));

 /* Draw the 2nd stock chart */
 $myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>30));
 $myBubbleChart->drawbubbleChart($bubbleDataSeries,$bubbleWeightSeries);

 /* Write the chart legend */
 $myPicture->drawLegend(550,215,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

 /* Render the picture (choose the best way) */
 $myPicture->autoOutput("pictures/example.drawBubbleChart.png");
```

## drawBubbleChart - Draw a linear bubble chart

*Require pChart 2.0.13*

This function allows you to draw a bubble chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

## Calling this function

```
drawBubbleChart($DataSeries,$WeightSeries,$Format);
```

Where :

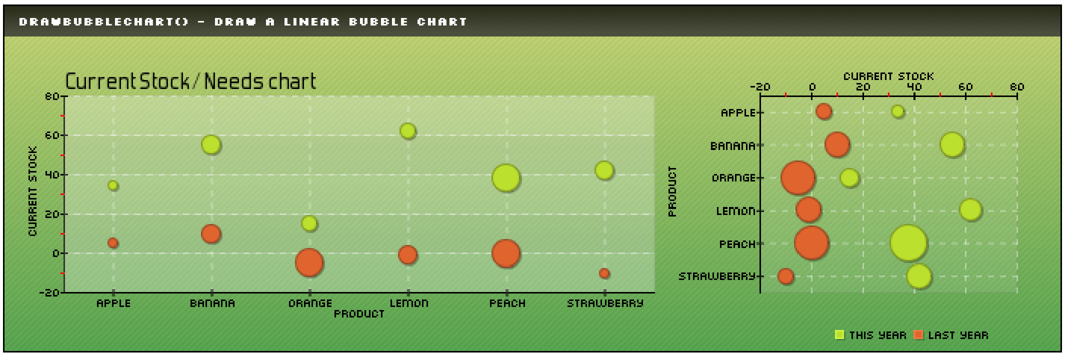
- DataSeries is an array containing the name of the serie providing the Y values.
- WeightSeries is an array containing the name of the serie providing the Z values.
- Format is an array containing the drawing parameters of the chart.

## Customisation array - Tune up your chart!

It is possible to customize the bubble chart rendering by playing with this array. Providing a detailled configuration is not mandatory. You'll see below a representation of all the customization possible :

- You can specify if a border will be drawn setting DrawBorder to TRUE.
- You can draw squares instead of circles setting DrawSquare to TRUE.
- You can specify the border color with BorderR, BorderG, BorderB.
- You can specify the border alpha transparency factor with BorderAlpha.
- You can specify a surrounding factor that will be added to the R,G,B values to compute the border color with Surrounding.

## Sample script #1

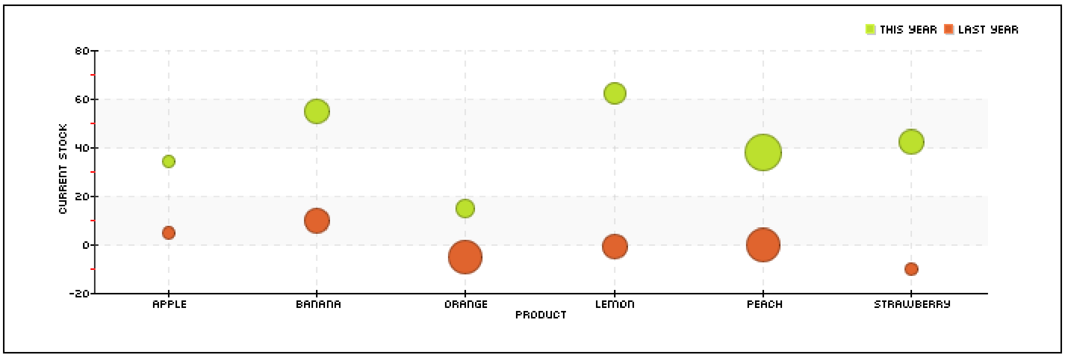


Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
include("../class/pBubble.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(34,55,15,62,38,42),"Probe1");
$MyData->addPoints(array(5,10,8,9,15,10),"Probe1Weight");
$MyData->addPoints(array(5,10,-5,-1,0,-10),"Probe2");
$MyData->addPoints(array(6,10,14,10,14,6),"Probe2Weight");
$MyData->setSerieDescription("Probe1","This year");
$MyData->setSerieDescription("Probe2","Last year");
$MyData->setAxisName(0,"Current stock");
$MyData->addPoints(array("Apple","Banana","Orange","Lemon","Peach","Strawberry"),"Product");
$MyData->setAbscissa("Product");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,700,230,$Settings);

/* Overlay with a gradient */
```

```
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,700,230,DIRECTION_VERTICAL,$Settings);



$myPicture->drawGradientArea(0,0,700,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>80));


/* Add a border to the picture */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));


/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawBubbleChart() - draw a linear bubble chart",array("R"=>255,"G"=>255,"B"=>255));

/* Write the title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Forgotte.ttf","FontSize"=>11));
$myPicture->drawText(40,55,"Current Stock / Needs chart",array("FontSize"=>14,"Align"=>TEXT_ALIGN_BOTTOMLEFT));

/* Change the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

/* Create the Bubble chart object and scale up */
$myBubbleChart = new pBubble($myPicture,$MyData);

/* Scale up for the bubble chart */
$bubbleDataSeries   = array("Probe1","Probe2");
$bubbleWeightSeries = array("Probe1Weight","Probe2Weight");
$myBubbleChart->bubbleScale($bubbleDataSeries,$bubbleWeightSeries);

/* Draw the 1st chart */
$myPicture->setGraphArea(40,60,430,190);
$myPicture->drawFilledRectangle(40,60,430,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("DrawSubTicks"=>TRUE,"CycleBackground"=>TRUE));
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>30));
$myBubbleChart->drawBubbleChart($bubbleDataSeries,$bubbleWeightSeries);

/* Draw the 2nd scale */
$myPicture->setShadow(FALSE);
$myPicture->setGraphArea(500,60,670,190);
$myPicture->drawFilledRectangle(500,60,670,190,array("R"=>255,"G"=>255,"B"=>255,"Surrounding"=>-200,"Alpha"=>10));
$myPicture->drawScale(array("Pos"=>SCALE_POS_TOPBOTTOM,"DrawSubTicks"=>TRUE));

/* Draw the 2nd stock chart */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>30));
$myBubbleChart->drawbubbleChart($bubbleDataSeries,$bubbleWeightSeries);

/* Write the chart legend */
$myPicture->drawLegend(550,215,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawBubbleChart.png");
```

## Sample script #2



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
include("../class/pBubble.class.php");

/* Create and populate the pData object */
$MyData = new pData();
$MyData->addPoints(array(34,55,15,62,38,42),"Probe1");
$MyData->addPoints(array(5,10,8,9,15,10),"Probe1Weight");
$MyData->addPoints(array(5,10,-5,-1,0,-10),"Probe2");
$MyData->addPoints(array(6,10,14,10,14,6),"Probe2Weight");
$MyData->setSerieDescription("Probe1","This year");
$MyData->setSerieDescription("Probe2","Last year");
$MyData->setAxisName(0,"Current stock");
$MyData->addPoints(array("Apple","Banana","Orange","Lemon","Peach","Strawberry"),"Product");
$MyData->setAbscissa("Product");

/* Create the pChart object */
$myPicture = new pImage(700,230,$MyData);

/* Turn of AAliasing */
$myPicture->Antialias = FALSE;

/* Draw the border */
$myPicture->drawRectangle(0,0,699,229,array("R"=>0,"G"=>0,"B"=>0));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

/* Define the chart area */
$myPicture->setGraphArea(60,30,650,190);

/* Draw the scale */
$scaleSettings = array("GridR"=>200,"GridG"=>200,"GridB"=>200,"DrawSubTicks"=>TRUE,"CycleBackground"=>TRUE);
$myPicture->drawScale($scaleSettings);

/* Create the Bubble chart object and scale up */
$myPicture->Antialias = TRUE;
$myBubbleChart = new pBubble($myPicture,$MyData);

/* Scale up for the bubble chart */
$bubbleDataSeries   = array("Probe1","Probe2");
$bubbleWeightSeries = array("Probe1Weight","Probe2Weight");
$myBubbleChart->bubbleScale($bubbleDataSeries,$bubbleWeightSeries);

/* Draw the bubble chart */
$myBubbleChart->drawBubbleChart($bubbleDataSeries,$bubbleWeightSeries);

/* Write the chart legend */
```

```
$myPicture->drawLegend(570,13,array("Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawBubbleChart.simple.png");
```

# Scatter chart

# drawScale - Compute & draw the scatter chart scale

This function will compute and draw the scale of your scatter charts. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide.

Be careful

This function must be called before any scatter charting functions.

## Calling this function

```
drawScatterScale($Format="");
```

Where :

- Format is an array containing the drawing parameters of the arrow.

## Customisation array - Tune up your scales!

It is possible to customize the way your scale will be rendered by playing with this array. Providing a detailled configuration is not mandatory, by default the axis will be drawn black in the Left / Right mode.

- The scale orientation can be set with Pos, valid values are SCALE_POS_LEFTRIGHT and SCALE_POS_TOPBOTTOM
- You can specify the rotation of the X Axis labels with LabelRotation.
- The minimum height of a scale div can be set using MinDivHeight.
- The scaling factors can be set using Factors.
- The X Axis margin can be set using XMargin.
- The Y Axis margin can be set using YMargin.
- The space between two scales on the same side can be set using ScaleSpacing.
- You can specify if you want to draw the X grid lines setting DrawXLines to TRUE or FALSE.
- You can specify if you want to draw the Y grid lines setting DrawYLines to ALL, NONE or to an array containing the axis IDs.
- You can draw dashed grid lines setting GridTicks to the width of the ticks.
- You can specify the grid color using GridR,GridG,GridB.
- You can specify the grid alpha factor using GridAlpha.
- You can specify the axis color using AxisR,AxisG,AxisB.
- You can specify the axis alpha factor using AxisAlpha.
- The inner ticks width can be set using InnerTickWidth.
- The outer ticks width can be set using OuterTickWidth.
- You can specify the ticks color using TickR,TickG,TickB.
- You can specify the ticks alpha factor using TickAlpha.
- If you want to draw the subticks of the Y axis set DrawSubTicks to TRUE.
- The inner subticks width can be set using InnerSubTickWidth.
- The outer subticks width can be set using OuterSubTickWidth.
- You can specify the subticks color using SubTickR,SubTickG,SubTickB.
- You can specify the subticks alpha factor using SubTickAlpha.
- You can specify if you want to draw the axis arrows setting DrawArrows to TRUE or FALSE.
- You can set the arrow size usingArrowSize.
- You can choose to display a 2-colors cycling background setting CycleBackground to TRUE.
- First background cycling color can be defined with BackgroundR1,BackgroundG1,BackgroundB1,BackgroundAlpha1.
- 2nd background cycling color can be defined with BackgroundR2,BackgroundG2,BackgroundB2,BackgroundAlpha2.

Depending on the chart you'll draw, you can define the way the scale will be computed with the **Mode** parameter :

- SCALE_MODE_FLOATING for min/max automatic scaling.
- SCALE_MODE_START0 for fixed min to 0.
- SCALE_MODE_ADDALL for stacked charts.
- SCALE_MODE_ADDALL_START0 for stacked charts with a fixed min to 0.

- SCALE_MODE_MANUAL to fix manually the min & max values.

If you choose the manual option, you‘ll have to provide an array with the **ManualScale** parameter to provide the min & max values per axis. This array should have the following structure :

Code sample

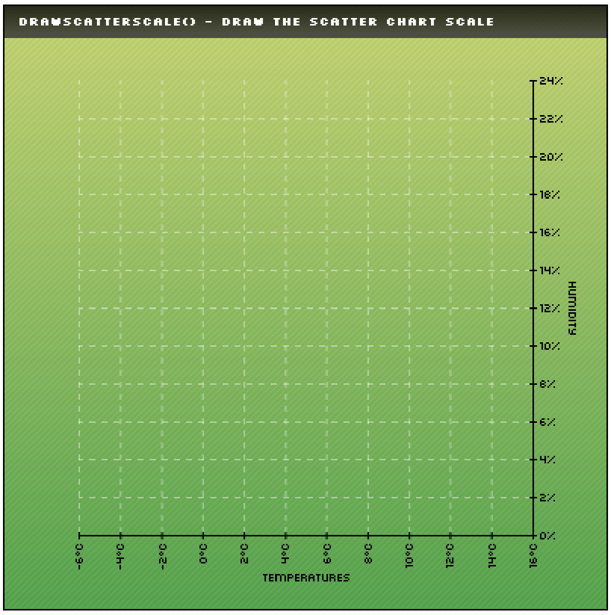
```
 $AxisBoundaries = array(0=>array("Min"=>0,"Max"=>100),1=>array("Min"=>10,"Max"=>20));
```

You can also provide the **Rows** and **RowHeight** parameters for each axis.

Information

This function may be a bit complex to understand, you will see more example within the charting functions section.

## Sample script #1



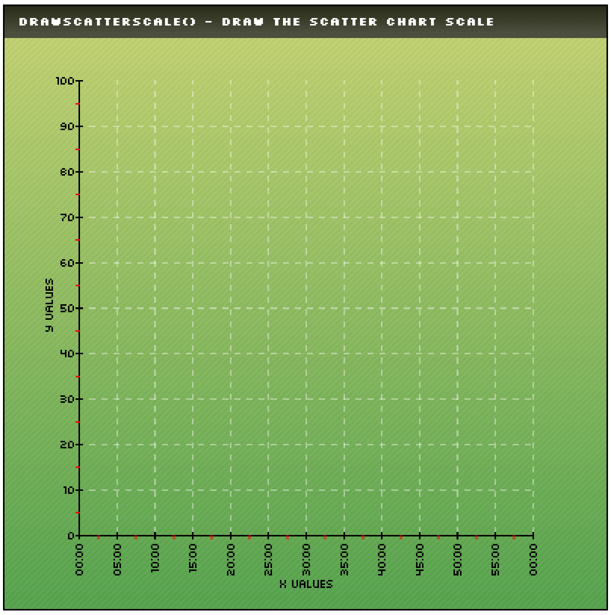
Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
include("../class/pScatter.class.php");

/* Create the pData object */
$myData = new pData();

/* Create the X axis and the binded series */
$myData->addPoints(array(-4,VOID,VOID,12,8,3),"Probe 1");
$myData->addPoints(array(3,12,15,8,5,-5),"Probe 2");
$myData->setAxisName(0,"Temperatures");
$myData->setAxisXY(0,AXIS_X);
$myData->setAxisUnit(0,"°C");
$myData->setAxisPosition(0,AXIS_POSITION_BOTTOM);

/* Create the Y axis and the binded series */
$myData->addPoints(array(2,7,5,18,19,22),"Probe 3");
$myData->setSerieOnAxis("Probe 3",1);
$myData->setAxisName(1,"Humidity");
$myData->setAxisXY(1,AXIS_Y);
$myData->setAxisUnit(1,"%");
$myData->setAxisPosition(1,AXIS_POSITION_RIGHT);

/* Create the pChart object */
$myPicture = new pImage(400,400,$myData);
```

```
/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
 $myPicture->drawFilledRectangle(0,0,400,400,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
 $myPicture->drawGradientArea(0,0,400,400,DIRECTION_VERTICAL,$Settings);
$myPicture->drawGradientArea(0,0,400,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>80));



/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
 $myPicture->drawText(10,13,"drawScatterScale() - Draw the scatter chart scale",array("R"=>255,"G"=>255,"B"=>255));


/* Add a border to the picture */
$myPicture->drawRectangle(0,0,399,399,array("R"=>0,"G"=>0,"B"=>0));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));


/* Set the graph area */
$myPicture->setGraphArea(50,50,350,350);

/* Create the Scatter chart object */
$myScatter = new pScatter($myPicture,$myData);

/* Draw the scale */
$myScatter->drawScatterScale();

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawScatterScale.png");
```

This example shows how to create two axis and set their position.

## Sample script #2



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
include("../class/pScatter.class.php");

/* Create the pData object */
$myData = new pData();

/* Create the X axis and the binded series */
$myData->addPoints(array(3,12,15,8,5,-5),"X Values");
$myData->setAxisName(0,"X Values");
$myData->setAxisXY(0,AXIS_X);
$myData->setAxisDisplay(0,AXIS_FORMAT_TIME,"i:s");
$myData->setAxisPosition(0,AXIS_POSITION_BOTTOM);

/* Create the Y axis and the binded series */
$myData->addPoints(array(2,7,5,18,19,22),"Y Values");
$myData->setSerieOnAxis("Y Values",1);
$myData->setAxisName(1,"Y Values");
$myData->setAxisXY(1,AXIS_Y);

/* Create the pChart object */
$myPicture = new pImage(400,400,$myData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
 $myPicture->drawFilledRectangle(0,0,400,400,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
 $myPicture->drawGradientArea(0,0,400,400,DIRECTION_VERTICAL,$Settings);
$myPicture->drawGradientArea(0,0,400,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>80));



/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
 $myPicture->drawText(10,13,"drawScatterScale() - Draw the scatter chart scale",array("R"=>255,"G"=>255,"B"=>255));


/* Add a border to the picture */
$myPicture->drawRectangle(0,0,399,399,array("R"=>0,"G"=>0,"B"=>0));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));


/* Set the graph area */
$myPicture->setGraphArea(50,50,350,350);

/* Create the Scatter chart object */
$myScatter = new pScatter($myPicture,$myData);

/* Draw the scale */
$AxisBoundaries = array(0=>array("Min"=>0,"Max"=>3600,"Rows"=>12,"RowHeight"=>300),1=>array("Min"=>0,"Max"=>100));
 $ScaleSettings = array("Mode"=>SCALE_MODE_MANUAL,"ManualScale"=>$AxisBoundaries,"DrawSubTicks"=>TRUE);
 $myScatter->drawScatterScale($ScaleSettings);

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawScatterScale.manual.png");
```

This example shows how to create a manual scale providing the min and max values for the X / Y axis and the divisions settings for the X one.

# drawScatterPlotChart - Draw a scatter plot chart

This function allows you to draw a scatter plot chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

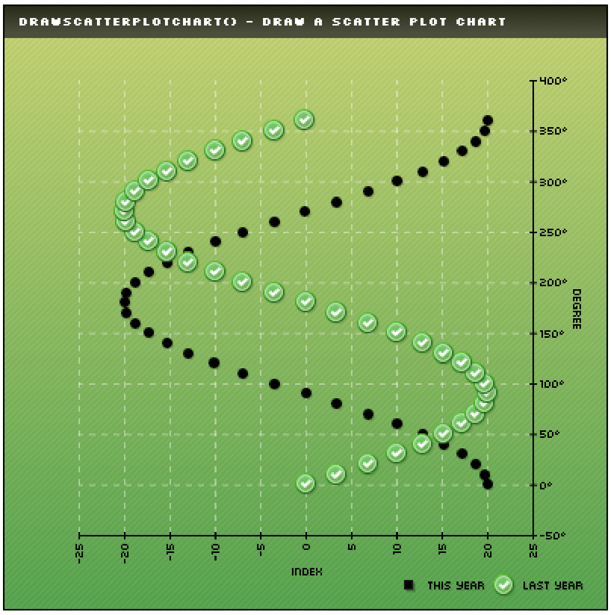
## Calling this function

```
drawScatterPlotChart($Format="");
```

Where :

- Format is an array containing the drawing parameters of the chart. (today none are defined)

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
include("../class/pScatter.class.php");

/* Create the pData object */
$myData = new pData();

/* Create the X axis and the binded series */
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints(cos(deg2rad($i))*20,"Probe 1"); }
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints(sin(deg2rad($i))*20,"Probe 2"); }
$myData->setAxisName(0,"Index");
$myData->setAxisXY(0,AXIS_X);
$myData->setAxisPosition(0,AXIS_POSITION_BOTTOM);

/* Create the Y axis and the binded series */
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints($i,"Probe 3"); }
$myData->setSerieOnAxis("Probe 3",1);
$myData->setAxisName(1,"Degree");
$myData->setAxisXY(1,AXIS_Y);
$myData->setAxisUnit(1,"°");
$myData->setAxisPosition(1,AXIS_POSITION_RIGHT);
```

```
/* Create the 1st scatter chart binding */
$myData->setScatterSerie("Probe 1","Probe 3",0);
$myData->setScatterSerieDescription(0,"This year");
$myData->setScatterSerieColor(0,array("R"=>0,"G"=>0,"B"=>0));

/* Create the 2nd scatter chart binding */
$myData->setScatterSerie("Probe 2","Probe 3",1);
$myData->setScatterSerieDescription(1,"Last Year");
$myData->setScatterSeriePicture(1,"resources/accept.png");

/* Create the pChart object */
$myPicture = new pImage(400,400,$myData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,400,400,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,400,400,DIRECTION_VERTICAL,$Settings);



$myPicture->drawGradientArea(0,0,400,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>80));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawScatterPlotChart() - Draw a scatter plot chart",array("R"=>255,"G"=>255,"B"=>255));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,399,399,array("R"=>0,"G"=>0,"B"=>0));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));


/* Set the graph area */
$myPicture->setGraphArea(50,50,350,350);

/* Create the Scatter chart object */
$myScatter = new pScatter($myPicture,$myData);

/* Draw the scale */
$myScatter->drawScatterScale();

/* Turn on shadow computing */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Draw a scatter plot chart */
$myScatter->drawScatterPlotChart();

/* Draw the legend */
$myScatter->drawScatterLegend(260,375,array("Mode"=>LEGEND_HORIZONTAL,"Style"=>LEGEND_NOBORDER));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawScatterPlotChart.png");
```

# drawScatterLineChart - Draw a scatter line chart

This function allows you to draw a scatter line chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

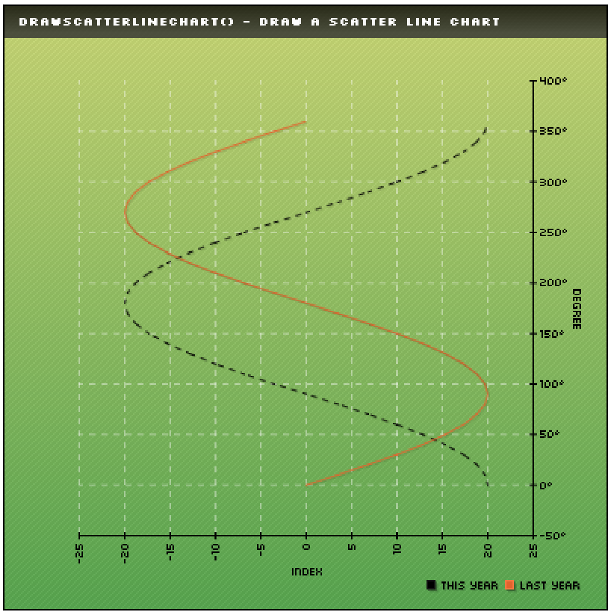
## Calling this function

```
drawScatterLineChart($Format="");
```

Where :

- Format is an array containing the drawing parameters of the chart. (today none are defined)

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
include("../class/pScatter.class.php");

/* Create the pData object */
$myData = new pData();

/* Create the X axis and the binded series */
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints(cos(deg2rad($i))*20,"Probe 1"); }
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints(sin(deg2rad($i))*20,"Probe 2"); }
$myData->setAxisName(0,"Index");
$myData->setAxisXY(0,AXIS_X);
$myData->setAxisPosition(0,AXIS_POSITION_BOTTOM);

/* Create the Y axis and the binded series */
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints($i,"Probe 3"); }
$myData->setSerieOnAxis("Probe 3",1);
$myData->setAxisName(1,"Degree");
$myData->setAxisXY(1,AXIS_Y);
$myData->setAxisUnit(1,"°");
$myData->setAxisPosition(1,AXIS_POSITION_RIGHT);

/* Create the 1st scatter chart binding */
$myData->setScatterSerie("Probe 1","Probe 3",0);
$myData->setScatterSerieDescription(0,"This year");
```

```
$myData->setScatterSerieTicks(0,4);
$myData->setScatterSerieColor(0,array("R"=>0,"G"=>0,"B"=>0));

/* Create the 2nd scatter chart binding */
$myData->setScatterSerie("Probe 2","Probe 3",1);
$myData->setScatterSerieDescription(1,"Last Year");

/* Create the pChart object */
$myPicture = new pImage(400,400,$myData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,400,400,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,400,400,DIRECTION_VERTICAL,$Settings);


$myPicture->drawGradientArea(0,0,400,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"
Alpha"=>80));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawScatterLineChart() - Draw a scatter line chart",array("R"=>255,"G"=>255,"B"=>255));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,399,399,array("R"=>0,"G"=>0,"B"=>0));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));


/* Set the graph area */
$myPicture->setGraphArea(50,50,350,350);

/* Create the Scatter chart object */
$myScatter = new pScatter($myPicture,$myData);

/* Draw the scale */
$myScatter->drawScatterScale();

/* Turn on shadow computing */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Draw a scatter plot chart */
$myScatter->drawScatterLineChart();

/* Draw the legend */
$myScatter->drawScatterLegend(280,380,array("Mode"=>LEGEND_HORIZONTAL,"Style"=>LEGEND_NOBORDER));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawScatterLineChart.png");
```

# drawScatterSplineChart - Draw a scatter spline chart

This function allows you to draw a scatter spline chart. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide.

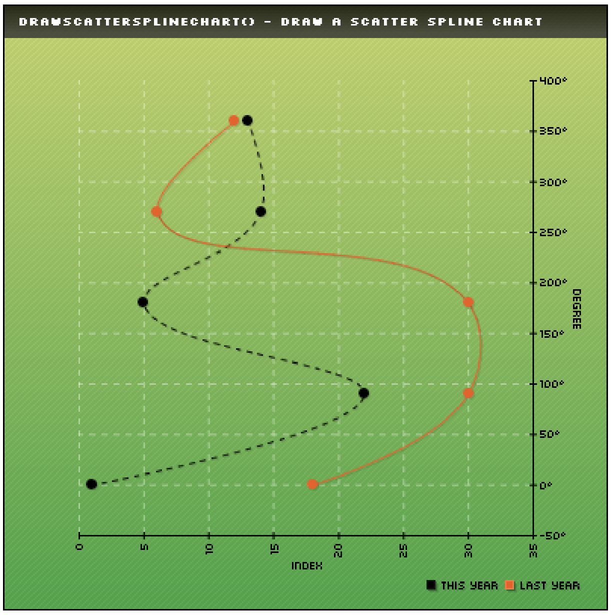
## Calling this function

```
drawScatterSplineChart($Format="");
```

Where :

- Format is an array containing the drawing parameters of the chart. (today none are defined)

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
include("../class/pScatter.class.php");

/* Create the pData object */
$myData = new pData();

/* Create the X axis and the binded series */
for ($i=0;$i<=360;$i=$i+90) { $myData->addPoints(rand(1,30),"Probe 1"); }
for ($i=0;$i<=360;$i=$i+90) { $myData->addPoints(rand(1,30),"Probe 2"); }
$myData->setAxisName(0,"Index");
$myData->setAxisXY(0,AXIS_X);
$myData->setAxisPosition(0,AXIS_POSITION_BOTTOM);

/* Create the Y axis and the binded series */
for ($i=0;$i<=360;$i=$i+90) { $myData->addPoints($i,"Probe 3"); }
$myData->setSerieOnAxis("Probe 3",1);
$myData->setAxisName(1,"Degree");
$myData->setAxisXY(1,AXIS_Y);
$myData->setAxisUnit(1,"°");
$myData->setAxisPosition(1,AXIS_POSITION_RIGHT);

/* Create the 1st scatter chart binding */
$myData->setScatterSerie("Probe 1","Probe 3",0);
$myData->setScatterSerieDescription(0,"This year");
```

```
$myData->setScatterSerieTicks(0,4);
$myData->setScatterSerieColor(0,array("R"=>0,"G"=>0,"B"=>0));

/* Create the 2nd scatter chart binding */
$myData->setScatterSerie("Probe 2","Probe 3",1);
$myData->setScatterSerieDescription(1,"Last Year");

/* Create the pChart object */
$myPicture = new pImage(400,400,$myData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,400,400,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,400,400,DIRECTION_VERTICAL,$Settings);

$myPicture->drawGradientArea(0,0,400,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"Alpha"=>80));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawScatterSplineChart() - Draw a scatter spline chart",array("R"=>255,"G"=>255,"B"=>255));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,399,399,array("R"=>0,"G"=>0,"B"=>0));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));

/* Set the graph area */
$myPicture->setGraphArea(50,50,350,350);

/* Create the Scatter chart object */
$myScatter = new pScatter($myPicture,$myData);

/* Draw the scale */
$myScatter->drawScatterScale();

/* Turn on shadow computing */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Draw a scatter plot chart */
$myScatter->drawScatterSplineChart();
$myScatter->drawScatterPlotChart();

/* Draw the legend */
$myScatter->drawScatterLegend(280,380,array("Mode"=>LEGEND_HORIZONTAL,"Style"=>LEGEND_NOBORDER));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawScatterSplineChart.png");
```

# drawScatterLegend - Write the legend of the active data series

This function will write the legend of your data series based on the description tag specified using the setScatterSerieDescription function. It is possible to tune the rendering by playing with the **$Format** array. To learn more about this please read the Format array guide.

This function accept the same parameters than the drawLegend function.

Information

If you want to specify multi-line labels then you can use the "backslash n" line break character in the serie description tag.

Information

This function preserves your serie special settings (ticks, weight) only in the LEGEND_FAMILY_LINE mode.

## Calling this function

```
drawScatterLegend($X,$Y,$Format="");
```

Where :

- X and Y are the coordinates where will be drawn the legend.
- Format is an array containing the drawing parameters of the arrow.

## Customisation array - Tune up your legend!

It is possible to customize the way your legend will be rendered by playing with this array. Providing a detailled configuration is not mandatory, by default the legend will be drawn in a soft grey box with curvy corners.

- You can specify the font file that will be used with FontName.
- You can specify the font size with FontSize.
- You can specify the font color with FontR, FontG, FontB.
- You can specify the width of the colored boxes with BoxWidth.
- You can specify the height of the colored boxes with BoxHeight.
- You can specify the inner margins with Margin.
- You can specify the background color with R, G, B.
- You can specify the background alpha with Alpha.
- You can specify the border color using BorderR, BorderG, BorderB.
- You can use the Surrounding option to define the border color. This value will be added to the R,G,B factors to define the border color.

You can choose the style that will be applied to you legend box using the **Style** parameter :

- LEGEND_NOBORDER no borders will be drawn around the legend.
- LEGEND_BOX a rectangle will be drawn around the legend.
- LEGEND_ROUND a rounded rectangle will be drawn around the legend.

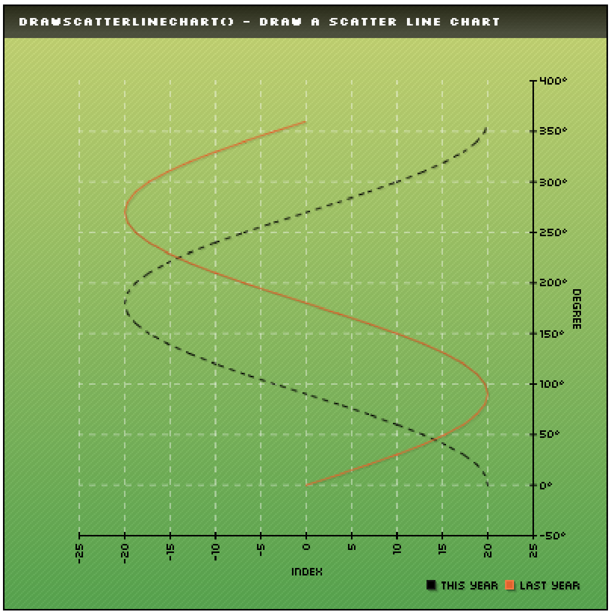
You can also define the way the legend will be written using the **Mode** parameter :

- LEGEND_VERTICAL that will stack vertically the series.
- LEGEND_HORIZONTAL that will stack horizontally the series.

Finally you can choose the way your data serie will be drawn using the **Family** parameter :

- LEGEND_FAMILY_BOX to use filled rectangles.
- LEGEND_FAMILY_CIRCLE to use filled circles.
- LEGEND_FAMILY_LINE to use lines. (preserving the Width & Ticks options)

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
include("../class/pScatter.class.php");

/* Create the pData object */
$myData = new pData();

/* Create the X axis and the binded series */
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints(cos(deg2rad($i))*20,"Probe 1"); }
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints(sin(deg2rad($i))*20,"Probe 2"); }
$myData->setAxisName(0,"Index");
$myData->setAxisXY(0,AXIS_X);
$myData->setAxisPosition(0,AXIS_POSITION_BOTTOM);

/* Create the Y axis and the binded series */
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints($i,"Probe 3"); }
$myData->setSerieOnAxis("Probe 3",1);
$myData->setAxisName(1,"Degree");
$myData->setAxisXY(1,AXIS_Y);
$myData->setAxisUnit(1,"°");
$myData->setAxisPosition(1,AXIS_POSITION_RIGHT);

/* Create the 1st scatter chart binding */
$myData->setScatterSerie("Probe 1","Probe 3",0);
$myData->setScatterSerieDescription(0,"This year");
$myData->setScatterSerieTicks(0,4);
$myData->setScatterSerieColor(0,array("R"=>0,"G"=>0,"B"=>0));

/* Create the 2nd scatter chart binding */
$myData->setScatterSerie("Probe 2","Probe 3",1);
$myData->setScatterSerieDescription(1,"Last Year");

/* Create the pChart object */
$myPicture = new pImage(400,400,$myData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,400,400,$Settings);

/* Overlay with a gradient */
```

```
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,400,400,DIRECTION_VERTICAL,$Settings);



$myPicture->drawGradientArea(0,0,400,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>80));

/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawScatterLineChart() - Draw a scatter line chart",array("R"=>255,"G"=>255,"B"=>255));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,399,399,array("R"=>0,"G"=>0,"B"=>0));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));


/* Set the graph area */
$myPicture->setGraphArea(50,50,350,350);

/* Create the Scatter chart object */
$myScatter = new pScatter($myPicture,$myData);

/* Draw the scale */
$myScatter->drawScatterScale();

/* Turn on shadow computing */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Draw a scatter plot chart */
$myScatter->drawScatterLineChart();

/* Draw the legend */
$myScatter->drawScatterLegend(280,380,array("Mode"=>LEGEND_HORIZONTAL,"Style"=>LEGEND_NOBORDER));

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawScatterLineChart.png");
```

# getScatterLegendSize- Determine the size of the legend box

This function returns the width and height of the scatter legend box that would be drawn with the specified parameters. You can use this function to align your legend boxes depending of its printed size.

This function will return an associative array containing the **Width** and **Height** keys.

Information

Basically you'll use the same Format array with this function than the one used for the drawLegend() function.

## Calling this function

```
getScatterLegendSize($Format="");
```

Where :

- Format is an array containing the drawing parameters of the legend.

### Customisation array - Tune up your legend!

It is possible to customize the way your legend will be rendered by playing with this array. Providing a detailled configuration is not mandatory, by default the legend will be drawn in a soft grey box with curvy corners.

- You can specify the font file that will be used with FontName.
- You can specify the font size with FontSize.
- You can specify the size of the colored boxes with BoxSize.
- You can specify the inner margins with Margin.

You can choose the style that will be applied to you legend box using the **Style** parameter :

- LEGEND_NOBORDER no borders will be drawn around the legend.
- LEGEND_BOX a rectangle will be drawn around the legend.
- LEGEND_ROUND a rounded rectangle will be drawn around the legend.

You can also define the way the legend will be written using the **Mode** parameter :

- LEGEND_VERTICAL that will stack vertically the series.
- LEGEND_HORIZONTAL that will stack horizontally the series.

### Sample script

Code sample

```
/* Rounded legend box */
$myPicture->setFontProperties(array("FontName"=>"fonts/pf_arma_five.ttf","FontSize"=>6));
$Size = $myPicture->getScatterLegendSize(array("Style"=>LEGEND_ROUND,"Mode"=>LEGEND_HORIZONTAL));

print_r($Size);
```

The script will returns an associative array :

```
Array
(
    [Width] => 123
    [Height] => 23
)
```

## drawScatterBestFit - Draw the "line of best fit"

This function allows you to draw the line of best fit associated with your given data series. All the drawing parameters are given trough a **$Format** array. To learn more about this please read the Format array guide. The mathematical formula used to compute it is described below :

$$Slope = m = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum(x^2) - (\sum x)^2}$$

$$Intercept = b = \frac{\sum y - m(\sum x)}{n}$$

### Calling this function

```
drawScatterBestFit($Format="");
```

Where :

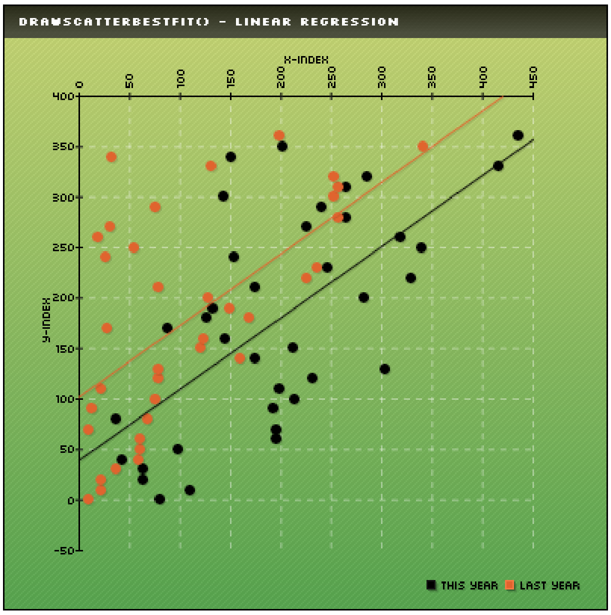
- Format is an array containing the drawing parameters of the chart. (today none are defined)

## Customisation array - Tune up your chart!

It is possible to customize the rendering by playing with this array. Providing a detailled configuration is not mandatory. You'll see below a representation of all the customization possible :

- You can optionally specify the line ticks width with Ticks.

## Sample script



Code sample

```
/* pChart library inclusions */
include("../class/pData.class.php");
include("../class/pDraw.class.php");
include("../class/pImage.class.php");
include("../class/pScatter.class.php");

/* Create the pData object */
$myData = new pData();

/* Create the X axis and the binded series */
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints(rand(1,20)*10+rand(0,$i),"Probe 1"); }
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints(rand(1,2)*10+rand(0,$i),"Probe 2"); }
$myData->setAxisName(0,"X-Index");
$myData->setAxisXY(0,AXIS_X);
$myData->setAxisPosition(0,AXIS_POSITION_TOP);

/* Create the Y axis and the binded series */
for ($i=0;$i<=360;$i=$i+10) { $myData->addPoints($i,"Probe 3"); }
$myData->setSerieOnAxis("Probe 3",1);
$myData->setAxisName(1,"Y-Index");
$myData->setAxisXY(1,AXIS_Y);
$myData->setAxisPosition(1,AXIS_POSITION_LEFT);

/* Create the 1st scatter chart binding */
$myData->setScatterSerie("Probe 1","Probe 3",0);
$myData->setScatterSerieDescription(0,"This year");
$myData->setScatterSerieColor(0,array("R"=>0,"G"=>0,"B"=>0));

/* Create the 2nd scatter chart binding */
$myData->setScatterSerie("Probe 2","Probe 3",1);
$myData->setScatterSerieDescription(1,"Last Year");

/* Create the pChart object */
```

```
$myPicture = new pImage(400,400,$myData);

/* Draw the background */
$Settings = array("R"=>170, "G"=>183, "B"=>87, "Dash"=>1, "DashR"=>190, "DashG"=>203, "DashB"=>107);
$myPicture->drawFilledRectangle(0,0,400,400,$Settings);

/* Overlay with a gradient */
$Settings = array("StartR"=>219, "StartG"=>231, "StartB"=>139, "EndR"=>1, "EndG"=>138, "EndB"=>68, "Alpha"=>50);
$myPicture->drawGradientArea(0,0,400,400,DIRECTION_VERTICAL,$Settings);



$myPicture->drawGradientArea(0,0,400,20,DIRECTION_VERTICAL,array("StartR"=>0,"StartG"=>0,"StartB"=>0,"EndR"=>50,"EndG"=>50,"EndB"=>50,"

Alpha"=>80));


/* Write the picture title */
$myPicture->setFontProperties(array("FontName"=>"../fonts/Silkscreen.ttf","FontSize"=>6));
$myPicture->drawText(10,13,"drawScatterBestFit() - Linear regression",array("R"=>255,"G"=>255,"B"=>255));

/* Add a border to the picture */
$myPicture->drawRectangle(0,0,399,399,array("R"=>0,"G"=>0,"B"=>0));

/* Set the default font */
$myPicture->setFontProperties(array("FontName"=>"../fonts/pf_arma_five.ttf","FontSize"=>6));


/* Set the graph area */
$myPicture->setGraphArea(50,60,350,360);

/* Create the Scatter chart object */
$myScatter = new pScatter($myPicture,$myData);

/* Draw the scale */
$myScatter->drawScatterScale();

/* Turn on shadow computing */
$myPicture->setShadow(TRUE,array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>10));

/* Draw a scatter plot chart */
$myScatter->drawScatterPlotChart();

/* Draw the legend */
$myScatter->drawScatterLegend(280,380,array("Mode"=>LEGEND_HORIZONTAL,"Style"=>LEGEND_NOBORDER));

/* Draw the line of best fit */
$myScatter->drawScatterBestFit();

/* Render the picture (choose the best way) */
$myPicture->autoOutput("pictures/example.drawScatterBestFit.png");
```

# Table of Contents