

JpGraph マニュアル

アシアル株式会社
<http://www.asial.co.jp/>
jpgraph@asial.co.jp

Copyright © 2005-2010 Asial Corporation. All Rights Reserved.
This manual is based on "JpGraph Documentation" by Aditus Consulting.Sweden.

アシアル株式会社
<http://www.asial.co.jp/>
jograph@asial.co.jp

[両面コピー用表紙裏]

コンタクト情報

(ア) JpGraphへのお問い合わせ

以下の連絡先までお問い合わせ下さい。

アシアル株式会社 (Asial Corporation)

〒113-0033 東京都文京区本郷 3-38-1-7F

TEL:03-5875-6862

E-Mail: jpgraph@asial.co.jp

(イ) よくあるご質問

FAQ は以下の URL で確認できます。

>> <http://www.asial.co.jp/jpgraph/faq.php>

(ウ) 製品のご購入

JpGraph 製品のご購入は以下の URL からできます。

>> http://www.asial.co.jp/phpshop/shop_detail.php?v=jpgraph

(エ) エラーや不具合

IonCube 製品の注意事項については、可能な限りこのユーザーガイドに書かれています。
それ以外のエラーや不具合を発見した場合、あるいは改善提案などがありましたら、
jpgraph@asial.co.jp にお送りください。

目次

コンタクト情報.....	1
(ア) JpGraphへのお問い合わせ	1
(イ) よくあるご質問.....	1
(ウ) 製品のご購入	1
(エ) エラーや不具合	1
目次.....	2
1 マニュアルについて	9
1.1 製品について	9
2 イントロダクション	9
2.1 バージョン	9
2.2 ソフトウェアのライセンス.....	9
2.3 JpGraph の機能.....	10
2.4 JpGraph に関するお問い合わせ.....	11
3 インストール	12
3.1 インストールの前に	12
3.1.1 GD ライブライリのインストールを確認する	12
3.1.2 GD2 のインストールを確認する	12
3.1.3 True Type フォントのファイルを用意する	13
3.1.4 日本語フォントを使用する.....	13
3.2 インストールをカスタマイズする	15
3.3 動作に必要なファイル	15
3.3.1 プロット拡張モジュール	15
3.4 イメージフォーマットと外部イメージのライブラリ	16
3.5 JpGraph の詳細なインストール	17
3.6 インストールのトラブルシューティング	17
3.7 PHP 4 をコンパイルする	19
4 はじめに : ダイナミック イメージの生成	19
4.1 この章の概要	20
4.1.1 この章について	20
4.1.2 この章で取り扱わない項目	20
4.2 画像を出力するには	20
4.2.1 JpGraph ライブライリを使用する	21

4.3 コマンドライン版の PHP を使用する	21
4.4 JpGraph の基本概念と画像の生成	22
4.5 JpGraph の画像形式を選択する	23
4.6 画像をブラウザに送る以外の処理方法	24
4.7 JpGraph でフォントを使用する	24
4.7.1 TrueType フォントをインストールする	24
4.7.2 TrueType フォントが動作していることを確認する	25
4.7.3 フォントを指定する	25
4.7.4 JpGraph にフォントを追加する	27
4.7.5 JpGraph で文字列の文字揃えを行う	28
4.8 JpGraph でカラーを指定する	30
4.8.1 透明度を調節する	30
4.8.2 定義済みのカラー	31
4.8.3 円プロット用のテーマ カラー	36
5 キャッシュ システムを使用する	37
5.1 キャッシュ システムを有効にする	37
5.2 スクリプトでキャッシュを使用する	38
5.3 クライアントサイド イメージマップを使用する	38
5.4 最後に	40
5.5 すべてのグラフで共通の機能	40
5.5.1 クリッピング	40
5.5.2 共通のプロパティ	40
5.5.3 共通のプロパティ	41
6 イメージ マップを使用する	41
6.1 イメージ マップ スクリプトの基本構造ゲット	41
6.2 イメージ マップ プロットにターゲットを指定する	42
6.3 StrokeCSIM() を使用する	43
6.4 イメージ マップの例	44
6.5 StrokeCSIM() の動き	45
6.6 イメージ マップの取得	45
6.7 イメージ マップとキャッシュ システム	45
7 直角の XY プロットを使用する	46
7.1 ライン プロット	46
7.1.1 ライン プロット XXX にプロット記号を追加する	49
7.1.2 各データ ポイントに値を表示する	50
7.1.3 同じグラフに複数のプロットを追加する	51
7.1.4 2 個目の Y 軸を追加する	52

7.1.5 グラフに凡例を追加する	53
7.1.5.1 凡例のレイアウトを調節する	56
7.1.6 ラインプロットで空白値を使用する	57
7.1.7 ステップ形式のラインプロットを描画する	58
7.1.8 対数グラフを使用する	58
7.1.9 スケールの詳細	61
7.1.10 プロットでグリッド(目盛り)ラインを調整する	62
7.1.11 塗りつぶされたグリッド線を使用する	63
7.1.12 X軸にテキストラベルを指定する	64
7.1.13 テキストスケールのチックを調整する	65
7.1.14 塗りつぶされたラインプロットを使用する	67
7.1.15 積み上げラインプロットを使用する	68
7.1.16 三次スプラインで滑らかなラインプロットを作成する	70
7.1.17 プロット記号をラインプロットに追加する	71
7.2 バー プロット	75
7.2.1 バーの幅を調整する	77
7.2.2 各バーの値を表示する	78
7.2.3 バーにドロップシャドウを追加する	81
7.2.4 テキストスケールのバーの配置を調整する	82
7.2.5 グループ化されたバー プロットを使用する	82
7.2.6 積み上げバー プロットを使用する	84
7.2.7 グループ化された積み上げバー プロットを使用する	85
7.2.8 水平のバー プロット	86
7.2.9 バー プロットにグラデーションカラーを使用する	91
7.2.10 一部を塗りつぶしたバー プロットの作成	95
7.3 エラー プロット	96
7.3.1 ラインエラー プロットを使用する	97
7.4 散布プロット	99
7.5 フィールド プロット	104
7.6 ボックスとストック チャート	105
7.6.1 ストック チャート	105
7.6.2 ボックスプロット：中央線の入ったストック チャート	106
7.6.3 ボックスとストック チャートのイメージマップ	107
7.7 複数のプロット形式を組み合わせる	107
7.8 スケールを手動で指定する	111
7.9 目盛りの間隔を自動で調節する	113
7.10 日付・時間スケールを使用する	114
7.10.1 自動日付・時間スケールを使用する	114
7.10.1.1 入力データを指定する	115
7.10.1.2 最初と最後の直線を調整する	116

7.10.1.3 ラベル フォーマットを調整する	117
7.10.1.4 密集したデータ スケールを調整する	118
7.10.2 手動コールバックで日付・時間スケールを指定する	118
7.11 テキストスケールのラベルを調整する	120
7.12 グラフにテキスト文字列を追加する	124
7.13 グラフにタイトルとフッターを追加する	127
7.14 タブ形式のタイトルを追加する	128
7.15 背景イメージを使用する	129
7.16 背景グラデーション カラーを使用する	131
7.17 プロットマークにコールバック関数を使用する	133
7.18 グラフを 90 度回転する	134
7.19 軸をフォーマットする	135
7.19.1 一般的な 2 軸グラフ	135
7.19.2 関数形式の軸	137
7.19.3 スケール ラベルの位置を調整する	138
7.19.4 スケール ラベルをフォーマットする	139
7.19.5 Y 軸を反転する	139
7.20 自動スケールの上限を設定する	140
7.21 バンド パターンとグラフのカラーを追加する	142
7.21.1 パターンをカスタマイズする	147
7.22 静止ラインをプロットに追加する	149
8 X、Y プロット以外を使用する	150
8.1 レーダー プロット	150
8.1.1 簡単なレーダー プロットを作成する	151
8.1.2 プロットの軸と凡例にタイトルを指定する	153
8.1.3 レーダー プロットにグリッド ラインを描画する	153
8.1.4 1つのグラフ内に複数のレーダー プロットを追加する	154
8.1.5 レーダー プロットにプロットマークを表示する	155
8.2 円プロット	156
8.2.1 円プロットにガイド ラインを追加する	159
8.2.2 3D 円プロットを作成する	162
8.2.3 円スライスを抜き出す	164
8.2.4 円プロットのラベルを指定して調整する	165
8.2.5 円プロットのスライスに色とテーマを指定する	166
8.2.6 スライスに影を表示させる	168
8.2.7 2D 円プロットに中心円を表示させる	169
8.3 極プロット	171
8.3.1 概要	171
8.3.2 半径スケールを調整する	173

8.3.3 グリッド ラインを調整する.....	174
8.3.4 ラベルのフォントを設定する.....	176
8.3.5 ラベルを調整する	178
8.3.6 イメージ マップを使用する.....	178
8.3.7 詳細に設定した極プロットの例.....	178
9 ガント チャート	179
9.1 ガント チャートの使用目的	179
9.2 JpGraph ガント チャート モジュールの機能	179
9.3 シンプルなガント チャート.....	180
9.4 ガント チャートの構造	183
9.5 ガント チャートを作成する	184
9.6 ガントチャート内のオブジェクトの位置を調整する	185
9.7 ガント バー	186
9.7.1 鉛直位置を指定する.....	186
9.7.2 バーの始点と終点を指定する	187
9.7.3 マイルストーン	188
9.7.4 バーティカル ライン	190
9.7.5 ガント バーにマーカーを追加する.....	193
9.7.6 バーの間の最小距離を調整する.....	196
9.8 スケール ヘッダーをフォーマットする	199
9.8.1 分スケール.....	201
9.8.2 時スケール.....	201
9.8.3 日スケール.....	202
9.8.4 週スケール.....	203
9.8.5 月スケール.....	204
9.8.6 年スケール.....	204
9.9 バーをより詳細にフォーマットする	205
9.9.6 バーに見出しを追加する	205
9.9.7 バーに進行指標(progress indicators)を追加する	206
9.10 アクティビティのグループ化	208
9.11 タイトルに複数行を設定する	208
9.11.6 列にタイトルを加える.....	211
9.11.7 列タイトルに対して CSIM エントリーを指定する	212
9.12 より全体的なガント チャートのフォーマット	212
9.12.6 テーブル タイトルを追加する	213
9.12.7 仕切り線を修正する	214
9.12.8 プロット周辺のボックスを修正する.....	215
9.12.9 水平なグリッドとラインカラーの交互切り替え	216
9.12.10 ガント チャートにアイコンを追加する.....	217

9.12.11 自動サイズ調整で余白を調整する	217
9.13 ガント チャート作成の簡略化	217
9.14 ガント チャートに CSIM (クライアントサイド イメージ マップ) を追加する	219
9.15 アクティビティ間に制限を追加する.....	220
9.16 より高度なフォーマット.....	222
9.16.6 ガント チャートの一部分のみを表示する	222
9.16.7 週の開始曜日を指定する	222
9.17 ローカライズ	222
9.18 JpGraph のアンチエイリアス	224
9.18.6 アンチエイリアスを有効にする.....	224
9.19 グラフを回転させる.....	226
9.20 イメージと背景用の明るさおよびコントラストを調整する.....	230
9.21 グラフ生成の時間を計る.....	231
9.22 様々なコンテキストで国旗を使用する.....	233
9.23 グラフ上のアイコンを追加する	235
10 キャンバス グラフを使用する.....	236
10.1 キャンバス グラフの使用方法.....	239
10.2 簡単なキャンバスの作成	241
10.3 キャンバスに線と長方形を追加する.....	243
10.4 キャンバス スケールを使用する	244
10.5 サンプル アプリケーション: DB schema を描画する.....	249
11 アンチスパム	254
11.1 アンチスパム画像を作成する	254
12 エンタープライズ版スタンダードで作成できるグラフ	256
12.1 オドメータ(走行距離積算計)	256
12.1.1 オドメータ・モジュールの機能	256
12.1.2 基本オドメータ・グラフの作成とフォーマット	257
12.1.2.1 基本オドメータを作成する	258
12.1.3 サンプル集	259
12.2 風配図	261
12.2.1 風配図モジュールの機能	262
12.2.2 風配図の種類	262
12.2.3 基本風配図の作成とフォーマット	266
12.2.3.1 風配図の目盛りを指定する	268
12.2.3.2 方位ラベルを指定する	270
12.2.4 サンプル集	273

目次

12.3 バーコード	283
12.3.1 バーコードの仕組み	284
12.3.2 バーコードの種類(シンボロジー)	285
12.3.2.1 数字のみのシンボロジー	285
12.3.2.2 アルファベットと数字のシンボロジー	285
12.3.2.3 使用するシンボロジーの選択	286
12.3.3 バーコード・モジュールで使える機能	286
12.3.4 バーコードを作成する - クイック・スタート	287
12.3.5 エラー・ハンドリング	288
12.3.6 コマンドラインを使用してバーコードを作成する	289
12.4 テーブル	291
12.4.1 テーブルの構成	292
12.4.1.1 基本テーブルを作成する	292
12.4.2 サンプル集	295
12.5 マトリックス・グラフ	300
12.4.2 マトリックス・プロットの機能	301
12.4.3 基本的なマトリックス・グラフの作成とフォーマット	302
12.4.4 入力データのメッシュ補間	303
12.4.5 モジュール・タイプを変更する	305
12.4.6 プロットのアルファ・ブレンディング	306
JPGGRAPH の設定を変更する	308

1 マニュアルについて

1.1 製品について

JpGraph は Aditus Consulting(スウェーデン)が開発する、PHP4／PHP5 対応の高機能なグラフ生成ライブラリです。JpGraph を使うと、ほとんどの種類のグラフ形式を、即時に作ることができます。アシアルでは日本語マニュアルや QR コード生成機能を加え、日本語版として提供いたします。

2 イントロダクション

2.1 バージョン

このマニュアルは JpGraph 1.2.x および 3.0.x に対応しています。JpGraph 1.x は PHP 4.3 以降をサポートしており(PHP 5.x はサポートしていません)、JpGraph 3.0.1 は PHP 5.1 から 5.3 をサポートしています(PHP 4.x はサポートしていません)。PHP 4.2 以下でも動作するという報告はされていますが、サポート対象外となります。

JpGraph 1.2.x サポート終了のお知らせ

2008 年 12 月末日をもって、開発元でのサポートの終了に伴い、弊社での「JpGraph 1.x (PHP4 対応)」のサポートを誠に勝手ながら、一部を除き終了することとなりました。なお、ソフトウェアの配布はこれまで通り行っています。

■ 継続されるサポート

開発元への問い合わせが不要な、既知の問題へのサポートは継続されます。終了となるのは、開発元への問い合わせが必要な、新しい現象についてのサポート、および新規バグへの対応です。

2.2 ソフトウェアのライセンス

JpGraph は、QPL 1.0 (Qt Free Licensee) ライセンスと、JpGraph エンタープライズ版 ライセンスのデュアルライセンスで提供されるオープンソースのライブラリです。個人やコミュニティなどの非営利目的での使用に限り、無料でコミュニティ版をご使用いただけます。しかし、商用などの営利的な目的で JpGraph を使用する場合は、有料のエンタープライズ版ライセンスのご購入が必要となります。エンタープライズ版では、QR コードを含む、多くのバーコード用ライブラリや、追加のプロット形式も提供しています。

2.3 JpGraph の機能

JpGraph は、最小限のコードで手軽に素早くグラフを描画すること、詳細な設定により機能的で美しいグラフを描画することの、両方を目的として作られた、オブジェクト指向のグラフ ライブラリです。ほとんどのパラメータでは、デフォルトの値が設定されているので、簡単にグラフを作ることができます。また、わずかなコードを記述することで、見た目の美しいグラフを作ることもできます。

JpGraph で利用できる機能には以下のものがあります。

- 柔軟なスケール機能(テキスト型、整数型、対数型、などから選択できます)。
- PNG、GIF、JPG グラフィック形式のサポート
- (ただし、利用できる形式は PHP のインストール状況によって異なります)。
- HTTP サーバーの負荷を削減するためのキャッシュ機能。
- 画像をファイルに生成するための、バッチ モードのサポート。
- クライアント サイド イメージ マップ 機能のサポート。
- 賢い自動スケール機能。
- 手動のスケールも完全にサポート、目盛りの詳細な調整も可能。
- 複数の書式設定による、背景画像の表示機能。
- PHP でカラーや明度を直接調節できる機能。
- 自動スケールの明示的な範囲調整機能。
- 種類までの Y 軸のサポート。
- ライン プロット、円プロット、エラー プロットなどを含む、多くの種類のプロット形式のサポート。
- 1 つのグラフに描画できるプロットの数は無制限。
- ユーザー定義の軸位置。
- 7 種類の形式でグラデーション カラーをサポート。
- 柔軟なオブジェクト指向フレームワークによる設計。
- 自動の凡例生成機能。
- 垂直、水平グリッドのサポート。
- 線のアンチエイリアスのサポート。
- グラフの背景画像のサポート。
- リニア グラフの回転のサポート。
- 400 以上の定義済みカラー。
- モジュール化された設計。
- その他の機能。

これらの機能に加え、ライブラリの各オブジェクトの命名規則は非常に直感的です。たとえば、各オブジェクトのカラーを指定したい場合は、axis や grid オブジェクトの SetColor() メソッドを使用します。

2.4 JpGraph に関するお問い合わせ

JpGraph のお問い合わせは、以下のメールアドレスよりお願い致します。

JpGraph 製品についてのいかなる御質問・お問い合わせにも迅速に対応いたします。
些細な事でもお問い合わせください。

お問い合わせ先メールアドレス : jpgraph@asial.co.jp

3 インストール

3.1 インストールの前に

インストールを行う前に、PHP ファイルが必要なライブラリ ファイルをインクルードできることを確認してください。JpGraph を使用するには、PHP をインストールした際に、image (GD) エクステンションをサポートしておく必要があります。このことを確認するには、phpinfo() 関数の出力を確認するか、または PHP が imagecreate() 関数をサポートしているかどうか確認します。JpGraph を動作させるには、GD ライブラリが PHP 上で正常に動作している必要があります。JpGraph を使う際には、バージョン 4.3.x 以上の PHP をご使用下さい。JpGraph は PHP 4.3.x 以前のバージョンでは、**動作確認テストを行っておりません**。PHP4.2 以下はサポートしていませんので、ご注意下さい。1.x バージョンのライブラリは PHP 5.x をサポートしておりません。逆に、3.x バージョンのライブラリは PHP 4.x をサポートしておりません。

3.1.1 GD ライブラリのインストールを確認する

GD のインストールを確認するには以下の例を参照してください。例では、単純な GD を使用し非常にシンプルなイメージを作成し、PNG フォーマットでイメージを出力します。スモーク・テストは GD ライブラリが PHP から使用可能であるかを確認するためのものです。JpGraph ライブラリで作業するにはこれは絶対的に必要な条件です。(※最新 JpGraph では、GD1 に対応していないため GD2 をご利用ください。)

以下のコードをドキュメント ルートに保存し、正しく起動するか確認してください。

```
$im = @ImageCreate (150, 100);
or die ("Cannot create a new GD image.");
$background_color = ImageColorAllocate ($im, 255, 255, 255);
$text_color = ImageColorAllocate ($im, 233, 14, 91);
ImageString ($im, 1, 5, 5, "A Simple Text String", $text_color);
header ("Content-type:image/png");
ImagePng ($im);
```

上記のスクリプトが動かなかった場合、またはエラーや警告がスクリーンに表示された場合、インストールする前にこれらの問題を修正する必要があります。

3.1.2 GD2 のインストールを確認する

より高度な機能を使うには、GD2 ライブラリをインストールする必要があります。GD2 ライブラリでは、アルファベット混じり(alpha-blending) およびトゥルー・カラー イメージのような機能が使用できます。GD2 ライブラリは、4.2.x 以上の PHP バージョンに対応しています。GD2 ライブラリがインストールされているか確認するには、以下のスクリプトが動かなければなりません。

```
$im = imagecreatetruecolor (300, 200);
$black = imagecolorallocate ($im, 0, 0, 0);
$white = imagecolorallocate ($im, 255, 255, 255);
imagefilledrectangle($im,0,0,399,99,$white);
```

```
imagerectangle($im,20,20,250,190,$black);
header ("Content-type:image/png");
imagepng ($im);
```

上記のスクリプトを実行することで、ブラウザ上に黒い長方形が表示されます。

3.1.3 True Type フォントのファイルを用意

JpGraph は、標準の ASCII 7 ビットの文字セットをサポートするだけのデフォルトの標準的なビットマップのフォントを含んでいます。アクセントのある文字、UTF-8、中国語、日本語を使用するには、TTF フォントをダウンロードする必要があります。様々なライセンス上の問題により、TTF フォントは JpGraph パッケージでは提供されていません。TTF フォントを有効にするには、3 の代案があります。

1. Windows プラットフォームの場合、標準のフォント ディレクトリで JpGraph の TTF ディレクトリをポイントします。
2. Unix プラットフォームの場合、<http://corefonts.sourceforge.net/> からコアな MS WEB-initiative フォントをダウンロードしてインストールすることができます。
3. また、<http://www.gnome.org/fonts/> のヴェラ・ビットストリーム TTF フォントを使用することも可能です。

PHP インストール TTF をサポートするということを確認する必要があります (FreeType 1、または FreeType 2 ライブライのどちらか)。また、複数の適した TTF フォント ファイルを使用可能にする必要があります。ライブラリでフォントファイルを使用可能にするには、フォント ファイルの場所にディレクトリ パスを構成ファイル、jpg-config.inc.php で指定する必要があります。

GD ライブラリが動作しているかどうかを確かめるには、以下のスクリプトを実行します。

このネーミング規則は、利用可能なフォント・ファイルの標準のネーミングに従います。

ライブラリのインストールが MS Windows を起動しているコンピューターで行われた場合、Windows すでに使用可能なフォント ファイルの使用を推奨します。

(たいてい C:¥WINDOWS¥FONTS にあります)

X11 を派生的に起動している UNIX で行う場合、フォントの場所はバージョンと UNIX ブランドでは異なります。共通で使われるパスは、/usr/X11R6/lib/X11/fonts/truetype/” です。

最後に、ライブラリで本来サポートされていない追加のフォントをインストールすることができます。これはライブラリ ファイルを増やすことを要求し高度な使用とみなされるので、イントロダクションでは詳細については解説しません。

3.1.4 日本語フォントを使用する

1. 使用する日本語 TrueType フォントをフォント用のディレクトリに格納します。

ここでは例として、独立行政法人 情報処理推進機構が提供している IPA フォントを使用します。

<http://ossipedia.ipa.go.jp/ipafont/old/index.html>

上記 URL より「一般利用者向けIPAフォント使用許諾契約書」に同意の上、

IPAfont00203.zip (15.3 MB)

をダウンロードして解凍してください。

生成された fonts ディレクトリの中にある、ipam.ttf、ipamp.ttf、ipag.ttf、ipagp.ttf を使用します。

この 4 つのフォントファイルを、用意したフォントディレクトリに移動してください。

ここでは例として、

/usr/share/fonts/ja/TrueType/

にフォントファイルを格納することにします。

2. jpg-config.inc 40 行目付近にある以下の文章のコメントアウトを外します。

```
// DEFINE("MBTTF_DIR","/usr/local/fonts/ttf/");  
DEFINE("MBTTF_DIR","/usr/local/fonts/ttf/");
```

3. 先ほどの定義文の 2 番目のパラメーターに、フォントを格納したディレクトリ名を指定します。

例) DEFINE("MBTTF_DIR","/usr/share/fonts/ja/TrueType/");

4. jpg-config.inc.php 80 行目付近で定義されている下記の定義文を修正します。

```
DEFINE('MINCHO_TTF_FONT','ipam.ttf');  
DEFINE('PMINCHO_TTF_FONT','ipam.ttf');  
DEFINE('GOTHIC_TTF_FONT','ipag.ttf');  
DEFINE('PGOTHIC_TTF_FONT','ipagp.ttf');
```

これらの 2 番目のパラメーターを、それぞれ用意したフォントファイル名に変更します。

ここでは用意した IPA フォント名がすでに指定されているのでこのままで使用します。

日本語フォントはここで指定した 4 種類のみ使用できます。

以上の設定で JpGraph で日本語フォントを使用できるようになります。

フォントファミリーは上から順に、FF_MINCHO、FF_PMINCHO、FF_GOTHIC、FF_PGOthic と設定されます。

例えば、タイトルに ipam.ttf を使用したい場合は JpGraph のソースコード内で以下のように設定します。

例) \$graph->title->SetFont(FF_PMINCHO);

日本語フォントでは「太字」、「斜体字」、「太斜体字」のスタイルを使用することはできません。

日本語フォントで文字のサイズを変更したいときは、フォントスタイルに FS_NORMAL を指定してください。

例) \$graph->title->SetFont(FF_MINCHO,FS_NORMAL,20);

3.2 インストールをカスタマイズする

JpGraph を動作させるには、キャッシングの設定と TTF フォントの格納されているディレクトリを指定する必要があります。デフォルトでは、TTF ファイルは `/usr/local/fonts/ttf/` にあり、キャッシング ファイルは `/tmp/jpgraph_cache/` に格納されていることを前提としています。これらは、`jpg-config.inc.php` の最上部にて定義されています。

キャッシング機能を使用する場合、キャッシング ディレクトリに PHP が書き込み可能であることを確認してください。もし “Can't write file xxx.yyy” といったエラーが発生した場合、キャッシング ディレクトリに書き込みできないのが原因です。キャッシングに関する詳細は、キャッシング システムを使用するを参照してください。

3.3 動作に必要なファイル

JpGraph の動作には、以下のファイルが必要です。

- `jgraph.php` - 基礎ライブラリで常に必要です
- `jpg-config.inc.php` - 構成ファイル

3.3.1 プロット拡張モジュール

プロットをグラフに追加する場合、作成するプロットの機能によって以下のファイルが必要です。

- `jpgraph_log.php` - X 軸と Y 軸に対数軸を使う場合の拡張モジュール。
- `jpgraph_line.php` - プロットの拡張モジュール。様々な種類のライン プロットを描画するためには必要です。
- `jpgraph_date.php` - データ スケールを扱うプロットの拡張モジュール。
- `jpgraph_bar.php` - プロットの拡張モジュール。様々な種類のバー プロットを描画するためには必要です。
- `jpgraph_error.php` - プロットの拡張モジュール。様々な種類のエラー プロットを描画するために必要です。
- `jpgraph_scatter.php` - プロットの拡張モジュール。散布 プロットとインパルス プロットを描画するために必要です。
- `jpgraph_spider.php` - プロットの拡張モジュール。スパイダー プロットを描画するために必要です。
- `jpgraph_pie.php` - プロットの拡張モジュール。円プロットの描画に必要です。
- `jpgraph_pie3d.php` - プロットの拡張モジュール。3D 円プロットの描画に必要です。
- `jpgraph_gantt.php` - プロットの拡張モジュール。ガント プロットの作成に必要です。
- `jpgraph_radar.php` - プロットの拡張モジュール。レーダー プロットの描画に必要です。

- `jpgraph_polar.php` – プロットの拡張モジュール。極プロットの描画に必要です。
- `jpgraph_gantt.php` – プロットの拡張モジュール。ガントチャートの描画に必要です。
- `jpgraph_regstat.php` – 統計用のエクステンション。プロットのスプラインとベジェ曲線が必要です。
- `jpgraph_stock.php` – プロットの拡張モジュール。ストック チャートの描画に必要です。
- `jpgraph_antispam.php` – OCR プログラムでデジタル化で読みにくい文字+イメージから成るアンチ・スパムのイメージを生成する特別な拡張です。
- `jpgraph_gradient.php` – 内部モジュール。すべてのカラー グラデーションを保持しています。
- `jpgraph_gb2312.php` – GB2312 フォント エンコーディングを扱うための拡張モジュール。
- `jpgraph_plotmark.inc` – ライン プロットおよび散布プロットでプロットマークを使うための拡張モジュール。
- `jpgraph_iconplot.php` – プロットに追加するアイコンをサポートするための拡張モジュール。
- `jpgraph_plotband.php` – グラフにプロットバンドを追加するための拡張モジュール。
- `jpgraph_flags.php` – プロット マークや背景に国のフラグを扱うための拡張モジュール。
- `jpgraph_utils.php` – 様々なユーティリティ関数。
- `imgdata_*.inc` – プロットマークに内蔵されたイメージ。
- `flags*.dat` – 国旗のイメージ データ。国旗のあらかじめコンパイルしたデータ。
- `jpgraph_canvas.php` – キャンバス上で任意のグラフィックを描画するための拡張モジュール。
- `jpgraph_canvtools.php` – キャンバスのアドオンとして使用する、任意の形状を描画するための拡張モジュール。
- エンタープライズ版では、以下のファイルも利用可能になります。
- `jpgraph_windrose.php` – 風配図用のプロット拡張モジュール。
- `jpgraph_odo.php` – オドメーター用のプロット拡張モジュール。
- `jpgraph_barcode.php` – バーコード用のプロット拡張モジュール。
- `jpgraph_pdf417.php` – PDF417 2D のバーコード拡張。

3.4 イメージ フォーマットと外部イメージのライブラリ

デフォルトで、GD イメージ ライブラリは PNG 画像フォーマットをサポートします。JpGraph の動作には、GD ライブラリが必要です。詳細に関しては、PHP のドキュメントを参照してください。GD のバー

ジョンによっては、GIF 形式が扱えないものがあります。そのため、デフォルトでは PNG 形式のサポートが行われています。

JPEG 形式を使用する場合、追加でライブラリをインストールする必要があります。詳細に関しては、PHP のドキュメントを参照してください。ただし、圧縮率の高さなどから PNG 形式を推奨します。JPEG 形式と比較しても、PNG 形式のグラフの方が美しく表示されます。そのため、相応の理由が無ければ PNG 形式以外の形式を使用しない方がいいでしょう。

デフォルトの画像形式は auto に設定されています。これは、PHP がサポートしている画像形式のなかから、JpGraph が自動的に最適な画像形式を使用するものです。PNG、GIF、JPG の順に評価が行われます。

3.5 JpGraph の詳細なインストール

1. 1.x ブランチをインストールしている場合、使用している PHP のバージョンが 4.3.x で、GD ライブラリでコンパイルをサポートしているということを確認してください。GD が完全に作動しているということは重要です。確認には、前セクションに記述されている内容を確認してください。JpGraph では GD2.x をサポートしています。パフォーマンスの面や True Color 画像の扱いなどを考慮すると、GD2.x を用いることを強く推奨します。
2. GD2.x ブランチをインストールしている場合、使用している PHP のバージョンが 5.1 以上で、GD2.x ライブラリでコンパイルをサポートしているということを確認してください。GD2.x ブランチは PHP 4.x に対応していません。
3. ファイルを解凍し、適当なディレクトリにコピーします。
4. jpg-config.inc.php を修正し、キャッシュ ディレクトリと TTF ディレクトリを設定します。Apache がキャッシュ ディレクトリに書き込み可能であることを確認してください。
5. jpggraph.php の最上部にある、定数の定義部分を確認して、お使いの環境に合わせてください。多くの方は、デフォルトの設定で問題ないでしょう。特に、USE_GD2_LIBRARY 設定を確認してください。GD2 がインストールされている場合には true に設定する必要があります。
6. キャッシュ機能を使用する場合、PHP がキャッシュ ディレクトリに書き込み可能であることを確認してください。
7. Windows 環境においては、PHP スクリプトの最終行に改行が付加されてしまう可能性があります (Header already sent エラーが発生します)。この場合、JpGraph のすべてのファイルについて、最後にある改行を取り除いてください。
8. JpGraph の [FAQ](#) をお読みください。

3.6 インストールのトラブルシューティング

ユーザーのうち 99% の方が、何の問題もなくインストールできています。しかし、一部の環境において JpGraph がうまく動作しないこともあります。

多くの場合は、TTF ライブラリのインストールや、GD ライブラリのバージョンが古いことに起因しています。以下の通り手段を踏むことで、解決を進めてください。

1. 背景画像が表示されない場合(黒い立体ボックスが表示される代わりに)、可能性は GD2.x は利用可能ですが、`jpg-config.inc.php` が変更されたのでトゥルーカラーのイメージが無効になったことです。`USE_TRUECOLOR` 定数を有効にしてください。
2. 背景画像に問題がある場合、`USE_GD2_LIBRARY` 定数が間違っている可能性があります。例、GD2 をインストールしていない場合、この定数を `false` に設定する必要があります。
3. Windows 2000 と IIS の環境下で動作させており、“Can't find font” エラーが発生する場合は、UNIX 形式で記述されている TTF フォントへのパスを変更してください。ただし、フォント フォルダへのパスは、`htdocs` からの相対パスではなく、絶対パスで記述する必要があることに注意してください。
4. 画像が表示されなかったり、エラー メッセージがブラウザに送り返されなかった場合、HTTP サーバーの PHP モジュール(例、Apache PHP)に変更があった可能性があります。これは、PHP のインストールが不完全だったり、True Type ライブラリに問題がある可能性があります。これらの種類の問題をトラックする最善の方法は、PHP クラッシュの証拠の HTTP サーバーログ、またはシステム ログを調査することです。他にも、GD ライブラリの自動検出に失敗する可能性があります。
5. Windows 版で IIS を使っている場合、TTF フォントを使うと空白のページになることがあります。この場合は、`jpg-config.php` で `TTF_DIR` パスを手動でセット（通常は `C:\WINDOWS\Fonts`）してください。
6. キャッシュが有効な場合、キャッシュ ディレクトリの許可が正しく設定されているので、Apache/PHP を起動している処理がキャッシュ ディレクトリに書き込みできることを確認してください。
7. TTF フォントが黄色に表示される場合は、FreeType フォント ライブラリにバグがあります。この場合は、PHP と GD を再インストールする以外に方法がありません。

3.7 PHP 4 をコンパイルする

PHP の構成、またはコンパイルについてすべて説明するということではありません。JpGraph でよく起動していることで知られる PHP の構成の例ということです。

以下は、JpGraph で使用する PHP を構成しコンパイルするために使用される標準の構成です。

特別なインストールの必要事項によって、他のオプションは指定される必要がある場合があります、特に外部のライブラリのパスは調整する必要がある場合があります。

```
./configure --prefix=/usr/share \
--datadir=/usr/share/php \
--with-apxs=/usr/sbin/apxs \
--libdir=/usr/share \
--includedir=/usr/include \
--bindir=/usr/bin \
--with-config-file-path=/etc \
--enable-mbstring --enable-mbregex \
--with-pdflib=/usr \
--with-mysql \
--with-ttf-dir=/usr/lib \
--with-freetype-dir=/usr/lib \
--with-gd --enable-gd-imgstrttf --enable-gd-native-ttf \
--with-zlib-dir=/usr/lib \
--with-png-dir=/usr/lib --with-jpeg-dir=/usr/lib --with-xpm-
dir=/usr/X11R6 \
--with-tiff-dir=/usr/lib \
--enable-ftp \
--enable-memory-limit --enable-safe-mode \
--bindir=/usr/bin \
--enable-bcmath -enable-calendar \
--enable-ctype --with-ftp \
--enable-magic-quotes \
--enable-inline-optimization \
--with-bz2 \
--with-iconv
```

4 はじめに : ダイナミック イメージの生成

この章では、ダイナミック イメージ生成して、スクリプトを生成するイメージを呼び出す HTML タグの使用法を示します。PHP について理解していても、コンセプトを理解するために、この章に目を通すことを強く勧めます。

すべて理解していて、MIME タイプのコンセプト、HTTP ストリーム、そして、PHP でダイナミック イメージを生成したとき、“Headers already sent error(ヘッダーはエラーをすでに送信しました)”というエラーメッセージが表示される理由を説明できる場合、この章を飛ばしても大丈夫です。そうでない場合、ぜひこの章を通してください。

4.1 この章の概要

4.1.1 この章について

1. PHP で動的にイメージを生成する方法
2. 特定のイメージ フォーマットの選び方 (例、JPG、PNG、GIF)
3. 生成したイメージを使用する、ブラウザへ送り返す、ファイルに送信する、または後のポスト処理のためイメージ ハンドラを取得する方法
4. JpGraph でフォント (ビットマップ フォント、TTF フォント) を指定する
5. JpGraph で任意のフォントを使用する方法
6. 非ラテン系のフォントの使用法
7. JpGraph でカラーを指定する方法
8. 使用可能な定義済みカラーのリスト
9. キャッシュ機能を使用する

4.1.2 この章で取り扱わない項目

1. JpGraph ライブライアリでグラフを生成する詳細な方法

4.2 画像を出力するには

画像を生成する PHP スクリプトは、通常 HTML の タグで指定する必要があります。たとえば、以下の HTML では PHP スクリプト fig1.php が生成する画像を取り込んでいます。

```

```

“align”、“width”、“height”は明示する必要はありませんが、指定すると画像が完全にブラウザに送られる前に、画像を正確な位置に配置するようになります。

JpGraph ライブライアリは、PNG、GIF、JPEG などの画像形式を正確に判別できるように、必要なヘッダ情報と画像データを自動的に生成し、ブラウザに送信します。その後ブラウザは画像をデコードします。

ただし、画像出力用のスクリプトでは、画像以外の形式を返すことができないことに注意してください。HTML ページ(正確に言えば HTTP ストリーム)は、特定のストリームのヘッダ情報によって定義される1つの MIME タイプだけで成り立ちます。

よくある間違いとして、画像生成用スクリプトの先頭行に空白文字が入っていることがあります。この場合明確なヘッダを受け取っていないので、ブラウザはスクリプトから出力されているデータをテキストデータと認識してしまいます。こうして画像のヘッダ部分がブラウザに送られると、ブラウザはすでに送られてきているデータ ストリームがテキスト ストリームだと認識し、警告を出します。これがあの忌まわしい “Headers already sent error” の原因です。

HTML 内で画像を複数個表示したい場合は、その数だけの タグを用意し、それぞれ画像生成のスクリプトを GET や POST データなどと共に指定する必要があります。

4.2.1 JpGraph ライブライアリを使用する

JpGraph ライブライアリを利用する場合、最低でも 2 種類のファイルをインクルードする必要があります。1 つは基本ライブライアリで、もう 1 つはプロット拡張ライブライアリです。例として、次のスクリプトは基本的なラインプロットを生成する場合に PHP ファイルの先頭に必ず記述しなければならない行です。

```
<?php
include ('jpgraph.php');
include ('jpgraph_line.php');
```

// JpGraph ライブライアリを使用するコード

注意: include の代わりに require を使用する場合もあります。これらの違いはほとんどありません。include() 関数は、実際にインクルードされた文が実行されるときにだけ、ファイルをインクルードします。require() 関数は、指定されたファイルが必ず置き換えられます。詳細は PHP ドキュメントを参照してください。ほとんどの場合、これらは同義です。

4.3 コマンドライン版の PHP を使用する

コマンドライン版の PHP から、画像を直接生成することもできます。ブラウザを使用した方法とまったく同じように画像が生成されますが、例外として HTTP ヘッダーが生成されません。バイナリのイメージデータのみが出力されます。

使用しているコマンドライン版の PHP (CLI 版) のバージョンを確認してください。CGI 版の SAPI を使った場合は、HTTP ヘッダーが生成されます。注意: CGI 版を使用した場合、「-q」オプションがヘッダの生成を禁止する場合があります。

インストールされたバージョンを確認するには、以下のコマンドを実行します。

```
| php --version
```

そうすると以下のような結果が得られます。

```
| PHP 4.3.8 (cli) (built:Aug 29 2004 22:48:10)
| Copyright (c) 1997-2004 The PHP Group
| Zend Engine v1.3.0, Copyright (c) 1998-2004 Zend Technologies
```

ここで重要なのは、

```
| (cli)
```

と書かれた部分です。JpGraph のライブライアリは、どの SAPI API を使っているかを確認し、そのバージョンによってヘッダーの生成方法を調整します。

上記の条件が揃っている場合には、コマンドラインから直接画像を生成できます。その場合は、以下の通り実行します。

```
| $> php myimage.php > image.png
```

イメージ ファイルの拡張子を、生成されるイメージのフォーマットと同じにしてください。

4.4 JpGraph の基本概念と画像の生成

グラフを生成する共通のパターンは

1. スクリプトを作成し、画像、形式、カラーなどを構成する。
2. タグを持つスクリプト（ラッパー スクリプト）を作成し、 タグを用いて HTML ページの適切な場所にグラフを設置する。

もちろん、生成した画像をブラウザで表示するために、画像生成スクリプトをブラウザから呼び出すこともできます。

また GET や POST などの HTTP パラメータを使用して、引数を画像スクリプトに手渡すことも可能です。たとえば、

```

```

これにより、画像の見た目を制御したり、表示させる画像を変更したりすることができます。ただし、この方法で大量のデータを送信することは推奨しません。代わりに、データベースなどから大容量のデータを取得する方法があります。大容量のデータをイメージ スクリプトに送るもうひとつ的方法としては、イメージ スクリプトに POST リクエストを作成する方法があります。

注意:表示される画像はブラウザに強制的に更新させてください。ブラウザの中には、ユーザーが [更新] ボタン(多くの場合 F5)をクリックするまで、リクエストを送信しないものがあります。これが原因で、ユーザーには古い画像が表示されてしまうことがあります。これを回避するための簡単な方法は、スクリプトには影響しないダミーの日付を引数として追加することです。たとえば、

```
| echo '';
```

ブラウザの内部のキャッシュ機能を認識することも重要です。動的に生成された画像で最もよく起こる問題は、画像生成のスクリプト(ファイル)が同じものを表示してしまうことです。これは、データが変更されていないとブラウザが認識てしまい、そして、キャッシュされたデータがあると、ファイルのタイムスタンプが同じ場合キャッシュされたデータを使用するので、新しい GET リクエストが送られることはありません。

画像を生成するスクリプトは次のような構造になります。

```
// ...必要なスクリプトを include する部分  
$graph = new Graph($width,$height...);  
// ...グラフの細部を設定する部分  
$graph->Stroke();
```

JpGraph は完全にオブジェクト指向に基づいており、すべての呼び出しはクラスのインスタンスを用いて行われます。その基本クラスの 1 つとして、グラフすべての基となる Graph() クラスが挙げられます。

Graph() オブジェクトを作成したあとで、プログラム コードを追加していきグラフの細部を構築していきます。

イメージ スクリプトで呼ばれる最後のメソッドは、そのほとんどが Graph::Stroke() メソッドとなります。これによって構築された画像がブラウザに送り返されます。グラフにイメージ マップを使用する場合は、別の表示用関数を使用します。この場合、最後に呼び出す関数は Graph::StrokeCSIM() となります。

加えて、Graph::Stroke() 関数は以下のような場合にも使用します。

- ... 生成したグラフをファイルとして直接保存する。
- ... 画像を処理するために GD イメージ ハンドラにアクセスする。(PDF ファイル内の画像を扱う場合も必要となります。)
- ... すでに生成されている画像をキャッシュ システムを使用してブラウザに送る。

キャッシュ システムは PHP サーバーの負荷を削減するために用意されています。最初の Graph() オブジェクトの呼び出しの時点で、既に画像を作成済みかどうかを確認し、その場合は以前に保存している画像をブラウザへと送信します。キャッシュ システムを利用する場合は、キャッシュのファイル名とキャッシュの存続期間を指定する必要があります。

このマニュアル内の多くのサンプルでは以下の構文が使用されています。

```
$graph = new Graph(300,200,"auto");
```

このサンプルでは、最初の 2 つの引数はグラフの横幅と縦幅を指定しています。3 番目の引数は、キャッシュ ディレクトリに保存するファイル名を指定しています。“auto”は、キャッシュ ファイル名をスクリプトと同じ名前にすることを意味します。ただし、拡張子はグラフの形式に合わせ、.jpg や .png に変換されます。

キャッシュ システムはデフォルトでは無効になっており、“jpg-config.inc.php” ファイル内で適切に設定することで有効になります。

4.5 JpGraph の画像形式を選択する

デフォルトでは、JpGraph は PNG、JPEG、GIF の順に自動的に画像形式を選択します。実際に選択される形式はシステムに依存します。ファイルの形式を明示的に指定するには、以下の 2 通りの方法があります。

1. DEFINE を用いてデフォルトの画像形式を変更する

```
DEFINE("DEFAULT_GFORMAT","auto");
```

2. `SetImgFormat()` メソッドを使用して、スクリプトにて画像形式を設定する。たとえば、スクリプトで JPEG の使用を強制する場合は、以下のとおり記述します。

```
$graph->img->SetImgFormat("jpeg");
```

4.6 画像をブラウザに送る以外の処理方法

ブラウザに画像を送信するのではなくファイルに保存する場合は、`Graph::Stroke()` メソッドで絶対ファイル名を指定します。

```
$graph->Stroke("/usr/home/peter/images/result2002.png");
```

この場合、指定したディレクトリに Apache/PHP がファイルを作成する権限があることを確認してください。

また、ファイル名には特別な意味を表す 2 種類の文字列が定義されています。

- “auto” – 画像をスクリプトのファイル名と同じ名前でスクリプトと同じディレクトリに作成します。この場合、拡張子は正しいものに変換されます。
- `_IMG_HANDLER` – この定数は `jpgraph.php` で定義されています。このファイル名を指定すると、画像ファイルを作成したり、ブラウザに返したりしないようになります。その代わりに、`Stroke()` 関数により GD イメージのハンドルが返却されます。これは JpGraph でサポートしていない画像の操作をする場合に役に立ちます。たとえば、PDF ファイル内で生成された画像を扱う場合に使用できます。

例) `$handle = $graph->Stroke(_IMG_HANDLER);`

4.7 JpGraph でフォントを使用する

JpGraph ではビットマップ フォントと TrueType フォントを使用できます。軸のスケール値には、ビットマップ フォントを使用することを推奨します。この理由は、多くの人にとって可読性が高く、表示速度が高速だからです。TrueType フォントは、ヘッドライン部分や、グラフのタイトルなどに使用してください。デフォルトでは、TrueType フォントはアンチエイリアスを有効にして描画されます。

多くのサンプルでは、TrueType フォントとビットマップ フォントが組み合わされています。

しかしながら、ビットマップ フォントと TrueType フォントの扱いは若干異なります。JpGraph では、その違いの 99.9% を緩和しているため、ユーザーはこれらのフォントをほぼ同等に扱うことができます。

4.7.1 TrueType フォントをインストールする

内部的には、TrueType フォントを描画する場合、TrueType フォントを対象ディレクトリにインストールしておく必要があります。JpGraph では、TrueType フォントがあるパスを `FONT_PATH` 定数 (`jpgraph.php` 内にあります) で指定する必要があります。なお、ここは絶対パスを指定する必要があります。デフォルトでは、以下のパスを使用します。

```
DEFINE("TTF_DIR","/usr/local/fonts/ttf/");
```

JpGraph では、同じフォント ファミリーの斜体字と太字を判断するために、特定の規則でファイル名を命名する必要があります。また、それらのフォントは DEFINE 部分で宣言しておく必要があるので、単に TrueType フォントをディレクトリにコピーするだけでは正しく動作しません。現状では、簡単にフォントを追加することはできません。しかし、将来のバージョンの JpGraph では変更される予定です。

4.7.2 TrueType フォントが動作していることを確認する

TrueType フォントが JpGraph で動作していることを確認するには、以下の GD スクリプトの動作を確認します。ただし、スクリプト中のフォントのパスは、環境に応じて変更してください。

```
DEFINE("TTF_DIR","/usr/X11R6/lib/X11/fonts/truetype
/");
$im = imagecreatetruecolor (400, 100);
$black = imagecolorallocate ($im, 0, 0, 0);
$white = imagecolorallocate ($im, 255, 255, 255);
imagerectangle($im,0,0,399,99,$black);
imagefilledrectangle($im,0,0,399,99,$white);
imagefttext ($im, 30, 0, 10, 40, $black,
TTF_DIR."arial.ttf", "Hello World!");
header ("Content-type: image/png");
imagepng ($im);
```

上記のスクリプトを動作させるには、GD2 ライブラリが必要です。また、スクリプトは PHP のコンパイル状況に応じて、UTF-8、EUC-JP、Shift_JIS のいずれかで動作することを確認する必要があります。

4.7.3 フォントを指定する

すべてのグラフ オブジェクトに対して、SetFont() メソッドを呼び出すことでフォントを指定することができます。その際、以下の 3 個の引数を指定できます。

1. フォント ファミリー、FF_ 形式で定義されます。
2. フォント スタイル、FS_ 形式で定義されます。
3. フォント サイズ、TTF フォントのみ有効です。数値を指定します。

内蔵フォントに関しては、3 番目のフォント サイズは無視されます。これは、JpGraph で使用できる全 3 種類のフォントでは、文字サイズが固定されているからです。JpGraph では、以下のフォント ファミリーを指定できます。

フォント ファミリー	形式	概要
FF_FONT0	内蔵フォント	非常に小さなフォント
FF_FONT1	内蔵フォント	中ぐらいのサイズのフォント
FF_FONT2	内蔵フォント	大きなサイズのフォント
FF_ARIAL	TTF フォント	Arial フォント
FF_VERDANA	TTF フォント	Verdana フォント
FF_COURIER	TTF フォント	固定幅の Courier フォント
FF_BOOK	TTF フォント	Bookman
FF_COMIC	TTF フォント	Comic sans
FF_TIMES	TTF フォント	Times New Roman

FF_GEORGIA	TTF フォント	Georgia
FF_TREBUCHE	TTF フォント	Trebuchet
FF_VERA	TTF フォント	Gnome Vera フォント。http://www.gnome.org/fonts/ からダウンロードできます。
FF_VERAMONO	TTF フォント	Gnome Vera Mono フォント。http://www.gnome.org/fonts/ からダウンロードできます。
FF_VERASERIF	TTF フォント	Gnome Vera Serif フォント。http://www.gnome.org/fonts/ からダウンロードできます。
FF_CHINESE	TTF フォント	中国語用のフォント。
FF_SIMSUN	TTF フォント	中国語用のフォント。
FF_BIG5	TTF フォント	中国語 BIG5 用のフォント (iconv() 関数が必要です)

上記のフォント ファミリーが、すべてのスタイルをサポートしているわけではありません。
以下に、それぞれのフォント ファミリーで利用できるフォント スタイルを紹介します。

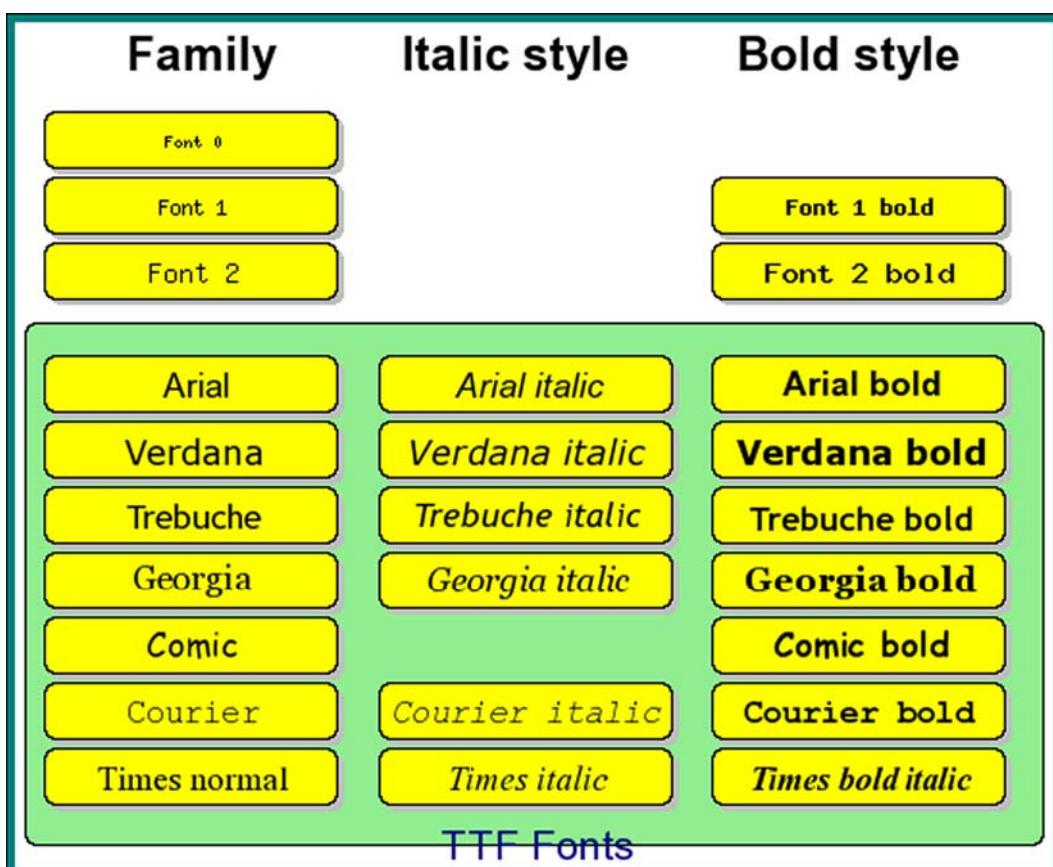


図 1:JpGraph で使用できるフォント一覧

```
<?php
// $Id: listfontsex1.php,v 1.3 2002/10/25 22:44:15 aditu
s Exp $
```

```
include "../jpgraph.php";
include "../jpgraph_canvas.php";
include "../jpgraph_canvtools.php";
```

```
$g = new CanvasGraph(550,450,'auto');
$scale = new CanvasScale($g);
$scale->Set(0,27,0,53);
$g->SetMargin(5,6,5,6);
$g->SetColor('white');
$g->SetMarginColor("teal");
$g->InitFrame();

$t = new CanvasRectangleText();
$t->SetFillColor('lightgreen');
$t->SetFontColor('navy');
$t->SetFont(FF_ARIAL,FS_NORMAL,16);
$t-
>Set("¥n¥n¥n¥n¥n¥n¥n¥n¥n¥n¥n¥nTTF Fonts",0,5,19,26,32
);
$t->Stroke($g->img,$scale);

$t = new CanvasRectangleText();
$t->SetFillColor('');
$t->SetFontColor('black');
$t->SetColor('');
$t->SetShadow('');
$t->SetFont(FF_ARIAL,FS_BOLD,18);

$t->Set('Family',1,1,8);
$t->Stroke($g->img,$scale);

$t->Set('Italic style',9,1,8);
$t->Stroke($g->img,$scale);

$t->Set('Bold style',17,5,1,8);
$t->Stroke($g->img,$scale);

$t->SetFillColor('yellow');
$t->SetFontColor('black');
$t->SetColor('black');
$t->SetShadow('gray');

$c=6;$c=1;$w=7.5;$h=3.5;

$fonts=array(
  array("Font 0",FF_FONT0,FS_NORMAL),
  array("",FF_FONT0,FS_ITALIC),
  array("",FF_FONT0,FS_BOLD),

  array("Font 1",FF_FONT1,FS_NORMAL),
  array("",FF_FONT1,FS_ITALIC),
  array("Font 1 bold",FF_FONT1,FS_BOLD),

  array("Font 2",FF_FONT2,FS_NORMAL),
  array("",FF_FONT2,FS_ITALIC),
```

```
array("Font 2 bold",FF_FONT2,FS_BOLD),

array("Arial",FF_ARIAL,FS_NORMAL),
array("Arial italic",FF_ARIAL,FS_ITALIC),
array(" Arial bold",FF_ARIAL,FS_BOLD),

array("Verdana",FF_VERDANA,FS_NORMAL),
array("Verdana italic",FF_VERDANA,FS_ITALIC),
array("Verdana bold",FF_VERDANA,FS_BOLD),

array("Trebuchet",FF_TREBUCHE,FS_NORMAL),
array("Trebuchet italic",FF_TREBUCHE,FS_ITALIC),
array("Trebuchet bold",FF_TREBUCHE,FS_BOLD),

array("Georgia",FF_GEORGIA,FS_NORMAL),
array("Georgia italic",FF_GEORGIA,FS_ITALIC),
array("Georgia bold",FF_GEORGIA,FS_BOLD),

array("Comic",FF_COMIC,FS_NORMAL),
array("",FF_COMIC,FS_ITALIC),
array("Comic bold",FF_COMIC,FS_BOLD),

array("Courier",FF_COURIER,FS_NORMAL),
array("Courier italic",FF_COURIER,FS_ITALIC),
array("Courier bold",FF_COURIER,FS_BOLD),

array("Times normal",FF_TIMES,FS_NORMAL),
array("Times italic",FF_TIMES,FS_ITALIC),
array("Times bold italic",FF_TIMES,FS_BOLDITALIC),
);

$n=count($fonts);

for( $i=0; $i < $n; ++$i ) {

  if( $i==9 ) $r += 3;

  if( $fonts[$i][0] ){
    $t->SetTxt($fonts[$i][0]);
    $t->SetPos($c,$r,$w,$h);
    $t->SetFont($fonts[$i][1],$fonts[$i][2],13);
    $t->Stroke($g->img,$scale);
  }

  $c += $w+1;
  if( $c > 30-$w-2 ){
    $c = 1;
    $r += 4;
  }
}

$g->Stroke();
?>
```

以下に、使用例を紹介します。

```
$graph->title->SetFont(FF_FONT2);
$graph->title->SetFont(FF_FONT2,FS_BOLD);
$graph->title->SetFont(FF_ARIAL);
$graph->title->SetFont(FF_ARIAL,FS_BOLD,24);
```

4.7.4 JpGraph にフォントを追加する

注意:ここにかかれている情報は高度な知識を持ったユーザーが活用してください。新しいフォントを追加した場合、サポート対象から外れる可能性があります。対応させるには、以下の通りソースコードの修正が必要です。

フォントを追加する場合、以下の手続きを踏んでください。

1. フォント ファミリーに適合する FF_ で始まる定数を定義する。
2. TrueType フォントを入手して、フォント ディレクトリに追加します。フォント ファイルは使用するスタイル別にそれぞれ分けて保存する必要があります。その際 TTF クラス内で使用するファイル名を定義してください。フォントの定義には FF_ で始まる定数を使用してください。

4.7.5 JpGraph で文字列の文字揃えを行う

JpGraph を使用する場合、テキスト文字列の文字揃えは必ずしも必要ではありません。ただし、グラフに多くの文字列を追加したり、直接キャンバスに文字列を描画する場合は、以下の知識を知っておいたほうが良いでしょう。

グラフに追加する文字列は、[Text\(\)](#) オブジェクトとして扱えます。また、文字揃えは[Text::Align\(\)](#) で指定することができます。ここで指定するのは、文字列のアンカー ポイントです。詳しくは、以下のサンプルを参照してください。

下の図は文字揃えの 9 つのパターンを表しています。赤い十字のマークは文字列が配置される場所を示しています。文字揃えはそれぞれ下の図のように使用されます。

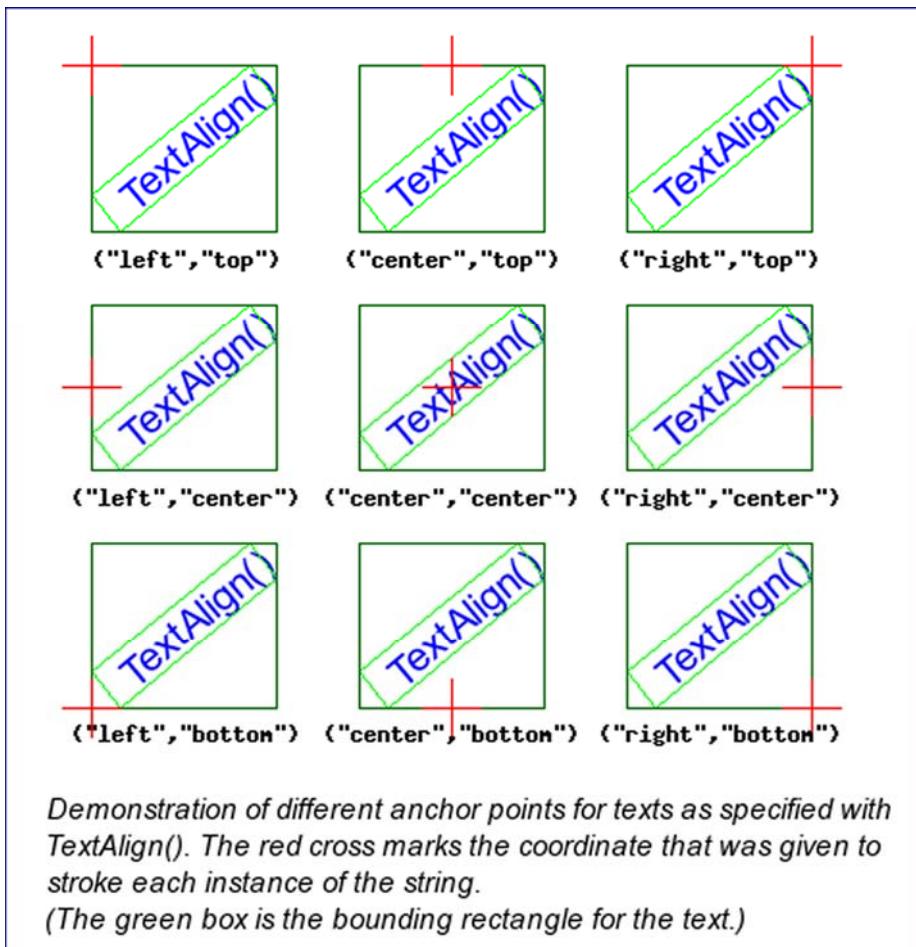


図 2: 文字揃えを指定する

```

<?php
// $Id: textalignex1.php,v 1.1 2002/10/19 17:42:53 aditus Exp $
require_once "../jpgraph.php";
require_once "../jpgraph_canvas.php";

if( empty($_GET['a']) ) {
    $angle=40;
}
else {
    $angle=$_GET['a'];
}

$caption = "Demonstration of different anchor points for texts as specified with TextAlign(). The red cross marks the coordinate that was given to stroke each instance of the string.\n(The green box is the bounding rectangle for the text.)";
$txt=".TextAlign()";

// Initial width and height since we need a "dummy" canvas to
// calculate the height of the text strings
$w=480;$h=50;
$xm=90;$ym=80;

$g = new CanvasGraph($w,$h);
$almg = $g->img;
$almg->setFont(FF_ARIAL,FS_NORMAL,16);

$tw=$almg->GetBBoxWidth($txt,$angle);
$th=$almg->GetBBoxHeight($txt,$angle);

$almg->setFont(FF_ARIAL,FS_NORMAL,11);
$ch=$almg->GetBBoxHeight($caption);

// Calculate needed height for the image
$h = 3*$th+2*$ym + $ch;
$g = new CanvasGraph($w,$h);
$almg = $g->img;

$prof = array('left','top',
             'center','top',
             'right','top',
             'left','center',
             'center','center',
             'right','center',
             'left','bottom',
             'center','bottom',
             'right','bottom');
$n = count($prof)/2;

for( $i=0,$r=0,$c=0; $i < $n; ++$i ) {
    $x = $c*($tw+$xm)+$xm/2;
    $y = $r*($th+$ym)+$ym/2-10;
    $almg->setColor('blue');
    $almg->setTextAlign($prof[$i*2],$prof[$i*2+1]);
    $almg->setFont(FF_ARIAL,FS_NORMAL,16);
    $almg->strokeText($x,$y,$txt,$angle,"left",true);
}

```

```
$almg->SetColor('black');
$almg->SetFont(FF_FONT1,FS_BOLD);
$almg->SetTextAlign('center','top');
$align = sprintf("(%s","%s").$prof[$i*2].$prof[$i*2
+1]);
$almg-
>StrokeText($c*($tw/2+$xm)+$xm/2+$tw/2,$r*($th/2
+$ym)+$th+$ym/2-4,$align);
$c++;
if( $c==3 ) {
$c=0;$r++;
}
```

```
}
```

```
$almg->SetTextAlign('center','bottom');
$almg->SetFont(FF_ARIAL,FS_ITALIC,11);
$almg->StrokeText($w/2,$h-10,$caption,0,'left');

$almg->SetColor('navy');
$almg->Rectangle(0,0,$w-1,$h-1);

$g->Stroke();

?>
```

4.8 JpGraph でカラーを指定する

カラーは 3 種類の方法で指定できます。

1. 400 個の定義済みのカラー名を使用できます。

```
SetColor("yellow:1.2");
```

定義済みのカラー名は係数を用いて調節することができます。定数は $0 < f < 1$ でより暗く、0 あるいは 1 で元のカラー、 $f > 1$ でより明るくなります。値 > 1 は色を明るくします 以下にサンプルを掲載します。

```
SetColor("khaki:0.5"); // "khaki" の暗いバージョン
SetColor("yellow:1.2"); // "yellow" の明るいバージョン
```

2. RGB 形式で指定できます。以下にサンプルを掲載します。

```
SetColor(array(65,100,176));
```

3. カラーを 16 進数形式で指定できます。以下にサンプルを掲載します。

```
SetColor("#A16BFF");
```

4.8.1 透明度を調節する

JpGraph 1.10 からは、GD2.x からサポートされた透明度の設定が可能です。

明度の設定をするには、カラーの設定時に @ 文字に続いて数値を指定します。

たとえば、40% の透明度を持った赤色を指定する場合、以下の通りとなります。

```
SetColor("red@0.4");
```

あるいは、90% の透明度を持った赤色を指定する場合、以下の通りとなります。

```
SetColor("red@0.9");
```

背景イメージを持つバー グラフが使用できる透過の例

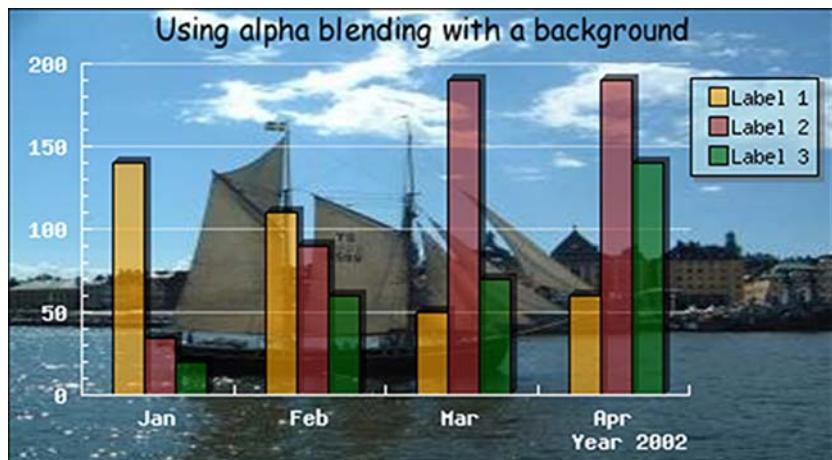


図 3: ラインプロットをバー プロットに追加する

```

<?php

include ("./jpgraph.php");
include ("./jpgraph_bar.php");

// Some data
$datay1=array(140,110,50,60);
$datay2=array(35,90,190,190);
$datay3=array(20,60,70,140);

// Create the basic graph
$graph = new Graph(450,250,'auto');
$graph->SetScale("textlin");
$graph->img->SetMargin(40,80,30,40);

// Adjust the position of the legend box
$graph->legend->Pos(0.02,0.15);

// Adjust the color for the shadow of the legend
$graph->legend->SetShadow('darkgray@0.5');
$graph->legend->SetFillColor('lightblue@0.3');

// Get localised version of the month names
$graph->xaxis->SetTickLabels($gDateLocale->GetShortMonth());

// Set a nice summer (in Stockholm) image
$graph->SetBackgroundImage('stship.jpg',BGIMG_COPY);

// Set axis titles and fonts
$graph->xaxis->title->Set('Year 2002');
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetColor('white');

$graph->xaxis->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->SetColor('white');

$graph->yaxis->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->SetColor('white');

// $graph->grid->Show(true);
// $graph->grid->SetColor('white@0.5');

// Setup graph title
$graph->title->Set('Using alpha blending with a background');
// Some extra margin (from the top)
$graph->title->SetMargin(3);
$graph->title->SetFont(FF_COMIC,FS_NORMAL,12);

// Create the three var series we will combine
$bplot1 = new BarPlot($datay1);
$bplot2 = new BarPlot($datay2);
$bplot3 = new BarPlot($datay3);

// Setup the colors with 40% transparency (alpha channel)
$bplot1->SetFillColor('orange@0.4');
$bplot2->SetFillColor('brown@0.4');
$bplot3->SetFillColor('darkgreen@0.4');

// Setup legends
$bplot1->SetLegend('Label 1');
$bplot2->SetLegend('Label 2');
$bplot3->SetLegend('Label 3');

// Setup each bar with a shadow of 50% transparency
$bplot1->SetShadow('black@0.4');
$bplot2->SetShadow('black@0.4');
$bplot3->SetShadow('black@0.4');

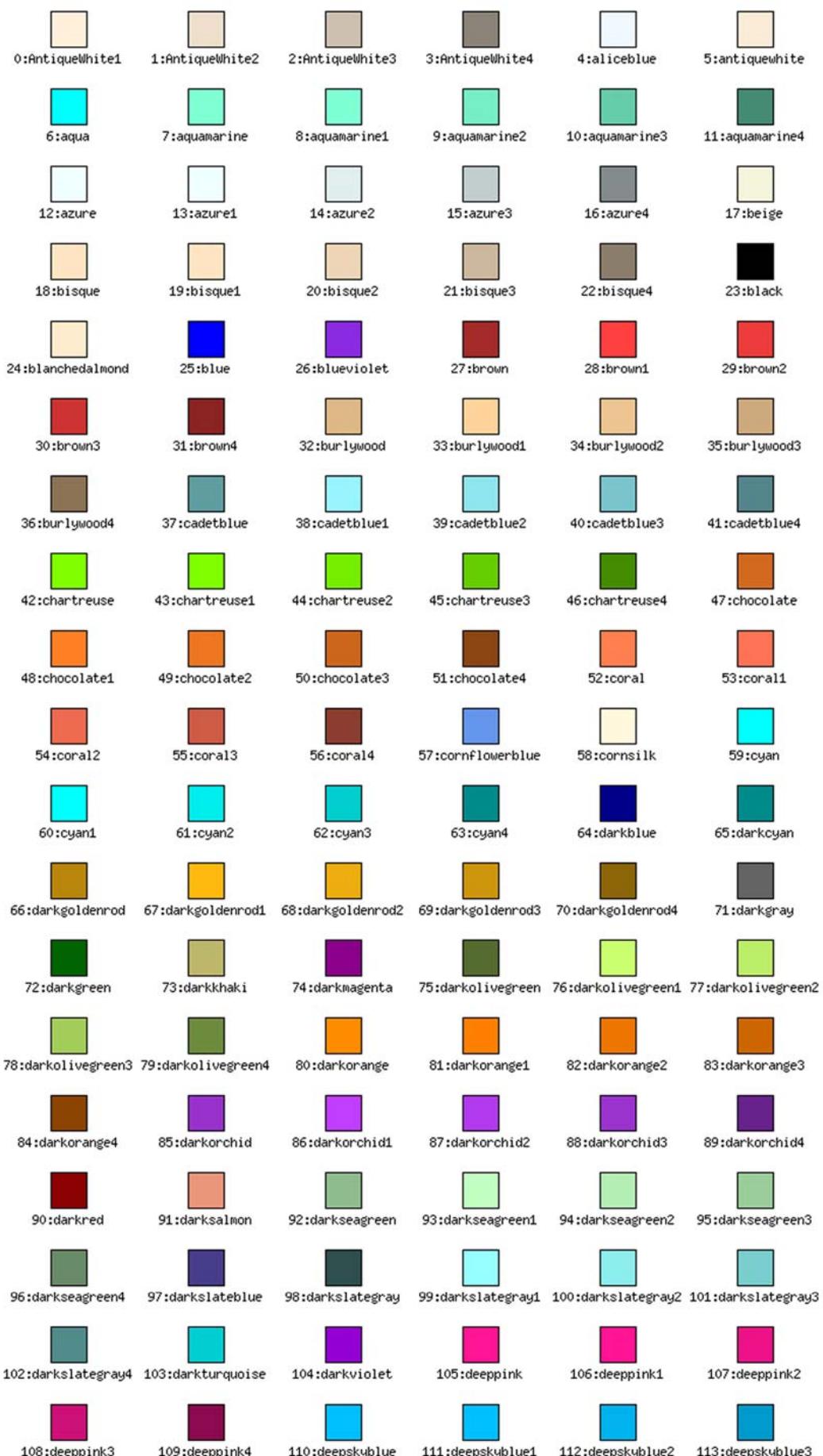
$gbarplot = new GroupBarPlot(array($bplot1,$bplot2,$bplot3));
$gbarplot->SetWidth(0.6);
$graph->Add($gbarplot);

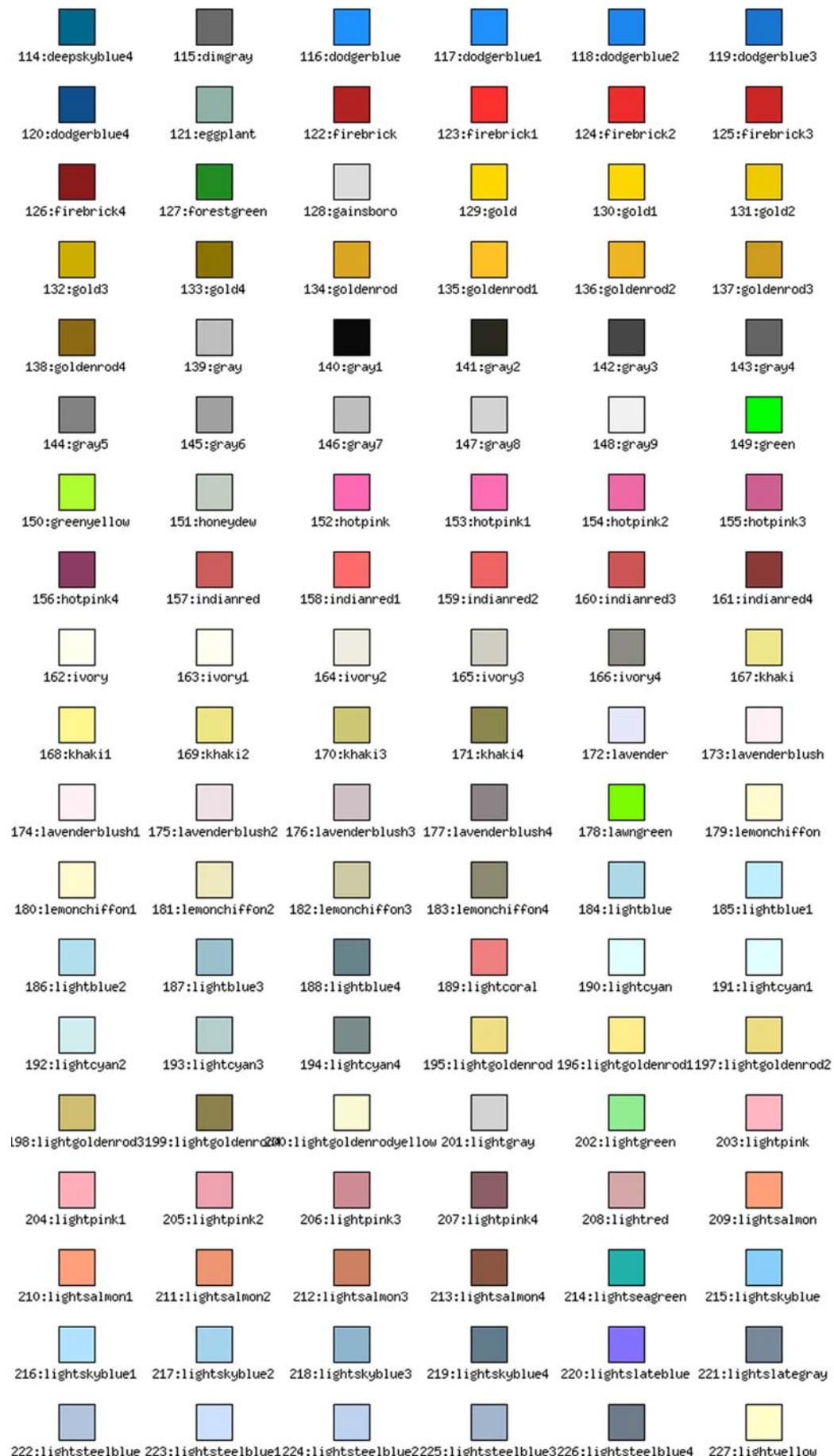
$graph->Stroke();
?>

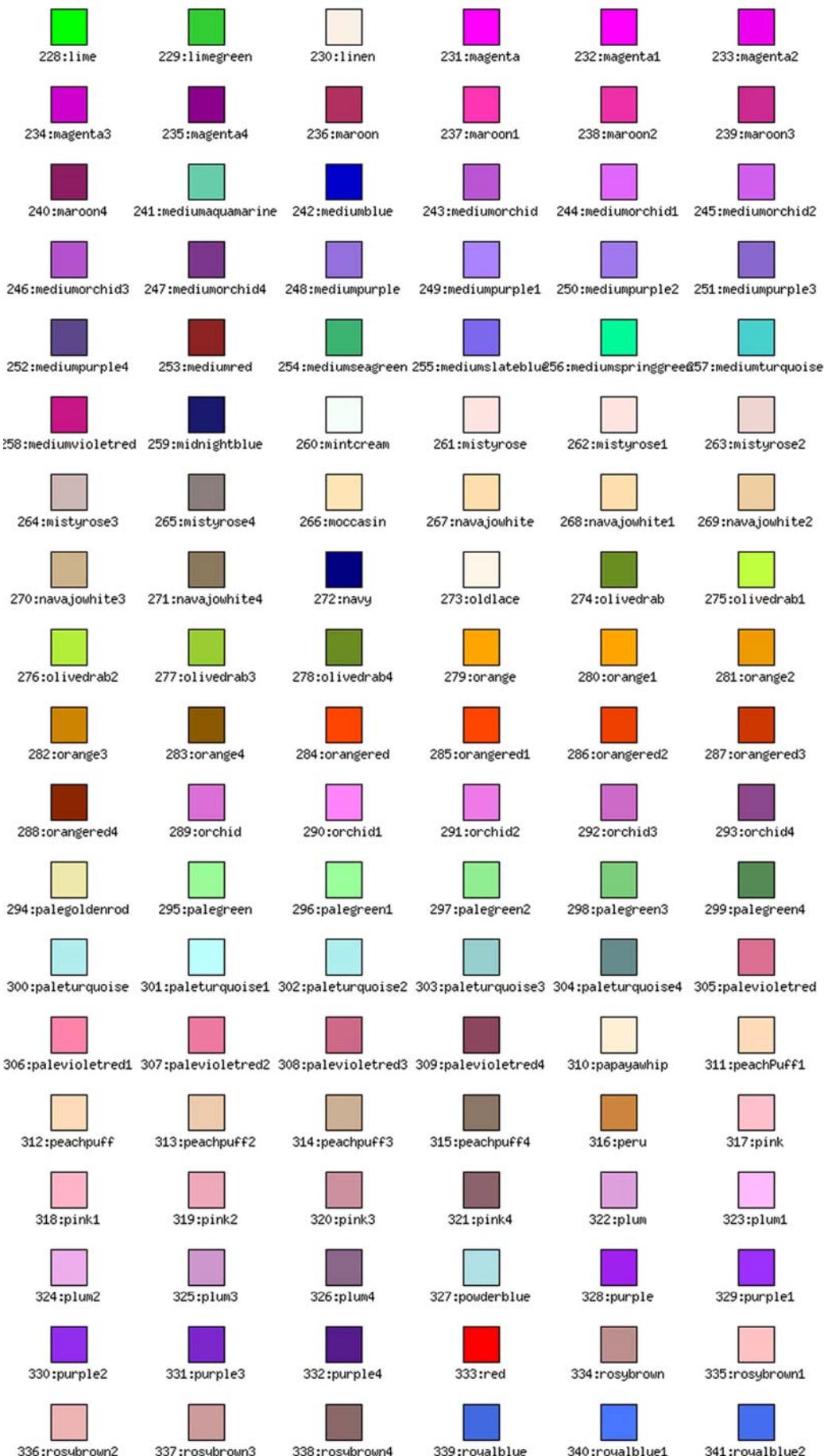
```

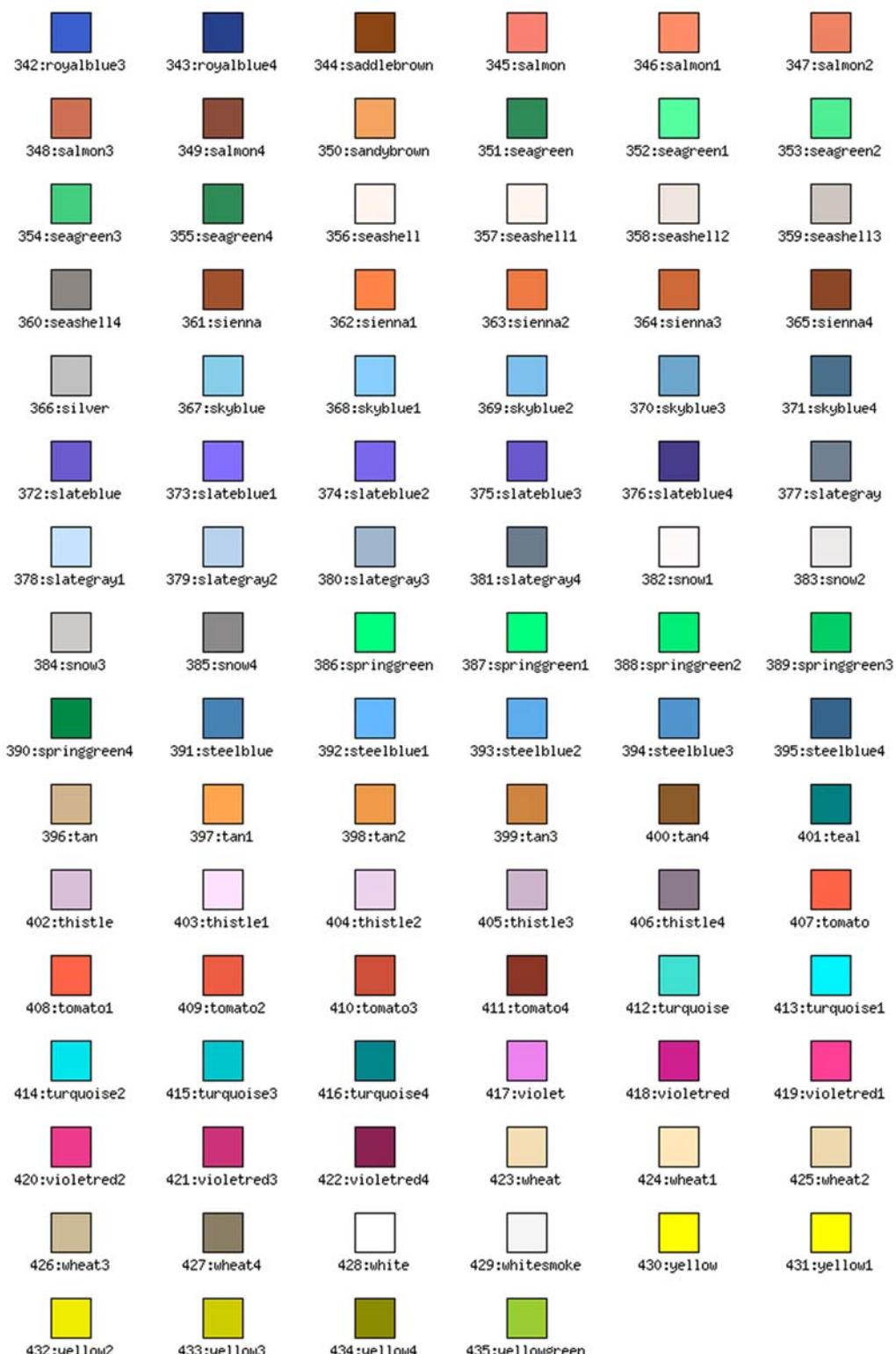
4.8.2 定義済みのカラー

以下に、定義済みのカラーをすべて掲載します。



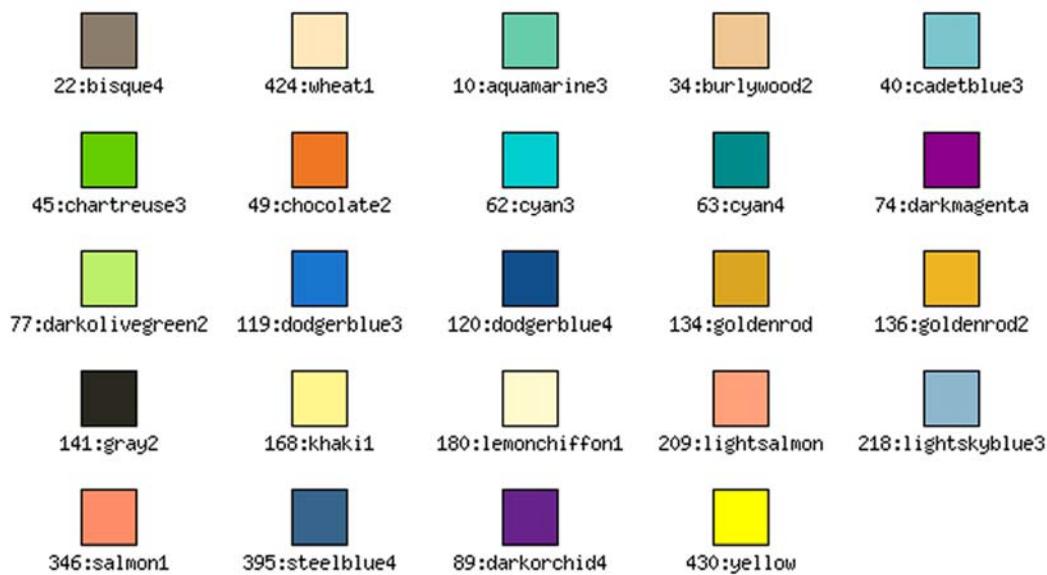






4.8.3 円プロット用のテーマ カラー

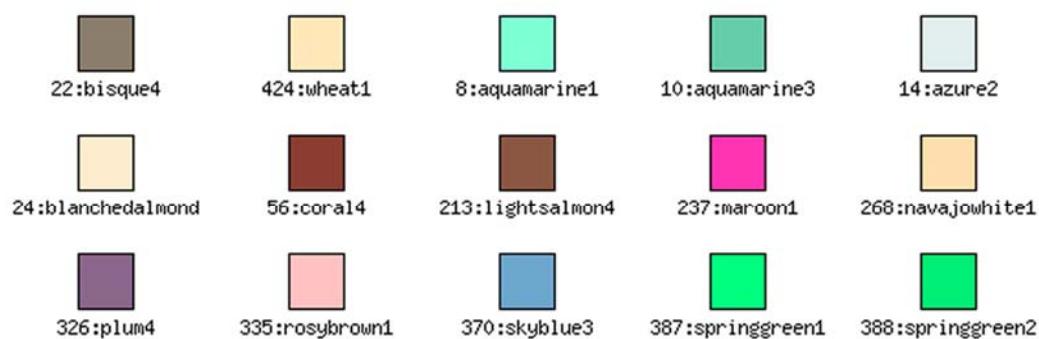
円プロットのカラーを設定する異なるテーマの使用法は、“Working with 2D & 3D pie plots” を参照してください。



テーマ 1: アース



テーマ 2: パステル



テーマ 3: ウオーター



テーマ 4: サンド

5 キャッシュシステムを使用する

Web サーバーの負荷を削減するため、JpGraph では毎回画像スクリプトを実行することを避けるキャッシュシステムを提供します。

画像スクリプトの複雑さによりますが、キャッシュシステムにより大幅な描画速度の向上が見込めます。

しかしながら、リアルタイムで描画処理が必要なグラフ（呼ばれるたびに更新が必要なグラフなど）の場合にはキャッシュシステムの利用が適しません。

5.1 キャッシュシステムを有効にする

キャッシュシステムを有効にするには、`jpgraph.php` の 2 個の定義を使用します。

```
DEFINE("USE_CACHE",true);
DEFINE("READ_CACHE",true);
```

`USE_CACHE` 定数は、キャッシュシステムを使用する際は必ず `true` に設定する必要があります。また、`READ_CACHE` 定数は一般的には変更する必要がないでしょう。

`READ_CACHE` 定数は、JpGraph がキャッシュを確認するかどうかを設定します。`READ_CACHE` 定数を `false` にして、`USE_CACHE` 定数を `true` に設定した場合、画像を必ず生成するようになります。この設定は、主に新しい画像が常にキャッシュディレクトリに保存されるために必要です。

キャッシュを有効にする場合、適切なキャッシュディレクトリを設定する必要があります。キャッシュディレクトリは以下の通りに指定できます。

```
DEFINE("CACHE_DIR","/tmp/jpgraph_cache/");
```

もちろん、ディレクトリ設定は変更できます。ただし、キャッシュ・ディレクトリは Apache が必ず書き込み可能になっている必要があります。

5.2 スクリプトでキャッシュを使用する

キャッシュスクリプトを使用する場合、画像を保存するキャッシュファイル名を指定する必要があります。また、キャッシュの存続期間を設定するタイムアウト値を指定することもできます。

これらの引数は、最初に呼び出す `Graph()` メソッドにて手渡します。手動で使用するファイル名を指定するのではなく、`auto` と指定するとスクリプト名を使用できます。`auto` が指定された場合、キャッシュファイル名はスクリプト名と同じ名前を持ち、適切な拡張子が付加されます。

ファイル名を指定しない場合、`USE_CACHE` 定数を `true` にしている、いないに関係なく、キャッシュ機能は働きません。

以下に、キャッシュを有効にするサンプルを掲載します。

```
$graph = new Graph(300,200,"auto",60);
```

上記のコードでは、自動でキャッシュファイル名を設定し、60 分間のタイムアウト値を設定しています。

そこで、これらすべてはどう作動するのでしょうか。

スクリプトが最初に呼び出された際には、スクリプト全体が実行され、生成された画像がブラウザに転送されます。そして、キャッシュ機能が有効な場合は、画像がキャッシュディレクトリに転送されます。

次にスクリプトを呼び出した際には、キャッシュディレクトリにキャッシュファイルが既に存在するかどうかを確認します。その場合、画像が古すぎないかどうか、指定されたタイムアウト値で確認されます。タイムアウト値よりも新しい場合は、スクリプトを実行するのではなく、画像ファイルからブラウザへと直接転送されます。

そのため、キャッシュディレクトリ内に画像が見つかった場合、最初の `Graph()` メソッドの呼び出し以後スクリプトは実行されません。

この設計は、キャッシュを使用する場合、それ以外の動作を行わないようにするために必要です。これらはすべて自動的に行われます。

5.3 クライアントサイドイメージマップを使用する

JpGraph は、クライアントサイドイメージマップ(CSIM)機能を備えています。この場合、先述のキャッシュ機能とは少し異なり、画像とイメージマップの両方をキャッシュしておく必要があります。また、HTML 側の加工も行う必要があります。CSIM HTML パラダイム方法で変更の必要があります。標準のキャッシュ機能との 2 つの大きな違いは、以下の 2 つです。

- 過去にキャッシュされた画像が設定したキャッシュディレクトリに**保存されません**。詳しくは以下の記述を参考してください。
- キャッシュを確認するには、`CheckCSIMCache()` メソッドを使用する必要があります。

単純な CSIM 画像の場合、パフォーマンス効率は約 50%程度です。しかし、データベースの呼び出しや他の複雑な処理を回避できるため、キャッシュが使用可能できる場合には 1000% ほどの速度向上につながります。

CSIM の詳細については、クライアント サイド イメージマップを使う を参照してください。

CSIM を使用する場合、スクリプトは通常使用する [Graph::Stroke\(\)](#) メソッドの代わりに、[Graph::StrokeCSIM\(\)](#) メソッドを使用します。

CSIM でキャッシュ機能を使用する場合、[Graph::CheckCSIMCache\(\)](#) メソッドを呼び出す必要があります。通常のキャッシュの使用方法と同様に、任意の タイムアウト値とキャッシュ画像の名前を指定します。タイムアウトの標準値は、他で指定されなければ 60 分となります。

キャッシュ画像の名前には、もう少し説明が必要です。ここには、関係のある名前を指定します。例えば、myimage や firstpage/image3 などです。JpGraph のインストール状態にもよりますが、これは最終的に CSIMCACHE_DIR で定義されたディレクトリで終了します。これは通常のウェブサーバーがアクセス可能なディレクトリでなくてはなりません。標準設定では、csmcache ディレクトリが画像生成のスクリプトと同じディレクトリに作成されます。

スクリプトが実行されるディレクトリは、PHP が書き込み可能である必要があります。これを解決する最善の方法は、PHP が書き込み可能なディレクトリの数を最小に抑え、標準のキャッシュディレクトリを再利用することです。しかしこの方法の場合、HTTP サーバーがドキュメント ルートにアクセス可できるよう、アドミニストレーターがキャッシュディレクトリを設定する必要があります。

`CheckCSIMCache()` メソッドは、存在するキャッシュのバージョンを確認し、もし見つかれればそれを返してスクリプトの実行を停止します。キャッシュが見つかった場合は、スクリプトの実行をできるだけ抑えて画像を作成します。そのため、この関数は `Graph()` オブジェクトの作成の直後、そして処理の重い作業の前に呼び出します。

CSIM とキャッシュを使用する一般的なスクリプトの構造は、以下のようになります。

```
$graph = new Graph(400,300);

// キャッシュをチェックする、10 分間のタイムアウト
$graph->CheckCSIMCache("image1",10);

// !! キャッシュが存在する場合は、ここで終了 !!

//
// ... データベースの呼び出しなどで、画像を生成
//

$graph->StrokeCSIM();
```

キャッシュを使用しない場合は、`StrokeCSIM()` 関数の引数には何も指定する必要がないことに注意してください。

注意: CSIM キャッシュ機能では、キャッシュ ディレクトリに 2 種類のファイルを保存します。それぞれ、画像と HTML ファイルのイメージ マップに相当するファイルです。

5.4 最後に

- キャッシュのタイムアウトを無期限に設定したい場合は、タイムアウト値を 0 に設定するか、タイムアウト引数を空白にしてください。画像を再度生成する場合は、キャッシュ ディレクトリから画像ファイルを削除する必要があります。この削除作業は、毎晩のバッチに処理させてもよいでしょう。
- アンチエイリアス機能を有効にする場合は、一般的な描画と比較して処理時間がかかるため、キャッシュ機能を有効にすることを推奨します。

5.5 すべてのグラフで共通の機能

JpGraph の Graph オブジェクトを基にしたグラフで共通な機能を紹介します。

5.5.1 クリッピング

デフォルトでは、プロット エリア外のすべてのプロットにはクリッピング処理が行われます。そのため、手動でグラフ スケールを設定し、そのスケールよりも小さい、あるいは大きい値を持つプロットを指定された場合、それらの値は表示されません。

ライン プロットなどで、データ プロットがグラフの途中でとぎれてしまう場合などにも、クリッピング機能は正常に動作します。使用するアルゴリズムはデータ ポイントの番号 0(1) です。

クリッピングの有効、無効を設定するには、[Graph::SetClipping\(\)](#) を参照してください。

5.5.2 共通のプロパティ

1. Graph::title、Graph::subtitle、Graph::subsubtitle プロパティを使用して、3 種類のタイトルを指定できます。
2. Graph::legend プロパティを使用して、領域の設定ができます。
3. Graph::footer::left、Graph::footer::center、Graph::footer::right プロパティを使用してフッターの設定ができます。
4. Graph::xaxis、Graph::yaxis、Graph::y2axis プロパティを使用して軸の設定ができます。
5. Graph::xgrid、Graph::ygrid、Graph::y2grid プロパティを使用してグリッドの設定ができます。

5.5.3 共通のプロパティ

1. Graph::Add() メソッドを使用して、プロット オブジェクト（棒プロット、円プロット、テキスト、バンド、ライン）を追加できます。
2. Graph::SetMargin() メソッドを使用して、マージンの設定ができます。
3. Graph::SetColor() メソッドを使用して、プロット エリアにカラーを塗ることができます。
4. Graph::SetBox() メソッドを使用して、プロット エリアのボックスの設定ができます。
5. Graph::SetMarginColor() メソッドを使用して、指定したマージンのカラーを設定できます。
6. Graph::SetFrame() メソッドを使用して、フレームを設定できます。
7. Graph::SetShadow() メソッドを使用して、ドロップ シャドウを設定できます。
8. Graph::SetGridDepth() メソッドを使用して、グリッド ラインの設定（プロットとの前後関係）ができます。
9. Graph::SetAngle() メソッドを使用して、任意の角度にプロットを回転できます。
10. Graph::SetBackgroundImage() メソッドを使用して、背景画像を追加できます。
11. Graph::SetAxisStyle() メソッドを使用して、軸の外見を設定できます。

6 イメージ マップを使用する

イメージ マップ、またはサイド マップ (CSIM) は JpGraph でサポートされています。“drill-down” グラフのセットの構築を許可するグラフで、ホットスポットを作成する機会を与えます。

以下のセクションは、読者が HTML でクライアント側のイメージ マップの基本コンセプトを理解しているという仮定が基準となっています。

簡単に概説すると。クライアント側のイメージマップは、2 つのパートから成ります。最初のパートは実際のイメージで、2 つめのパートはホット スポットでマークされるイメージ エリアを調節するマップです。ライブラリは自動的にこれらの調節されたマップをグラフから生成します。

至る所で、ホットスポットとして使用されるグラフの手動エリアは、一般の記述エリアと連結します。

6.1 イメージ マップ スクリプトの基本構造ゲット

クライアント側のイメージ マップを使用する HTML ページの標準構造は、以下のラインを使用しています。

```
// "mapname" を持つイメージマップの仕様
<MAP NAME=...>
... specification ...
</MAP>
```

```
// イメージ タグ  

```

興味深い質問を持ち出します。

普通、 タグで直接グラフのスクリプトを呼ぶので、この “HTML ラッパー” スクリプトでどのようにイメージ マップ(イメージ スクリプトでのみ可能な)を維持しますか？

JpGraph には、これを解く 2 つの解決策があります。

1. 標準の Graph::Stroke() メソッドの代わりに、変更した Stroke() method Graph::StrokeCSIM() を使用する “builtin” 方法を使用してください。
2. より複雑なコード化を犠牲にしてあなたに素晴らしいコントロールを提供する Graph::GetHTMLImageMap() を直接使用してください。

最初の方法は変更したストロークを使用するので、イメージを返す前に(標準の Stroke() メソッドのように)、StrokeCSIM() はイメージ マップの仕様と正しい タグの両方を含む HTML ページを返します。

非 CSIM イメージと異なるイメージ スクリプトを返すイメージ マップを処理する必要があるということを意味しています。たとえば、イメージ マップと一緒にいイメージ タグを含む実際の HTML ページを送るので、 タグの “src” のターゲットとして、直接使用することができません。

6.2 イメージ マップ プロットにターゲットを指定する

標準のイメージ スクリプトを CSIM スクリプトに変えるために、最初にするべきことは、イメージのホットスポットに適切なターゲットの URL を与えることです。

ホットスポットが表すものはプロット タイプによります。以下のプロット タイプとグラフのエリアはイメージ マップをサポートします。

- 線 プロット。記号はホットスポットです。
- 散布プロット。記号はホットスポットです。
- 円プロットと 3D 円プロット。それぞれがホットスポットです
- バー グラフのすべてのタイプそれぞれがホットスポットです
- 説明
- テキスト文字、たとえば、タイトルや軸のタイトル

それぞれのホットスポットのリンクを特定するため、ホットスポットのあるグラフでそれぞれのプロット(または、特定のエリア)に [SetCSIMTargets\(\)](#) メソッドを使用します。

このメソッドには 2 つの引数があります

1. \$aTargets、有効な URL ターゲットの配列各ホットスポットの URL、たとえば、10 のバー プロットの値がある場合、10 個の URL が必要になります。たとえば、SetCSIMTarget() をテキストに適用した場合、当然 1 つの URL だけが特定されます。
2. \$aAlts、有効な alt-texts の配列多くのブラウザ(しかし、すべてではありません)が、マウスがホットスポット上にとどまった場合、このテキスト文字を表示します。

6.3 StrokeCSIM() を使用する

CSIM イメージの一番簡単な作成方法は、StrokeCSIM() メソッドを使用します。前に説明したとおり、このメソッドはイメージ マップの仕様と同様にイメージタグを含む（小さい）HTML ページを実際に返します。標準のイメージタグ src プロパティではこのメソッドで終わるスクリプトを使うことができません。

CSIM(または、イメージ マップを維持する)を作成する 2 つの方法があります。

1. 標準のアンカー リファレンスでターゲットとして CSIM イメージ スクリプトを使用します、たとえば


```
| <a href="mycsimscript.html">
```

イメージだけを含むイメージ ページには欠点があります。
2. もう 1 つの方法は、標準の “include” php ステートメントを使っている HTML ページでイメージ スクリプトを含むだけで、任意の HTML ページに含まれるイメージ スクリプトを許可します。たとえば、

```
| <h2> This is an CSIM image </h2>
| <?php
| include "mycsimscript.php"
| ?>
```

注意:同じページに複数の CSIM イメージがある場合、同じページにファイルが複数回含まれる、または何度か“既に定義されたエラー”が表示されるの、“jpgraph.php”を含むスクリプトで“include_once”を使う必要があります。

Stroke() を StrokeCSIM() に置き換えるプロセスは困難です。StrokeCSIM() に呼ぶのに等しい Stroke() への既存の呼び出しをすべて交換してください。

違いは、StrokeCSIM() メソッドで最小のファイル名を与える必要があるということのみです。最初の引数は、拡張子を含む実際のイメージ スクリプトの名前を使用します。たとえば、イメージ スクリプトは書き込み必要な “mycsimscript.php” と呼ばれます。

```
$graph->StrokeCSIM('mycsimscript.php')
```

ただし、ここでは小さな “trick” の適用が可能です。PHP は、常に現在のファイル名に設定される、 “_FILE_” と呼ばれる特別な変数を維持します。これは、以下のコンストラクションで使用できるということです。

```
$graph->StrokeCSIM(basename(_FILE_))
```

スクリプトは、別な方法で必要なファイルのコードを変更せずに名前を変更できるので、これはより良い方法です。

注意:なぜ、スクリプト名は最初のパラメーターとして使われるのですか？理由は、送られる HTML ページの生成で、イメージタグのスクリプトに関する必要があるからです。それなら、なぜ PHP_SELF リファレンスを使用することができないのですか？PHP_SELF での問題は、HTML ページにイメージ・スクリプトがあり、PHP_SELF を使用する場合、HTML ページの名前を取得し PHP_SELF で実際のスクリプトが使用されないということです。また、コンテキストで __FILE__ が “jpgraph.php” に設定されないので、ライブラリで __FILE__ トリックを使用することができません。したがって、上に示したように、クライアントはこれを指定しなければなりません。

`StrokeCSIM()` の他の引数はオプションです。同じ HTML ページで複数の CSIM イメージを使用する場合、すべてのイメージがそれぞれのイメージに対してそれぞれのイメージマップに適切に調和するためユニークでなければならないので、2 つ目のパラメーターとしてイメージマップの名前を指定する必要があるということに注意してください。[StrokeCSIM\(\)](#) の詳細に関してはクラスリファレンスを参照してください。

6.4 イメージマップの例

Example/ ディレクトリーには、イメージマップの様々なタイプを設定するいくつもの例があります。以下の例は現在使用可能です。

- bar_csimex1.php
- bar_csimex2.php
- bar_csimex3.php
- barline_csimex1.php
- barlinefreq_csimex1.php
- boxstockcsimex1.php
- ganttcsimex01.php
- ganttcsimex02.php
- imgmarker_csimex1.php
- pie3d_csimex1.php
- piec_csimex1.php
- pie_csimex1.php
- scatter_csimex1.php
- titlecsimex01.php

追加の引数 “t=2” を持つ testsuit.php を呼ぶことで、これらすべての例に簡単にアクセスできます。[testsuit.php?t=2](#) リンクで、別々のウィンドウが可能な CSIM の例と一緒に開きます。

6.5 StrokeCSIM() の動き

StrokeCSIM() の作動方法の正確な技術的詳細の知識は、おそらく多くの人によって必要ではありません。しかし、完全に対して、この短いセクションでそれらの詳細について概説します。

解決しなければならない根本的な問題は、2 つのモードでイメージ スクリプトを呼ぶことができなければならないということです。ユーザがイメージ スクリプトを持っている場合、StrokeCSIM() メソッドは HTML ページを返すはずですが、イメージ スクリプトがイメージ タグに直接後で呼ばれた場合、実際のイメージ以外 HTML ページを返しません。

この問題は、イメージ・タグでイメージ スクリプトの名前を使った場合、自動的に渡される 1 つの HTTP 引数を使うことで解決します。

生成された HTML を判断する場合、イメージ タグの src・プロパティの引数は単にスクリプト名ではありませんが、追加の引数ではスクリプト名であるということを理解します。

JpGraph の内部のコードで、このあらかじめ定義された引数は確認され、それが存在する場合、HTML ページではなくイメージが送られます。

この引数の名前は JpGraph の DEFINE() ステートメントによって定義されます。その定義は _CSIM_DISPLAY です。

6.6 イメージ マップの取得

ディスクにイメージを保存し、それをイメージ・タグで使用したい場合、イメージ タグを取得する必要があります。このため、関数 [Graph::GetHTMLImageMap\(\)](#) を使用する必要があります。

この使用例は以下のとおりです。これらのラインはイメージをファイルに書き込みます。そこで、スクリプトはクライアント側のイメージ マップを含む HTML ページを以前保存されたファイルを訂正するイメージ・タグを返します。

```
$graph->Stroke("/usr/local/httpd/htdocs/img/image001.png");
echo $graph->GetHTMLImageMap("myimagemap001");
echo "<img src=\"$img/image001.png\" ISMAP USEMAP=\"$#myimagemap001\" border=0>";
```

6.7 イメージ マップとキャッシュ システム

バージョン 1.9 では、キャッシュ システムは CSIM マップで含むように拡張されています。それぞれの CSIM グラフでは、2 つのファイルは、キャッシュ、実際のイメージ マップを持つラッパーの HTML と同じようにイメージ ファイルに保存されます。詳しい情報に関しては、5章 “キャッシュ システムを理解する” を見てください。

7 直角の XY プロットを使用する

この章での目的は、様々なタイプの基本プロットを精製する JpGraph でスクリプトを作成する基本コンセプトを紹介することです。テキスト全体で、表示されたイメージをキャプションで “[ソース]” リンクをクリックすることで、すべてのグラフの正確なソースを表示することができます。これは、別々なウィンドウでソースと一緒にイメージを開きます。この方法は、イメージを生成したスクリプトで実際のイメージの比較を簡単にします。

7.1 ライン プロット

最初の例は、10 個の Y 値でライン グラフにしたものです。この最初の例で、すべてのコードを掲載しています。以下の例では、コードの一部のみを掲載します。

```
(File:example0.php)
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

// データの用意
$ydata = array(11,3,8,12,5,1,9,13,5,7);

// グラフを作成。以下の 2 種類の呼び出しが必ず必要です
$graph = new Graph(350,250,"auto");
$graph->SetScale("textlin");
```

```
// リニア プロットを作成
$lineplot=new LinePlot($ydata);
$lineplot->SetColor("blue");

// プロットをグラフに追加
$graph->Add($lineplot);

// グラフを表示
$graph->Stroke();
?>
```

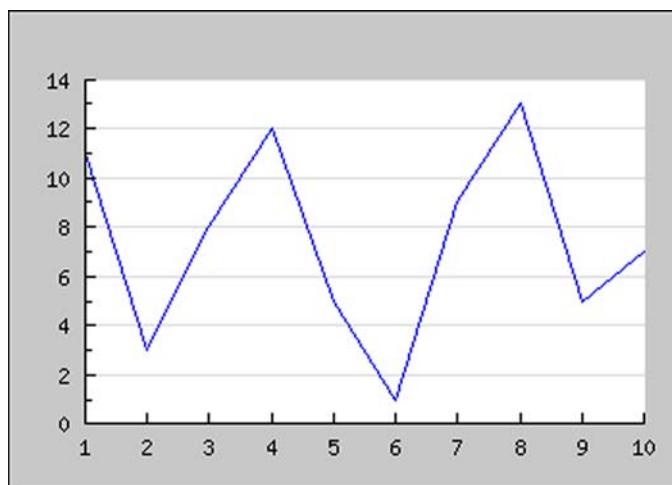


図 4:簡単なライン プロット

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

// Some data
$ydata = array(11,3,8,12,5,1,9,13,5,7);

// Create the graph. These two calls are always required
$graph = new Graph(350,250,"auto");
$graph->SetScale("textlin");
```

```
// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot->SetColor("blue");

// Add the plot to the graph
$graph->Add($lineplot);

// Display the graph
$graph->Stroke();
?>
```

いくつかのこと注意します。

- X 軸と Y 軸のスケールは自動的に設定されます。後ほど、表示されるいくつもの目盛り幅(Tick)を決定する自動スケールの詳細を学習します。
- デフォルトでは、Y 軸は薄いカラーで描画されます。
- デフォルトでは、画像に枠線がつき、余白は灰色になります。
- デフォルトでは、Y 軸の 0 ラベルは表示されません。

このグラフの見栄えを良くしてみましょう。JpGraph では、たとえば以下のようなものを追加できます。

- グラフのタイトル
- 軸のタイトル
- タイトルや軸に対応して余白を広げる

先ほどのサンプルの通り、JpGraph ではすべてのオブジェクトはプロパティを通じて指定できます。オブジェクトには、Graph() や LineProt() などがあります。一般的に、グラフに表示されているすべてのオブジェクトはインスタンス名を通じてアクセスできます。

たとえば、グラフのタイトルは Graph::title プロパティを通じて指定できます。グラフのタイトルを指定するには、title プロパティの Set() メソッドを使用します。

```
$graph->title->Set('Example 2');
```

以下のグラフを表示するには、先ほどのコードに数行を追記します。

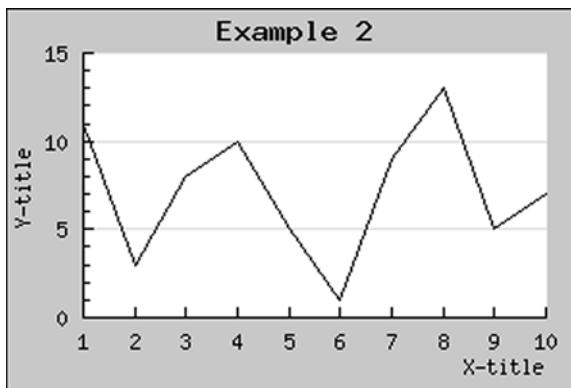


図 5: グラフと軸にタイトルを付加する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

$ydata = array(11,3,8,10,5,1,9,13,5,7);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Setup margin and titles
$graph->img->SetMargin(40,20,20,40);

$graph->title->Set("Example 2");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

// Create the linear plot
$lineplot=new LinePlot($ydata);

// Add the plot to the graph
$graph->Add($lineplot);

// Display the graph
$graph->Stroke();
?>
```

これを完成させるには、数行を追加します。(下は変更した例 1 の一部です。完全なソースコードは上記ソースコードを参照してください。)

```
// 余白とタイトルを設定  
$graph->img->SetMargin(40,20,20,40);  
$graph->title->Set("Example 2");  
$graph->xaxis->title->Set("X-title");  
$graph->yaxis->title->Set("Y-title");
```

タイトルには以下の特徴があります。

- 文字列には、デフォルトのフォントとサイズが適用されます。
- デフォルトでグラフのタイトルは上部に配置されます。
- X 軸のタイトルのデフォルト位置は右側で、Y 軸のタイトルのデフォルト位置は中心です。また、Y 軸のタイトルは 90 度回転されます。

タイトル文字列を太字にして、ラインプロットを太く、青色に設定してみましょう。これには、以下のコードを記述します。

```
$graph->title->SetFont(FF_FONT1,FS_BOLD);  
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);  
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);  
$lineplot->SetColor("blue");  
$lineplot->SetWeight(2); // 2 ピクセルの幅
```

再度、矛盾のないインターフェースに注意してください。フォントの指定には、オブジェクトのSetFont() メソッドを使用します。この方法は、これから学ぶほとんどの関数で使用可能です。JpGraph では同様の手順でオブジェクトの外見を変更できます。たとえば、Y 軸のカラーを赤色にしたい場合は以下のようにします。

```
$graph->yaxis->SetColor("red")
```

さらに、Y 軸を太くしたい場合は以下のようにします。

```
$graph->yaxis->SetWidth(2)
```

最後に、ドロップシャドウを画像に適用します。これはデフォルトでは無効になっています。これは以下のコードで作動します。

```
$graph->SetShadow()
```

これで、以下のような画像が生成されました。

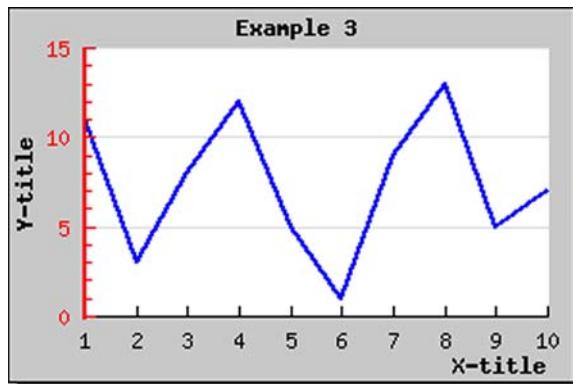


図 6: カラーを追加してフォントを変更する

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Create the linear plot
$lineplot=new LinePlot($ydata);

// Add the plot to the graph
$graph->Add($lineplot);

$graph->img->SetMargin(40,20,20,40);
$graph->title->Set("Example 3");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);
$graph->yaxis->SetColor("red");
$graph->yaxis->SetWeight(2);
$graph->SetShadow();

// Display the graph
$graph->Stroke();
?>

```

7.1.1 ライン プロット XXX にプロット記号を追加する

データ ポイントをハイライトする方法を説明します。その場合、線グラフの mark プロパティで、プロットのマーカー種類を指定します。マーカーの種類は [PlotMarks](#) クラス リファレンスを参照してください。

今回は、先ほどのグラフに三角形の形状（▲）をしたマーカーを付け加えてみましょう。

```
$lineplot->mark->SetType(MARK_UTRIANGLE);
```

これで、以下のようなグラフが表示されます。

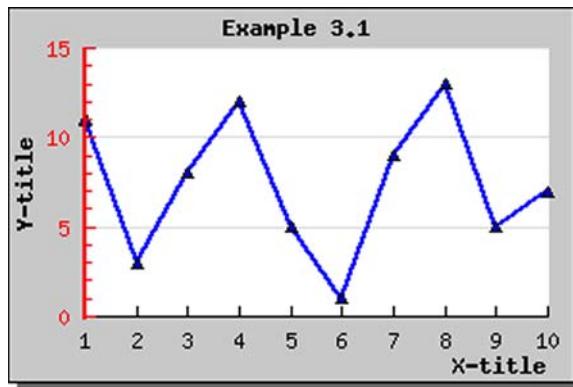


図 7:マーカーを追加する

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Create the linear plot
$lineplot=new LinePlot($ydata);

// Add the plot to the graph
$graph->Add($lineplot);

$graph->img->SetMargin(40,20,20,40);
$graph->title->Set("Example 3");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);
$graph->yaxis->SetColor("red");
$graph->yaxis->SetWeight(2);
$graph->SetShadow();

// Display the graph
$graph->Stroke();
?>

```

もちろん、マーカーのサイズやカラー、フレーム カラーなどの変更も可能です。

マーカーのカラーは明示的に指定されない限り、ライン カラーに従います。マーカーとラインに異なったカラーを指定したい場合は、ライン カラーを設定した後にマーカーの SetColor() メソッドを呼び出してください。ライン カラーを指定すると、自動的にマーカー カラーはリセットされます。

7.1.2 各データ ポイントに値を表示する

各データ ポイントの上にデータ値を表示することができます。値は、プロットの value プロパティを通じて指定できます。(詳細については、クラス リファレンスの [値を表示する](#) を参照してください。)

データ値の表示を有効にするには Show() メソッドを使用します。

```
$lineplot->value->Show()
```

このコードを先ほどのサンプルに追加すると、下記のような外見になります。

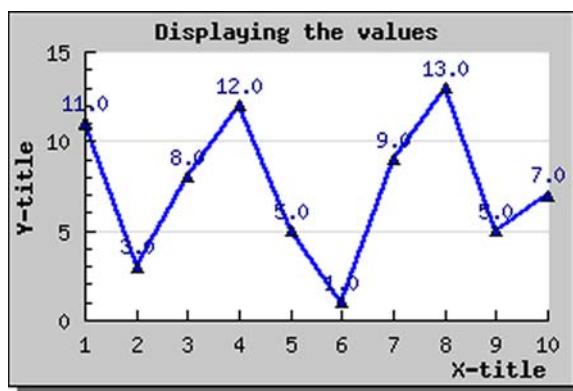


図 8:各ポイントのデータ値を表示する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Adjust the margin
$graph->img->SetMargin(40,20,20,40);
$graph->SetShadow();

// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot->mark->SetType(MARK_UTRIANGLE);
$lineplot->value->show();
```

```
// Add the plot to the graph
$graph->Add($lineplot);

$graph->title->Set("Displaying the values");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);

// Display the graph
$graph->Stroke();
?>
```

もちろん、データ値のカラー、フォント、書式を変更できます。たとえば、赤褐色の太字に設定し、先頭に \$ 記号をつける場合は以下の通りになります。

```
$lineplot->value->SetColor("darkred");
$lineplot->value->SetFont(FF_FONT1,FS_BOLD);
$lineplot->value->SetFormat("$ %0.1f");
```

この結果、下記のような外見になります。

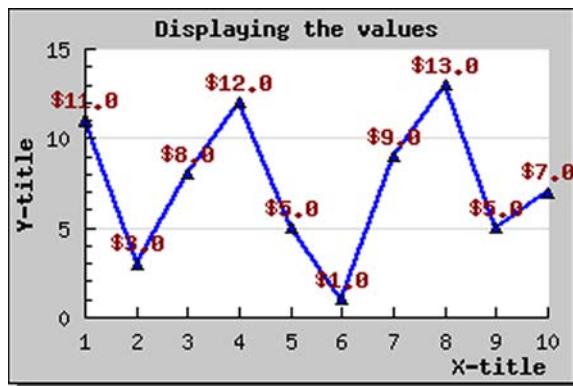


図 9:データ値に修飾を加える

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Adjust the margin
$graph->img->SetMargin(40,20,20,40);
$graph->SetShadow();

// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot->mark->SetType(MARK_UTRIANGLE);
$lineplot->value->show();
$lineplot->value->SetColor('darkred');
$lineplot->value->SetFont(FF_FONT1,FS_BOLD);

$lineplot->value->SetFormat('$%0.1f');

// Add the plot to the graph
$graph->Add($lineplot);

$graph->title->Set("Displaying the values");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);

// Display the graph
$graph->Stroke();
?>
```

注意:書式のコールバック関数を使用することで、printf() 形式の書式よりも高度な書式設定が行えます。そうすることで、数字を 3 行ずつに区切るといった表現が行えます。

7.1.3 同じグラフに複数のプロットを追加する

2 個目のプロットをグラフに追加してみたい場合どうですか？これはとても簡単で、2 つの単純な方法を必要とします。

1. 2 番目のプロットを追加する。
2. グラフに追加する。

2 番目のプロットを作成するために、新しいデータを用意します。

以下の通り、そのデータを用いて新しいプロットを作成し、グラフに追加します。

```
$ydata2 = array(1,19,15,7,22,14,5, 9,21,13);
$lineplot2=new LinePlot($ydata2);
$lineplot2->SetColor("orange");
$lineplot2->SetWeight(2);
$graph->Add($lineplot2);
```

先ほどのサンプル スクリプトに上記の変更点を追加すると、下記のようなグラフが生成されます。

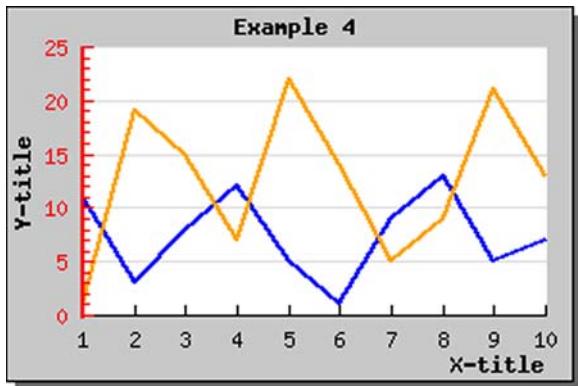


図 10:2 番目のプロットをグラフに追加する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);
$ydata2 = array(1,19,15,7,22,14,5,9,21,13);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Create the linear plot
$lineplot=new LinePlot($ydata);

$lineplot2=new LinePlot($ydata2);

// Add the plot to the graph
$graph->Add($lineplot);
$graph->Add($lineplot2);

$graph->img->SetMargin(40,20,20,40);

$graph->title->Set("Example 4");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);

$lineplot2->SetColor("orange");
$lineplot2->SetWeight(2);

$graph->yaxis->SetColor("red");
$graph->yaxis->SetWeight(2);
$graph->SetShadow();

// Display the graph
$graph->Stroke();
?>
```

以下の点を確認します。

- 2 番目のグラフに対応させるため、Y 軸の範囲は大きくなりました。
- 1 つのグラフに複数のプロットを追加する場合、同じ数のデータ ポイントを持つことが望まれます。これは必須ではありません（自動的に最大のポイント数に対応します）が、プロットの 1 つがグラフの途中で切れてしまうので、見栄えは良くありません。

7.1.4 2 個目の Y 軸を追加する

先ほどのサンプルの通り、複数のプロットを 1 つのグラフの Y 軸に追加できます。しかし、複数の範囲を持つグラフをプロットする場合はどうすれば良いでしょう。たとえば、1 つは上記のような Y の値を持っているかもしれません、ほかは 100:s で Y の値を持っている場合があります。たとえ、上記のような追加が完全に可能でも、スケールは 2 個目のプロットのより大きなダイナミックの位置を遂行しなければならないので、最小値を持つグラフは非常に低いダイナミック位置となります。

この解決策は、異なるスケールで 2 つ目の Y 軸を使い、変わりに Y 軸に 2 つ目のプロットを追加します。どのように遂行されたか見てみましょう。

2 番目の Y 軸では、異なったスケールを指定できます。たとえば、以下の通りにして大幅に値が異なった新しいデータを用意し、2 番目の Y 軸を作成します。コードは以下のようになります。

```
$y2data = array(354,200,265,99,111,91,198,225,293,251);
$graph->SetY2Scale("lin");
```

そして、ついに新しいラインプロットを作成し、それを 2 つめの Y 軸に追加します。新しい Y 軸を追加するには、AddY2() メソッドを使用します。JpGraph は最大で 2 個の Y 軸をサポートします。同じグラフに 3 つ以上のスケールを使用すると、グラフが非常に見づらくなります。そのため、JpGraph では最大で 2 つの実装になっています。

グラフをより分かりやすくするため、2 番目の Y 軸とプロットの色を指定しました。

詳細はソースコードを参照してください。

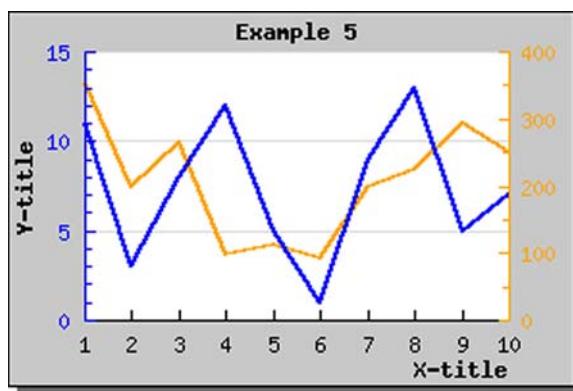


図 11:1 個のグラフに 2 番目の Y 軸を追加する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);
$y2data = array(354,200,265,99,111,91,198,225,293,251);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->img->SetMargin(40,40,20,40);
$graph->SetScale("textlin");
$graph->SetY2Scale("lin");
$graph->SetShadow();

// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot2=new LinePlot($y2data);

// Add the plot to the graph
$graph->Add($lineplot);
$graph->AddY2($lineplot2);

$lineplot2->SetColor("orange");
$lineplot2->SetWeight(2);
$graph->y2axis->SetColor("orange");

$graph->title->Set("Example 5");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);

$lineplot2->SetColor("orange");
$lineplot2->SetWeight(2);

$graph->yaxis->SetColor("blue");

// Display the graph
$graph->Stroke();
?>
```

7.1.5 グラフに凡例を追加する

1 つのグラフに複数のプロットがある場合、どのプロットが何を示しているかを説明する必要があります。そのためには凡例を追加します。

各プロットには、凡例を指定するための `SetLegend()` メソッドを有しています。凡例を追加するためには以下の行を加えます。

```
$lineplot->SetLegend("Plot 1");
$lineplot2->SetLegend("Plot 2");
```

詳細はソースコードを参照してください。

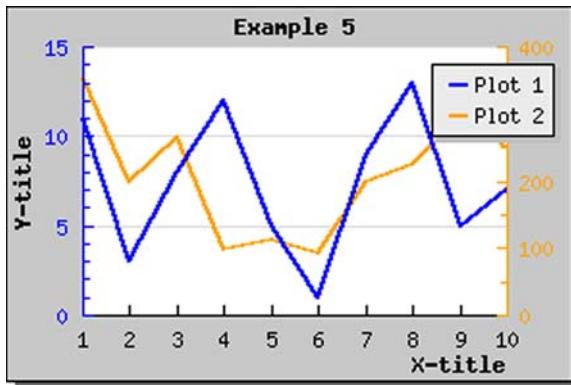


図 12:円プロットに凡例を追加する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);
$y2data = array(354,200,265,99,111,91,198,225,293,251);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->img->SetMargin(40,40,20,40);
$graph->SetScale("textlin");
$graph->SetY2Scale("lin");
$graph->SetShadow();

// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot2=new LinePlot($y2data);

// Add the plot to the graph
$graph->Add($lineplot);
$graph->AddY2($lineplot2);
$lineplot2->SetColor("orange");
$lineplot2->SetWeight(2);
```

```
$graph->y2axis->SetColor("orange");

$graph->title->Set("Example 5");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);

$lineplot2->SetColor("orange");
$lineplot2->SetWeight(2);

$graph->yaxis->SetColor("blue");

$lineplot->SetLegend("Plot 1");
$lineplot2->SetLegend("Plot 2");

// Display the graph
$graph->Stroke();
?>
```

これで、表示するための凡例テキストを持つプロットがいくつあるかによって、凡例は自動的に配置されます。デフォルトで、凡例は画像の右上に配置されます。凡例の位置の調整や、余白を追加も調節できます。両方してみましょう。

まず右側の余白を増やし、凡例がほぼ中央に置かれるよう配置します。また、プロット エリアが圧迫されないように、画像全体の大きさを広げます。

凡例を修正するには、グラフの `legend` プロパティを使用してください。凡例の詳細な機能に関しては、クラス リファレンスの [legend クラス](#) を参照してください。

たとえば、凡例の位置を指定するには SetPos() メソッドを使用します。

```
$graph->legend->Pos(0.05,0.5,"right","center");
```

これにより、以下の通りの結果が得られます。

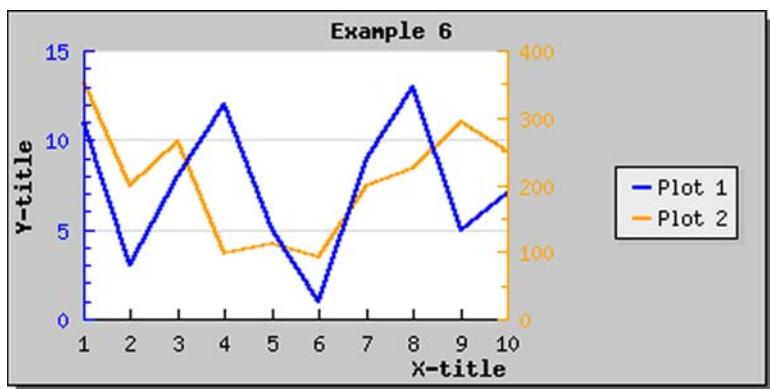


図 13: グラフに凡例を追加する

```
<?php
include ("../jgraph.php");
include ("../jgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);
$y2data = array(354,200,265,99,111,91,198,225,293,2
51);

// Create the graph and specify the scale for both Y-
axis
$graph = new Graph(400,200,"auto");
$graph->SetScale("textlin");
$graph->SetY2Scale("lin");
$graph->SetShadow();

// Adjust the margin
$graph->img->SetMargin(40,140,20,40);

// Create the two linear plot
$lineplot=new LinePlot($ydata);
$lineplot2=new LinePlot($y2data);

// Add the plot to the graph
$graph->Add($lineplot);
$graph->AddY2($lineplot2);
$lineplot2->SetColor("orange");
$lineplot2->SetWeight(2);

// Adjust the axis color
$graph->y2axis->SetColor("orange");
$graph->yaxis->SetColor("blue");

$graph->title->Set("Example 6");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Set the colors for the plots
$lineplot->SetColor("blue");
$lineplot->SetWeight(2);
$lineplot2->SetColor("orange");
$lineplot2->SetWeight(2);

// Set the legends for the plots
$lineplot->SetLegend("Plot 1");
$lineplot2->SetLegend("Plot 2");

// Adjust the legend position
$graph->legend->Pos(0.05,0.5,"right","center");

// Display the graph
$graph->Stroke();
?>
```

上記の SetPos() メソッドに説明を加えます。まず、最初の 2 個の引数には、画像の横幅と縦幅を 1 にしたときの割合を指定します。割合を指定することで、画像の大きさが変化した場合も相対的な大きさが引き継がれます。私たちは、同じメソッドがイメージに任意のテキストを置くということをその後理解するでしょう。

柔軟性を追加するために、一つは、さらに、与えられた位置が関連するどのエッジの凡例かを指定しなければなりません。今回の例では、水平位置は右 (right) 端から、垂直位置は中央 (center) であることを指定しています。

これにより、凡例は右から 5% の位置の中央に描かれることになります。もし凡例を、left と指定した場合は、画像の左側から 5% の位置に描画されます。

デフォルトでは、それぞれの凡例は縦に整列します。これ以外にも、横に整列させることができます。この指定には、[SetLayout\(\)](#) メソッドを使用します。先ほどのサンプルで横揃えの凡例を使用すると、以下の通りになります。

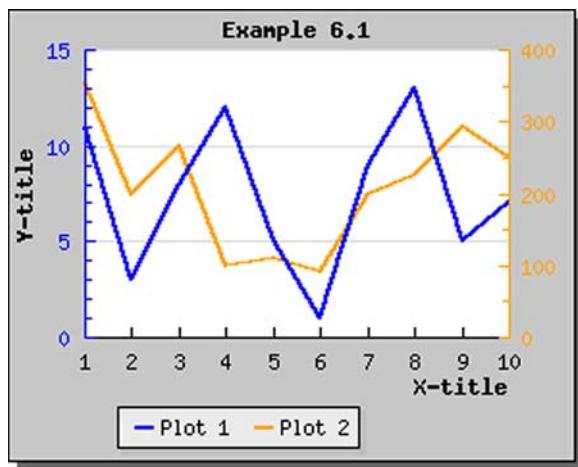


図 14: 凡例を水平に設置する

```
<?php
include ("../jgraph.php");
include ("../jgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);
$y2data = array(354,200,265,99,111,91,198,225,293,2
51);

// Create the graph and specify the scale for both Y
axis
$graph = new Graph(300,240,"auto");
$graph->SetScale("textlin");
$graph->SetY2Scale("lin");
$graph->SetShadow();

// Adjust the margin
$graph->img->SetMargin(40,40,20,70);

// Create the two linear plot
$lineplot=new LinePlot($ydata);
$lineplot2=new LinePlot($y2data);

// Add the plot to the graph
$graph->Add($lineplot);
$graph->AddY2($lineplot2);
$lineplot2->SetColor("orange");
$lineplot2->SetWeight(2);

// Adjust the axis color
$graph->y2axis->SetColor("orange");
$graph->yaxis->SetColor("blue");

$graph->title->Set("Example 6.1");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Set the colors for the plots
$lineplot->SetColor("blue");
$lineplot->SetWeight(2);
$lineplot2->SetColor("orange");
$lineplot2->SetWeight(2);

// Set the legends for the plots
$lineplot->SetLegend("Plot 1");
$lineplot2->SetLegend("Plot 2");

// Adjust the legend position
$graph->legend->SetLayout(LEGEND_HOR);
$graph->legend->Pos(0.4,0.95,"center","bottom");

// Display the graph
$graph->Stroke();
?>
```

7.1.5.1 凡例のレイアウトを調節する

もっと高度な凡例のフォーマットのため、調節することができます。

- 凡例のコラム
- 凡例のコラムに縦/横の余白を指定する

凡例で使用されるコラムの調節で、メソッド `Legend::SetColumns` が使用されます。たとえば、3 つのコラムの使用で調整される凡例を持つため、以下のラインをスクリプトに追加する必要があります。

```
$graph->legend->SetColumns(3);
```

7.1.6 ライン プロットで空白値を使用する

JpGraph では、データに空白値（不連続）を与える 2 通りの方法があります。データの“すべて”またはグラフの前後のデータ ポイントの間で拡張されるラインのどちらかを持つことができます。

もしデータが空白値（“”）、あるいは特別な値（“x”）である場合、データ ポイントはプロットされず、不連続な線分が描画されます。

もしデータの値が “–” だった場合、そのポイントを無視して線分が描画されます。

下記にこれらの例を掲載します。

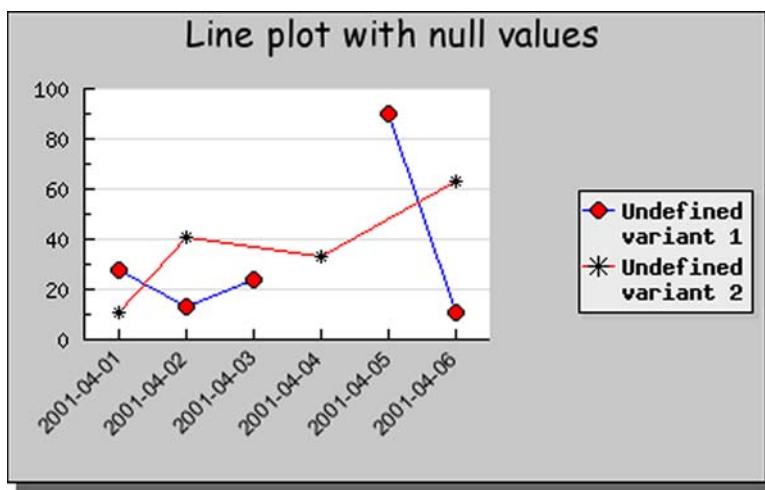


図 15: 線グラフで空白値を扱う

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

// Some data
$datax = array("2001-04-01","2001-04-02","2001-
04-03","2001-04-04","2001-04-05","2001-04-
06");
$datay = array(28,13,24,"",90,11);
$data2y = array(11,41,"-",33,"-",63);

// Setup graph
$graph = new Graph(400,250,"auto");
$graph->img->SetMargin(40,150,40,80);
$graph->SetScale("textlin");
$graph->SetShadow();

//Setup title
$graph->title->Set("Line plot with null values");

// Use built in font
$graph->title->SetFont(FF_COMIC,FS_NORMAL,14);

// Slightly adjust the legend from it's default position
$graph->legend->Pos(0.03,0.5,"right","center");
$graph->legend->SetFont(FF_FONT1,FS_BOLD);

// Setup X-scale
$graph->xaxis->SetTickLabels($datax);
$graph->xaxis->SetFont(FF_ARIAL,FS_NORMAL,8);
$graph->xaxis->SetLabelAngle(45);

// Create the first line
$p1 = new LinePlot($datay);
$p1->mark->SetType(MARK_FILLEDIRCLE);
$p1->mark->SetFillColor("red");
$p1->mark->SetWidth(4);
$p1->SetColor("blue");
$p1->SetCenter();
$p1->SetLegend("Undefined variant 1");
$graph->Add($p1);

// ... and the second
$p2 = new LinePlot($data2y);
$p2->mark->SetType(MARK_STAR);
$p2->mark->SetFillColor("red");
$p2->mark->SetWidth(4);
$p2->SetColor("red");
$p2->SetCenter();
$p2->SetLegend("Undefined variant 2");
$graph->Add($p2);

// Output line
$graph->Stroke();

?>
```

7.1.7 ステップ形式のラインプロットを描画する

ステップ形式のラインプロットでは、それぞれの値を直線的に結ぶのではなく、次の X 値までを描画する水平線と Y 値までを描画する垂直線の 2 線を用いて描画します。この場合、まず水平線を次の X 値まで引き、次にそのポイントから垂直線を Y 値まで引きます。以下の図を参照してください。

この形式のグラフを描画するには、[SetStepStyle\(\)](#) メソッドを呼び出します。

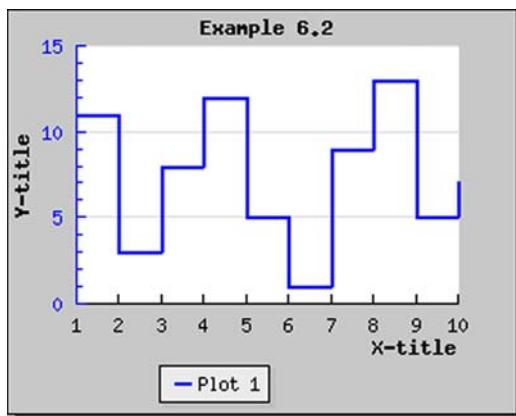


図 16:ステップ形式でラインプロットを描画する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);
$y2data = array(354,200,265,99,111,91,198,225,293,2
51);

// Create the graph and specify the scale for both Y-
axis
$graph = new Graph(300,240,"auto");
$graph->SetScale("textlin");
$graph->SetShadow();

// Adjust the margin
$graph->img->SetMargin(40,40,20,70);

// Create the two linear plot
$lineplot=new LinePlot($ydata);
$lineplot->SetStepStyle();

// Adjust the axis color
$graph->yaxis->SetColor("blue");

$graph->title->Set("Example 6.2");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Set the colors for the plots
$lineplot->SetColor("blue");
$lineplot->SetWeight(2);

// Set the legends for the plots
$lineplot->SetLegend("Plot 1");

// Add the plot to the graph
$graph->Add($lineplot);

// Adjust the legend position
$graph->legend->SetLayout(LEGEND_HOR);
$graph->legend->Pos(0.4,0.95,"center","bottom");

// Display the graph
$graph->Stroke();
?>
```

7.1.8 対数グラフを使用する

各軸に対数スケールを適用するには、対数モジュールである `jpgraph_log.php` を別途インクルードする必要があります。そのためには、プログラムコードの最上部で、以下のように記述します。

```
include("../jpgraph_log.php");
```

対数グラフの使用法を紹介するために、右側の Y 軸に対数軸を用いた片対数グラフを作成します。

```
$graph->SetY2Scale("log");
```

これにより、以下の結果が得られます。

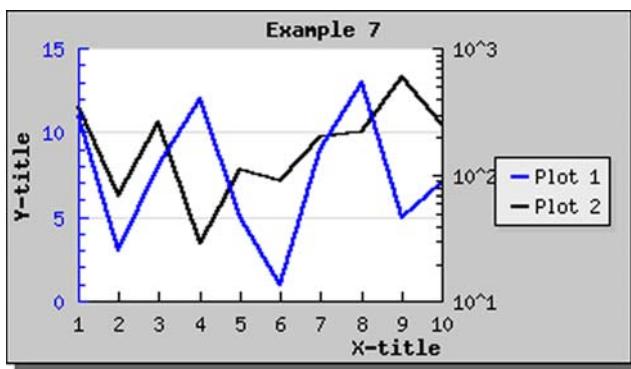


図 17:Y2 軸に対数スケールを使用する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_log.php");
include ("../jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);
$y2data = array(354,70,265,29,111,91,198,225,593,25
1);

// Create the graph.
$graph = new Graph(350,200,"auto");
$graph->SetScale("textlin");
$graph->SetY2Scale("log");
$graph->SetShadow();
$graph->img->SetMargin(40,110,20,40);

// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot2=new LinePlot($y2data);

// Add the plot to the graph
$graph->Add($lineplot);
$graph->AddY2($lineplot2);

$graph->yaxis->SetColor('blue');
$graph->title->Set("Example 7");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);
$lineplot2->SetWeight(2);

$lineplot->SetLegend("Plot 1");
$lineplot2->SetLegend("Plot 2");

$graph->legend->Pos(0.05,0.5,"right","center");

// Display the graph
$graph->Stroke();
?>

```

同様に、X 軸に対数軸を使用することもできます。以下に例を紹介します。

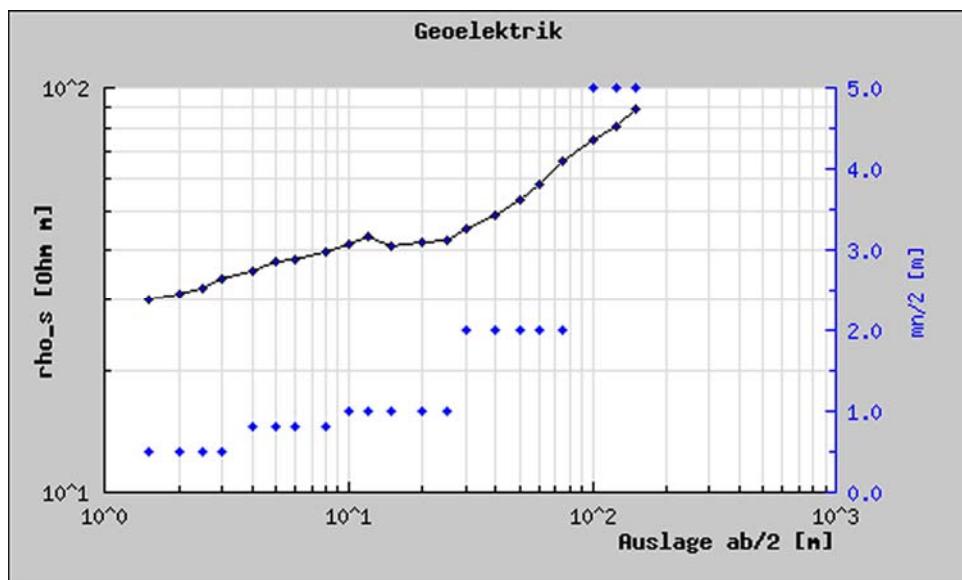


図 18:X 軸と Y 軸に対数スケールを用いる例

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_log.php");
include ("../jpgraph_line.php");

```

直角の XY プロットを使用する

```
include ("./jpgraph_scatter.php");

$ab2 = array( 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 6.0,
8.0, 10.0,
12.0, 15.0, 20.0, 25.0, 30.0, 40.0, 50.0,
60.0 ,75.0,
100., 125., 150.);
$mn2 = array( 0.5, 0.5, 0.5, 0.5, 0.8, 0.8, 0.8,
0.8, 1.0,
1.0, 1.0, 1.0, 1.0, 2.0, 2.0, 2.0,
2.0, 2.0,
5.0, 5.0, 5.0);
$rhos = array(30.0, 31.0, 32.0, 34.0, 35.5, 37.5, 38.0,
39.5, 41.5,
43.0, 41.0, 42.0, 42.5, 45.0, 49.0, 53.5,
58.0, 66.5,
75.0, 81.0, 89.0);

// Create the graph.
$graph = new Graph(500,300,"auto");
$graph->SetScale("loglog");
$graph->SetY2Scale("lin");
$graph->y2axis->SetColor("blue","blue");

$graph->img->SetMargin(50,70,40,50);
$graph->title->Set("Geoelektrik");
$graph->xaxis->title->Set("Auslage ab/2 [m]");
$graph->yaxis->title->Set("rho_s [Ohm m]");
$graph->y2axis->title->Set("mn/2 [m]");
$graph->y2axis->title->SetColor("blue");

$graph->y2axis->SetTitleMargin(35);
$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xgrid->Show(true,true);
$graph->ygrid->Show(true,true);

// Create the linear plot
$lineplot=new LinePlot($rhos,$ab2);
$lineplot->SetWeight(1);
$lineplot->mark->SetType(MARK_FILLEDIRCLE);
$lineplot->mark->SetWidth(2);

// Create scatter plot
$scplot=new ScatterPlot($mn2,$ab2);
$scplot->mark->SetType(MARK_FILLEDIRCLE);
$scplot->mark->SetColor("blue");
$scplot->mark->SetWidth(2);

// Add plots to the graph
$graph->AddY2($scplot);
$graph->Add($lineplot);

// Display the graph
$graph->Stroke();
?>
```

対数スケールはバー プロット、エラー プロット、散布プロットなどでも同様に使用できます。また、ライン プロットだけでなく、バー プロットの場合も対数軸を使用できます。以下にその例を紹介します。

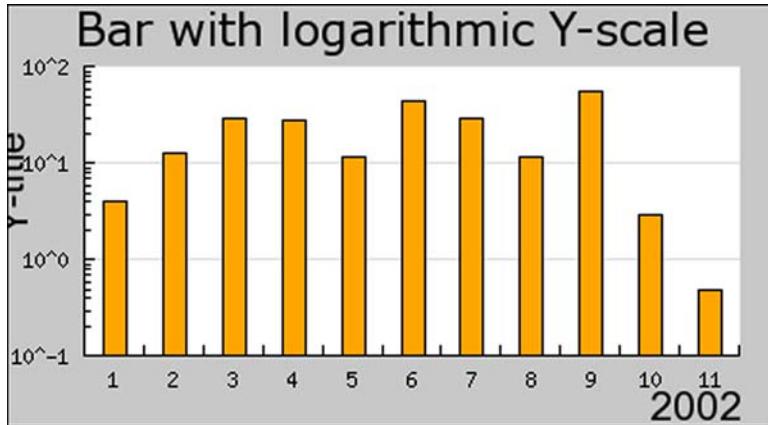


図 19:棒グラフに対数軸を使用する

```
<?php
// $Id: logbarex1.php,v 1.4 2003/05/30 20:12:43 aditus Exp $
include ("./jpgraph.php");
include ("./jpgraph_log.php");
include ("./jpgraph_bar.php");

$datay=array(4.13,30,28,12,45,30,12,55,3,0.5);
$datax=array("Jan","Feb","Mar","Apr","May","Jun","Jul",
"Aug","Sep","Oct","Nov");

// Create the graph.
$graph = new Graph(400,220,'auto');
// $graph->img->SetMargin(50,30,50,50);

$graph->SetScale("textlog");
// $graph->SetShadow();

// Setup titles for graph and axis
$graph->title->Set("Bar with logarithmic Y-scale");
$graph->title->SetFont(FF_VARDANA,FS_NORMAL,18);

$graph->xaxis->SetTitle("2002");
$graph->xaxis->title-
>SetFont(FF_ARIAL,FS_NORMAL,16);

$graph->yaxis->title-
>SetFont(FF_ARIAL,FS_NORMAL,16);
$graph->yaxis->SetTitle("Y-title",center);
```

```
$graph->yaxis->SetTitleMargin(30);

// Setup month on X-scale
// $graph->xaxis->SetTickLabels($datax);

// Create a bar plot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("orange");

// You can also set a manual base of the bars
// $bplot->SetYBase(0.001);

/*
$bplot->SetShadow();
$bplot->value->Show();
$bplot->value->SetFont(FF_ARIAL,FS_BOLD);
$bplot->value->SetAngle(45);
$bplot->value->SetColor("black","darkred");
*/

$graph->Add($bplot);

$graph->Stroke();
?>
```

7.1.9 スケールの詳細

先ほどの例の通り、異なった種類のスケールを使用できます。JpGraph では、以下のスケールをサポートしています。

- リニア スケール。標準のスケールです。
- 対数スケール。
- 整数スケール。リニア スケールと似ていますが、スケール値（およびラベル）は整数に制限されます。
- テキスト スケール。整数スケールに似ていますが、テキスト スケールは X 軸で使用される場合がほとんどです。テキスト スケールはもっぱら X 軸で使われます。よく使われる例として、バー プロットの X 軸があります。テキスト スケールを組み合わせることもできます。リニア スケールと対数スケールの組み合わせは一般的です。テキスト スケールは X 軸でのみ使用できます。Y 軸に設定しても意味がありません。

これらの組み合わせを使用します。リニア スケールと対数スケールは、とても理解しやすいものです。テキスト スケールにはもう少し説明が必要です。テキスト スケールは、0、1、2、3、4 のように自然数を表すリニア スケールと同じようなものだと考えてください。このスケールは、連続でプロットしたい Y 値を持っていて、X 値の値が不要な場合のみに使用されます。上記のような場合は、X 軸にリニア スケールを使用することもできます。しかし、現在リニア スケールは実数として扱われるため、オートスケール機能により自然数ではない値をラベルに表示する可能性があります。例、ラベルが 2.5 の場合があります。テキスト スケールを使用する場合はこのようなことは起きません。

テキスト スケールでは初期設定の自然数の代わりに文字列を指定するのが慣例となっています。ラベルに文字列を指定するには、軸のプロパティに [SetTickLabels\(\)](#) メソッドを使用してください。

スケールを指定するには、[SetScale\(\)](#) メソッドを使用します。下記に使用例を掲載します。

- `textlin` と指定すると、X 軸にはテキスト スケールが、Y 軸にはリニア スケールが適用されます。
- `linlin` と指定すると、X 軸、Y 軸にリニア スケールが適用されます。
- `linlog` と指定すると、X 軸にはリニア スケールが、Y 軸には対数スケールが適用されます。

- loglog と指定すると、X 軸、Y 軸に対数スケールが適用されます。
- textint と指定すると、X 軸にテキストスケールが、Y 軸に整数スケールが適用されます。
- textlog と指定すると、X 軸にはテキストスケールが、Y 軸には対数スケールが適用されます。

一度は必ず SetScale() メソッドを呼び出す必要があります。通常は、Graph() オブジェクトを作成した直後になります。

Y2 軸のスケールを指定するには、[SetY2Scale\(\)](#) メソッドを使用します。対数の Y2 軸のスケールを設定するには

```
$graph->SetY2Scale("log");
```

を呼び出します。

7.1.10 プロットでグリッド(目盛り)ラインを調整する

デフォルトでは、Y 軸にのみグリッド線が適用されています。ラベル付きの目盛り(Tick)に設定するなど、この変更は可能です。グリッド線は、X、Y、Y2 軸のいずれの軸にも使用できます。また、グリッドに小さなチェック・マークをつけることも可能です。例えば、ラベルなしの Tick にすることもできます。上記のグラフにこれらを適用する方法を見てみましょう。

グリッドは、グラフの xgrid あるいは ygrid コンポーネントを通じて指定できます。Y グラフの補助グリッド線を表示する場合、以下の通りに記述します。

```
$graph->ygrid->Show(true,true)
```

1 番目の引数で、グリッドを表示するかどうかを指定します。2 番目の引数で、補助グリッド線を表示するかどうかを指定します。

Y2 軸にもグリッド線を表示する場合は、以下の通りに記述します。

```
$graph->y2grid->Show(true,true)
```

注意:結果のイメージがビューウィーでは読みづらくなるので、一般的には、Y 軸と Y2 軸の両方にグリッドを表示するのは良いアイデアではありません。

また、X 軸のグリッドを表示するには以下の通りに記述します。

```
$graph->xgrid->Show(true)
```

上記の場合は、補助グリッド線を表示しません。

これらをすべて使用することで、X 軸と Y 軸のグリッド線を表示できます。

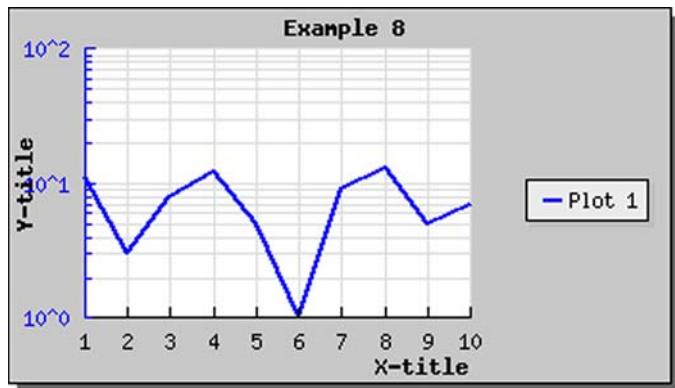


図 20:X 軸のグリッド線、Y 軸のグリッド線と補助グリッド線を表示する

```

<?php
include ("../jgraph.php");
include ("../jgraph_log.php");
include ("../jgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);
$y2data = array(354,200,265,99,111,91,198,225,293,2
51);

// Create the graph. These two calls are always required
$graph = new Graph(350,200,"auto");
$graph->SetScale("textlog");
$graph->SetShadow();
$graph->img->SetMargin(40,110,20,40);

// Show the gridlines
$graph->ygrid->Show(true,true);
$graph->xgrid->Show(true,false);

// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot2=new LinePlot($y2data);

// Add the plot to the graph
$graph->Add($lineplot);

$graph->title->Set("Example 8");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);

// Adjust the color of the Y axis
$graph->yaxis->SetColor("blue");

// Specify a legend
$lineplot->SetLegend("Plot 1");

// Adjust the position of the grid box
$graph->legend->Pos(0.05,0.5,"right","center");

// Display the graph
$graph->Stroke();
?>

```

注意: Y 軸の最初の値が X 軸に近すぎると感じたら、[SetLabelMargin\(\)](#) メソッドを使用して余白を増やすか、[HideFirstTickLabel\(\)](#) メソッドを使用して最初のラベルを非表示にしてください。

7.1.11 塗りつぶされたグリッド線を使用する

グリッド線の間を別々の 2 色のカラーで塗りつぶすことが可能です。以下にその例を掲載します。

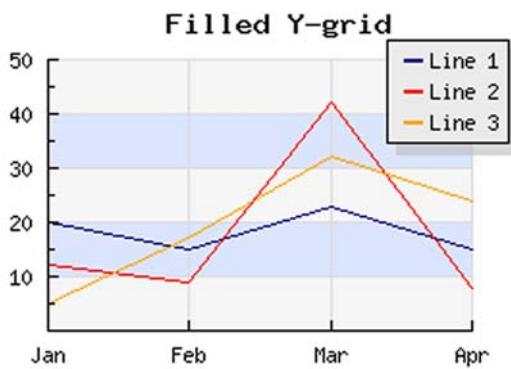


図 21: グリッド線の間に 2 種類のカラーを使用する

```

<?php
include ("../jgraph.php");
include ("../jgraph_line.php");

$datay1 = array(20,15,23,15);
$datay2 = array(12,9,42,8);
$datay3 = array(5,17,32,24);

// Setup the graph
$graph = new Graph(300,200);
$graph->SetMarginColor('white');
$graph->SetScale("textlin");
$graph->SetFrame(false);
$graph->SetMargin(30,50,30,30);

$graph->title->Set('Filled Y-grid');

$graph->yaxis->HideZeroLabel();
$graph->ygrid-
>SetFill(true,'#EFEFEF@0.5','#BBCCFF@0.5');
$graph->xgrid->Show();

$graph->xaxis->SetTickLabels($gDateLocale-
>GetShortMonth());
    
```

```

// Create the first line
$p1 = new LinePlot($datay1);
$p1->SetColor("navy");
$p1->SetLegend('Line 1');
$graph->Add($p1);

// Create the second line
$p2 = new LinePlot($datay2);
$p2->SetColor("red");
$p2->SetLegend('Line 2');
$graph->Add($p2);

// Create the third line
$p3 = new LinePlot($datay3);
$p3->SetColor("orange");
$p3->SetLegend('Line 3');
$graph->Add($p3);

$graph->legend->SetShadow('gray@0.4',5);
$graph->legend->SetPos(0.1,0.1,'right','top');
// Output line
$graph->Stroke();
?>
    
```

上記の例の場合、GD2.x 以上の場合には透明色の使用が可能です。標準設定では塗りつぶしのグリッド線は無効になっています。使用するには [Grid::SetFill\(\)](#) メソッドを使用します。

7.1.12 X 軸にテキストラベルを指定する

テキストスケールが指定されている場合、X 軸に特定のラベルを表示できます。上記の例では、Y 軸の値がそれぞれ最初の 10 ヶ月間を表しています。X スケールに各月の名前を表示したい場合は、以下のように設定します。

スケールにラベルを指定する場合、[SetTickLabels\(\)](#) メソッドを使用します。

JpGraph では、[DateLocale](#) インスタンスの \$gDateLocal オブジェクトを使用すると、月の名前が取得できます。

このクラスでは、さまざまな日付(月や週)の名前を取り出せます。

X 軸に短い形式の月名を指定したい場合、以下の通りに記述します。

```

$a = $gDateLocale->GetShortMonth();
$graph->xaxis->SetTickLabels($a);
    
```

これで、以下の通りの結果が得られます。

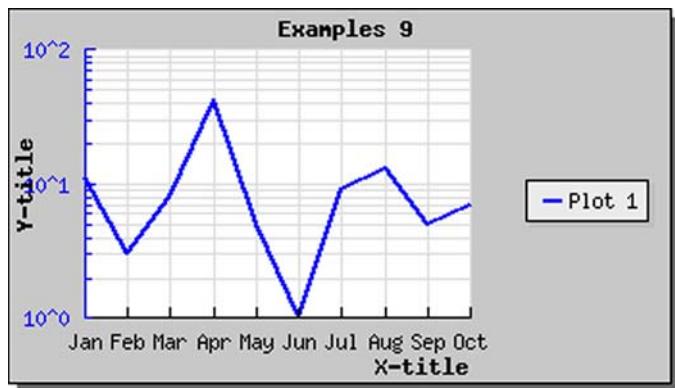


図 22:X 軸にテキストラベルを指定する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_log.php");
include ("./jpgraph_line.php");

$ydata = array(11,3,8,42,5,1,9,13,5,7);
$xdata = array("Jan","Feb","Mar","Apr","May","Jun","Ju
l","aug","Sep","Oct");

// Create the graph. These two calls are always requir
ed
$graph = new Graph(350,200,"auto");
$graph->SetScale("textlog");

$graph->img->SetMargin(40,110,20,40);
$graph->SetShadow();

$graph->ygrid->Show(true,true);
$graph->xgrid->Show(true,false);

// Specify the tick labels
$a = $gDateLocale->GetShortMonth();
$graph->xaxis->SetTickLabels($a);

// Create the linear plot
$lineplot=new LinePlot($ydata);

// Add the plot to the graph
$graph->Add($lineplot);

$graph->title->Set("Examples 9");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);

$graph->yaxis->SetColor("blue");

$lineplot->SetLegend("Plot 1");

$graph->legend->Pos(0.05,0.5,"right","center");

// Display the graph
$graph->Stroke();
?>
```

注意:同様に、Y 軸と Y2 軸のラベルを上書きできます。しかし、指定するラベルは系列の各数値にのみ適用されるので、系列に無い値にラベルを指定できません。十分な数のラベルが与えられないと、空のラベルが表示されます。

7.1.13 テキストスケールのチックを調整する

先ほどの例では、X 軸のラベル間の余白があまりありませんでした。イメージを拡大させるか、x-軸のより少ないチックのラベルを単に表示することにより、これを改正する場合があります。

たとえば、1 個置きにラベルを表示するには、[SetTextLabelInterval\(\)](#) メソッドを使用します。これにより、以下のグラフが描画されます。

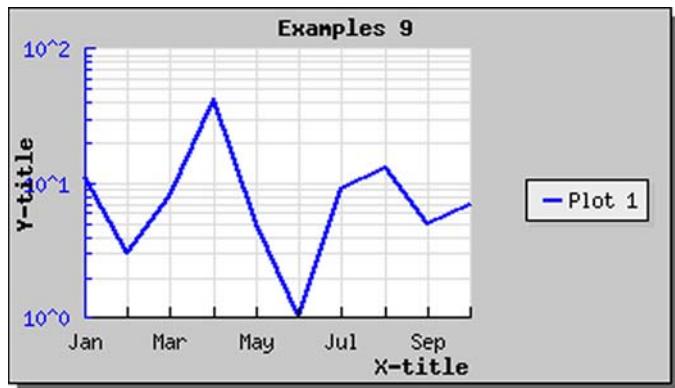


図 23:X 軸のラベルを 1 個置きに表示する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_log.php");
include ("../jpgraph_line.php");

$ydata = array(11,3,8,42,5,1,9,13,5,7);
$xdata = array("Jan","Feb","Mar","Apr","Maj","Jun","Ju
l","aug","Sep","Oct");

// Create the graph. These two calls are always required
$graph = new Graph(350,200,"auto");
$graph->SetScale("textlog");

$graph->img->SetMargin(40,110,20,40);
$graph->SetShadow();

$graph->ygrid->Show(true,true);
$graph->xgrid->Show(true,false);

// Specify the tick labels
$a = $gDateLocale->GetShortMonth();
$graph->xaxis->SetTickLabels($a);
$graph->xaxis->SetTextLabelInterval(2);

// Create the linear plot
$lineplot=new LinePlot($ydata);

// Add the plot to the graph
$graph->Add($lineplot);

$graph->title->Set("Examples 9");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);

$graph->yaxis->SetColor("blue");
$lineplot->SetLegend("Plot 1");

$graph->legend->Pos(0.05,0.5,"right","center");

// Display the graph
$graph->Stroke();
?>

```

テキスト ラベルが長い場合、テキストを回転することができます。今回は、X 軸のラベルを 90 度回転してみましょう。それには [SetLabelAngle\(\)](#) メソッドを使用します。

これにより、以下のグラフが得られます。

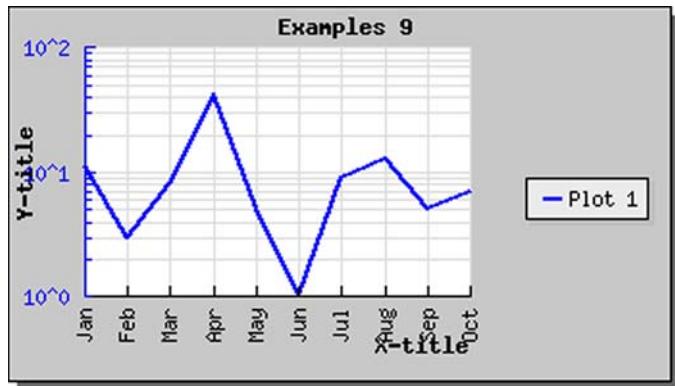


図 24:X 軸のラベルを 90 度回転します

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_log.php");
include ("./jpgraph_line.php");

$ydata = array(11,3,8,42,5,1,9,13,5,7);
$xdata = array("Jan","Feb","Mar","Apr","May","Jun","Ju
l","Aug","Sep","Oct");

// Create the graph. These two calls are always requir
ed
$graph = new Graph(350,200,"auto");
$graph->SetScale("textlog");

$graph->img->SetMargin(40,110,20,50);
$graph->SetShadow();

$graph->ygrid->Show(true,true);
$graph->xgrid->Show(true,false);

// Specify the tick labels
$a = $gDateLocale->GetShortMonth();
$graph->xaxis->SetTickLabels($a);
// $graph->xaxis->SetTextLabelInterval(2);
$graph->xaxis->SetLabelAngle(90);

// Create the linear plot
$lineplot=new LinePlot($ydata);

// Add the plot to the graph
$graph->Add($lineplot);

$graph->title->Set("Examples 9");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);

$graph->yaxis->SetColor("blue");

$lineplot->SetLegend("Plot 1");

$graph->legend->Pos(0.05,0.5,"right","center");

// Display the graph
$graph->Stroke();
?>

```

注意: 内部フォントは 0 度、あるいは 90 度しか回転できません。TrueType フォントの場合は任意の角度で回転できます。フォントに関する詳しい説明は後に表記します。

7.1.14 塗りつぶされたライン プロットを使用する

塗りつぶされたライン プロットは、通常のライン プロットとあまり変わりません。通常のライン プロットとの違いは、[SetFillColor\(\)](#)を呼び出す必要がある事です。そうすることで、指定したカラーで線グラフの下側を塗りつぶします。

以下の例ではプロットマークも指定しました(前のセクションを参照してください)。

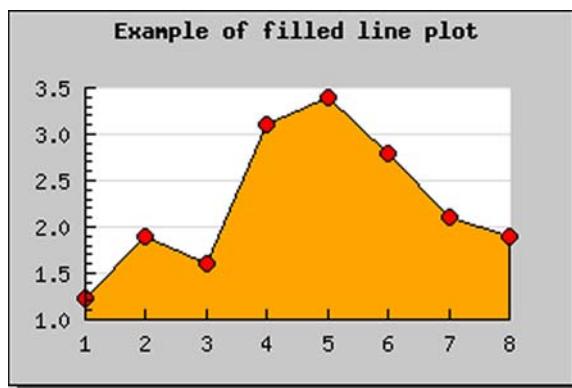


図 25: 塗りつぶされた線グラフを表示する

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$dataay = array(1.23,1.9,1.6,3.1,3.4,2.8,2.1,1.9);
$graph = new Graph(300,200,"auto");

$graph->SetScale("textlin");

$graph->img->SetMargin(40,40,40,40);
$graph->SetShadow();

$graph->title->Set("Example of filled line plot");

```

```
$graph->title->SetFont(FF_FONT1,FS_BOLD);
$p1 = new LinePlot($datay);
$p1->SetFillColor("orange");
$p1->mark->SetType(MARK_FILLEDCIRCLE);
$p1->mark->SetFillColor("red");

$p1->mark->SetWidth(4);
$graph->Add($p1);

$graph->Stroke();
?>
```

注意 1.後に追加されるプロットは、上書きして表示されます。そのため、複数の塗りつぶされたラインプロットを1つのグラフに使用する場合、Y値が高い方のプロットを先に追加する必要があります。プロットはグラフに追加された順番で表示されるので、最前列に表示したいプロットは最後に追加してください。

注意 2.塗りつぶされたラインプロットで凡例を使用する場合、凡例は上塗り色で表示されます。

注意 3.塗りつぶされたラインプロットは、正の値の場合のみサポートされます。負の値の場合は期待する動作を行いません。

上記の例では、グリッド線はグラフの下に描画されています。もしグリッド線をグラフの上部に設置したい場合は、[Graph::SetGridDepth\(\)](#)メソッドを使用して配置を指定できます。

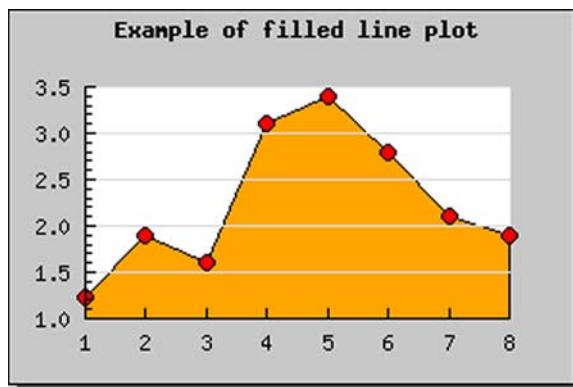


図 26:グリッド線の奥行きを指定する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$datay = array(1.23,1.9,1.6,3.1,3.4,2.8,2.1,1.9);
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

$graph->img->SetMargin(40,40,40,40);
$graph->SetShadow();
$graph->SetGridDepth(DEPTH_FRONT);

$graph->title->Set("Example of filled line plot");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

$p1 = new LinePlot($datay);
$p1->SetFillColor("orange");
$p1->mark->SetType(MARK_FILLEDCIRCLE);
$p1->mark->SetFillColor("red");
$p1->mark->SetWidth(4);
$graph->Add($p1);

$graph->Stroke();
?>
```

7.1.15 積み上げラインプロットを使用する

積み上げグラフは、それぞれの線グラフが積み重なったグラフです。要するに、Y軸の値は絶対値ではなく、すべてのグラフを積み重ねた値です。たとえば、[3, 7, 5] と [6, 9, 7] という値を持つ2

種類のグラフがある場合。1 番目のグラフの Y 軸は [3, 7, 5] になり、2 番目のグラフの Y 値は [3+6, 7+9, 5+7] になります。

線グラフは任意の数だけ積算できます。3 個の線グラフを使用して積み上げ線グラフを作成したい場合、以下のコードを記述します。

```
// 最初、個別にプロットを作成
$p1 = new LinePlot($datay_1);
$p2 = new LinePlot($datay_2);
$p3 = new LinePlot($datay_3);

// 次に、蓄積されてプロットを形成するためそれらと一緒に加える
$ap = new AccLinePlot(array($p1,$p2,$p3));

// 蓄積されたライン プロットをグラフに追加
$graph->Add($ap);
```

また、それぞれのライン プロットを塗り重ねることができます。

```
$p1->SetFillColor("red");
$p2->SetFillColor("blue");
$p3->SetFillColor("green");
```

これにより、以下のようなグラフが作成されます。

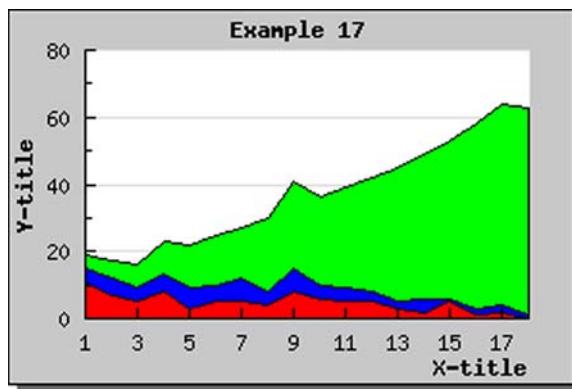


図 27: 積み上げ線グラフを作成する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

$datay1 = array(11,7,5,8,3,5,5,4,8,6,5,5,3,2,5,1,
2,0);
$datay2 = array( 4,5,4,5,6,5,7,4,7,4,4,3,2,4,1,2,2,1);
$datay3 = array(4,5,7,10,13,15,15,22,26,26,26,30,
34,40,43,47,55,60,62);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");
$graph->SetShadow();
$graph->img->SetMargin(40,30,20,40);

// Create the linear plots for each category
$dplot[] = new LinePlot($datay1);
$dplot[] = new LinePlot($datay2);
$dplot[] = new LinePlot($datay3);
```

```
$dplot[0]->SetFillColor("red");
$dplot[1]->SetFillColor("blue");
$dplot[2]->SetFillColor("green");

// Create the accumulated graph
$accplot = new AccLinePlot($dplot);

// Add the plot to the graph
$graph->Add($accplot);

$graph->xaxis->SetTextTickInterval(2);
$graph->title->Set("Example 17");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>
```

7.1.16 三次スプラインで滑らかなライン プロットを作成する

データ ポイントの数が少ないものの、滑らかなカーブでデータ ポイントを接続したい場合、JpGraph では三次スプライン補完を使用してグラフの描画を行えます。三次スプラインを聞いたことがなくとも心配しないでください。知っているデータ ポイントを与える必要があり、JpGraph にグラフに欲しい挿入されたポイントが合計でいくつかということを伝えます。この機能を使うと、JpGraph はすべてのデータ ポイントの間を滑らかな曲線で自動的に補完します。新しい曲線は、与えられた補完ポイントの数だけ正確に生成されます。

スプライン補完を行う場合、データ ポイントの X と Y 座標を指定する必要があります。

まずは [Spline](#) インスタンスを作成します。Spline クラスを使用する場合、jpgraph_regstat.php ファイルをインクルードする必要があります。インスタンスの作成時には、2 種類のデータ配列（X 座標と Y 座標）を指定します。

```
$spline = new Spline($xdata,$ydata);
```

これにより、データ ポイント間に三次スプライン補完を行います。また、これらのデータ ポイントは、スプラインの コントロール ポイント としても機能します。Spline クラスでは、実際の線の描画を行いません。その代わりに、補完値を含む新しいデータ配列を作成します。その後、このデータ配列をプロットに使用してください。この方法は補完データを使用する場合、高い柔軟性を与えてくれます。

[Spline::Get\(\)](#) メソッドにより、ポイントが補完されたデータ配列を取得できます。以下に、その例を掲載します。

```
list($sdatax,$sdatay) = $spline->Get(50);
```

この例では、\$sdatax および \$sdatay 配列を作成しています。それぞれ 50 データ ポイント分作成されています。これらの配列は、\$spline オブジェクトを作成する際に指定したコントロール ポイントから作成されます。

次にこれらのデータ配列を使用して、グラフを描画します。

以下に、その例を掲載します。

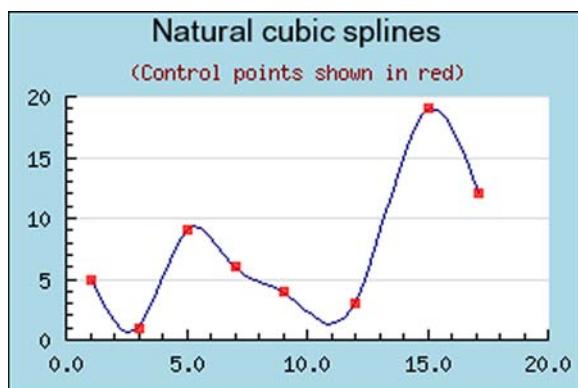


図 28:三次元スプライン補完でコントロール ポイントを接続する

```

<?php
include "../jpgraph.php";
include "../jpgraph_line.php";
include "../jpgraph_scatter.php";
include "../jpgraph_regstat.php";

// Original data points
$xdata = array(1,3,5,7,9,12,15,17,1);
$ydata = array(5,1,9,6,4,3,19,12);

// Get the interpolated values by creating
// a new Spline object.
$spine = new Spline($xdata,$ydata);

// For the new data set we want 40 points to
// get a smooth curve.
list($newx,$newy) = $spine->Get(50);

// Create the graph
$g = new Graph(300,200);
$g->SetMargin(30,20,40,30);
$g->title->Set("Natural cubic splines");
$g->title->SetFont(FF_ARIAL,FS_NORMAL,12);
$g->subtitle->Set('Control points shown in red');
$g->subtitle->SetColor('darkred');
$g->SetMarginColor('lightblue');

// $g->img->SetAntiAliasing();

```

```

// We need a linlin scale since we provide both
// x and y coordinates for the data points.
$g->SetScale('linlin');

// We want 1 decimal for the X-label
$g->xaxis->SetLabelFormat('%1.1f');

// We use a scatterplot to illustrate the original
// control points.
$splot = new ScatterPlot($ydata,$xdata);

//
// $splot->mark->SetFillColor('red@0.3');
// $splot->mark->SetColor('red@0.5');

// And a line plot to stroke the smooth curve we got
// from the original control points
$plot = new LinePlot($newy,$newx);
$plot->SetColor('navy');

// Add the plots to the graph and stroke
$g->Add($plot);
$g->Add($splot);
$g->Stroke();

?>

```

より良い例を作成するために、実際は 2 つのプロットを使用しました。一つは滑らかな曲線を得るためにライン プロットで、もう一つはコントロール ポイントがある場所を示すための散布プロットです。

7.1.17 プロット記号をライン プロットに追加する

特定のデータ ポイントを強調するため、プロット記号を各データ プロットに追加できます。プロット記号には以下のものがあります。

- 単純な形状（正方形、円、クロスなど）
- ファイルから読み込んだ任意の図形
- 内蔵の図形

プロット記号には mark インスタンスを通じて指定できます。たとえば以下の通りです。

```
$lineplot->mark->SetColor("red");
```

異なったプロット記号を選択する場合、[PlotMark::SetType\(\)](#) メソッドを使用します。

単純な形状には、以下のものが定義されています。

- MARK_SQUARE
- MARK_UTRIANGLE
- MARK_DTRIANGLE
- MARK_DIAMOND
- MARK_CIRCLE
- MARK_FILLED CIRCLE

- MARK_CROSS
- MARK_STAR
- MARK_X
- MARK_LEFTTRIANGLE
- MARK_RIGHTTRIANGLE
- MARK_FLASH

画像を読み込む場合、以下の定義を使用します。

- MARK_IMG

この場合は、別途プロットの画像に使用するのファイル名と、オプションでスケール サイズを指定します。例えば次のようになります。

```
$lineplot->mark->SetTYPE(MARK_IMG, "myimage.jpg", 1.5);
```

もし定義済みの画像を使用する場合、以下のものが利用可能です。ただし、すべての画像ですべてのカラーが使用可能でないということに注意してください。使用できるカラーの一覧は下記に列挙します。

以下の形状のプロットマークを使用できます。

1. MARK_SQUARE — 塗りつぶされた正方形
2. MARK_UTRIANGLE — 上向きの三角形
3. MARK_DTRIANGLE — 下向きの三角形
4. MARK_DIAMOND — ダイヤモンド型
5. MARK_CIRCLE — 円形
6. MARK_FILLED CIRCLE — 塗りつぶされた円
7. MARK_CROSS — 十字型
8. MARK_STAR — 星型
9. MARK_X — X 文字
10. MARK_FLASH — フラッシュ型
11. MARK_IMAGE — 外部ファイルを使用する場合の画像形式

定義済みの画像を使用する場合は、2番目の引数でカラーを指定する必要があります。

これらの画像には数種類のサイズの画像を持つものもあります。この理由は、3番目の引数でスケールを指定すると、オリジナルの画像にピクセル補完を行って表示するため画像が劣化してしまうからです。スケールを 1.0 より小さくすると、より綺麗な画像になります。

また、スケール機能は GD 2 で優れた品質で描画されます。

定義済みの画像では、以下のカラーが使用できます。

形式	説明	使用できるカラー
MARK_IMG_PUSHPIN, MARK_IMG_SPUSHPIN	プッシュピン型 画像	'red','blue','green','pink','orange'
MARK_IMG_LPUSHPIN	大サイズのプッシュ ピン型画像	'red','blue','green','pink','orange'
MARK_IMG_BALL, MARK_IMAGE_SBALL	丸い 3D ボール	'bluegreen','cyan','darkgray','greengray', 'gray','graypurple','green','greenblue','lightblue', 'lightred','navy','orange','purple','red','yellow'
MARK_IMAGE_MBALL	中 サイズ の 丸い 3D ボール	'blue','bluegreen','brown','cyan', 'darkgray','greengray','gray','green', 'greenblue','lightblue','lightred', 'purple','red','white','yellow'
MARK_IMAGE_LBALL	大 サイズ の 丸い 3D ボール	'blue','lightblue','brown','darkgreen','green', 'purple','red','gray','yellow','silver','gray'
MARK_IMAGE_SQUARE	3D 正方形	'bluegreen','blue','green', 'lightblue','orange','purple','red','yellow'
MARK_IMG_STAR	3D 星型	'bluegreen','lightblue','purple','blue','green', 'pink','red','yellow'
MARK_IMG_DIAMOND	3D ダイヤモンド型	'lightblue','darkblue','gray', 'blue','pink','purple','red','yellow'
MARK_IMG_BEVEL	3D 傾斜型	'green','purple','orange','red','yellow'

以下に、これらの機能を使用した例を掲載します。

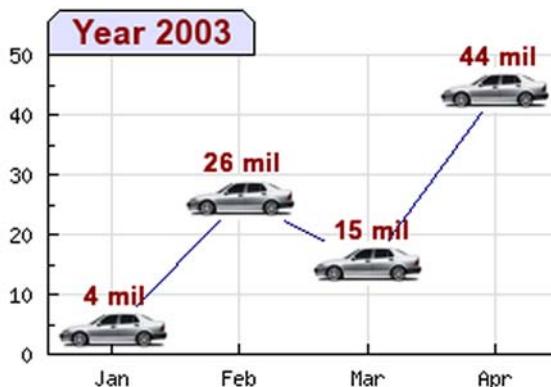


図 29:任意の画像をプロットマークに使用する

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");
include ("./jpgraph_scatter.php");

$datay1 = array(4,26,15,44);

// Setup the graph
$graph = new Graph(300,200);
$graph->SetMarginColor('white');
$graph->SetScale("texlin");
$graph->SetFrame(false);
$graph->SetMargin(30,5,25,20);

// Setup the tab
$graph->tabtitle->Set(' Year 2003 ');
$graph->tabtitle->SetFont(FF_ARIAL,FS_BOLD,13);
$graph->tabtitle->SetColor('darkred','#E1E1FF');

// Enable X-grid as well
$graph->xgrid->Show();

// Use months as X-labels
$graph->xaxis->SetTickLabels($gDateLocale->GetShortMonth());

// Create the plot

```

```

$p1 = new LinePlot($datay1);
$p1->SetColor("navy");

// Use an image of favourite car as marker
$p1->mark->SetType(MARK_IMG,'saab_95.jpg',0.5);

// Displays value on top of marker image
$p1->value->SetFormat('%d mil');
$p1->value->Show();
$p1->value->SetColor('darkred');
$p1->value->SetFont(FF_ARIAL,FS_BOLD,10);
// Increase the margin so that the value is printed above the
// img marker
$p1->value->SetMargin(14);

// Incent the X-
// scale so the first and last point doesn't
// fall on the edges
$p1->SetCenter();

$graph->Add($p1);

$graph->Stroke();

?>

```

Using Builtin PlotMarks

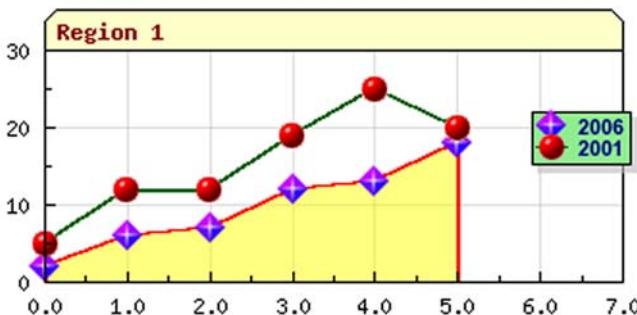


図 30: 定義済みの画像を使用する

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$datay1 = array(2,6,7,12,13,18);
$datay2 = array(5,12,12,19,25,20);

// Setup the graph
$graph = new Graph(350,200);
$graph->SetMargin(30,20,60,20);
$graph->SetMarginColor('white');
$graph->SetScale("linlin");

// Hide the frame around the graph
$graph->SetFrame(false);

// Setup title
$graph->title->Set("Using Builtin PlotMarks");
$graph->title->SetFont(FF_VARDANA,FS_BOLD,14);

// Note: requires jpgraph 1.12p or higher
// $graph-
>SetBackgroundGradient('blue','navy@0.5',GRAD_HOR,BGRA_D_PLOT);

```

```

$graph->tabtitle->Set('Region 1 ');
$graph->tabtitle->SetWidth(TABTITLE_WIDTHFULL);

// Enable X and Y Grid
$graph->xgrid->Show();
$graph->xgrid->SetColor('gray@0.5');
$graph->ygrid->SetColor('gray@0.5');

// Format the legend box
$graph->legend->SetColor('navy');
$graph->legend->SetFillColor('lightgreen');
$graph->legend->SetLineWeight(1);
$graph->legend->SetFont(FF_ARIAL,FS_BOLD,8);
$graph->legend->SetShadow('gray@0.4',3);
$graph->legend->SetAbsPos(15,120,'right','bottom');

// Create the line plots

$p1 = new LinePlot($datay1);
$p1->SetColor("red");
$p1->SetFillColor("yellow@0.5");
$p1->SetWeight(2);
$p1->mark->SetType(MARK_IMG,'diagonal',5,0.6);
$p1->SetLegend('2006');

```

```
$graph->Add($p1);

$p2 = new LinePlot($datay2);
$p2->SetColor("darkgreen");
$p2->SetWeight(2);
$p2->SetLegend('2001');
$p2->mark->SetType(MARK_IMG_MBALL,'red');
$graph->Add($p2);

// Add a vertical line at the end scale position '7'
$l1 = new PlotLine(VERTICAL,7);
$graph->Add($l1);

// Output the graph
$graph->Stroke();

?>
```

7.2 バー プロット

Jpgraph では 2D のバー プロットの描画もサポートします。バー プロットの使用には、jpgraph_bar.php ファイルを事前にインクルードしておく必要があります。

バー プロットの使用は非常に簡単で、これまでのサンプルで解説してきたライン プロットと同様です。[12, 8, 19, 3, 10, 5] というデータ配列があった場合、バー プロットで表示することを考えましょう。

以下に最も簡単な例を掲載します：

```
$datay=array(12,8,19,3,10,5);
$bplot = new BarPlot($datay);
$graph->Add($bplot);
```

このサンプルをこれまでの例 と比較すると、単にライン プロットを呼び出している部分 (LinePlot) をバー プロット [BarPplot\(\)](#) に置き換えているだけです。

また、理由は後述しますが、X 軸にはテキスト スケールを使用することを推奨します。この 2 つの単純な変更で、以下のイメージに表示されるバー グラフを得ます。

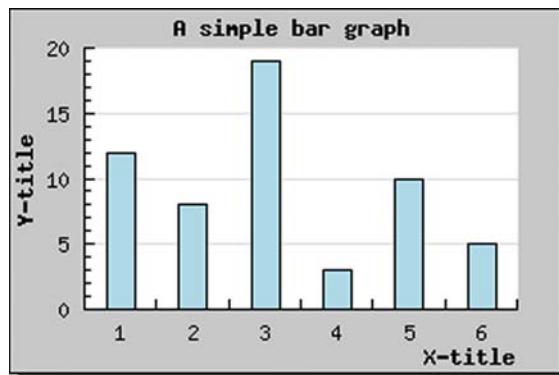


図 31:簡単なエラー バーを作成する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar pot
$bplot = new BarPlot($datay);
$graph->Add($bplot);

// Setup the titles
$graph->title->Set("A simple bar graph");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();

?>
```

当然、バー グラフの外観を変更することができます。カラーを上塗りするには、[BarPlot::SetFillColor\(\)](#) メソッドを使用します。これらの修正を適応すると、グラフの外見は下記のようになります。

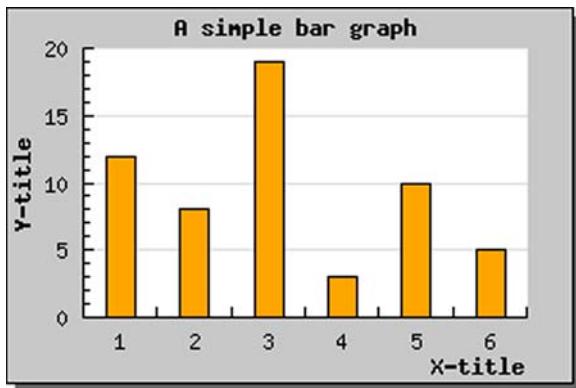


図 32:簡単なエラー バーを作成する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar plot
```

```
$bplot = new BarPlot($datay);

// Adjust fill color
$bplot->SetFillColor('orange');
$graph->Add($bplot);

// Setup the titles
$graph->title->Set("A simple bar graph");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>
```

注意:この二つのグラフでは、X スケールの表示が違っています。バープロットは、X 軸にテキストスケールを使用する場合、チェックマークの間で自動的に中央寄せされます。リニアスケールを使用する場合は、X 値の左端からプロットが描かれます。この例を、以下に表示します。

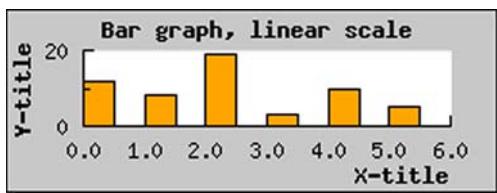


図 33:リニアな X 軸を使用したバー グラフ

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);
```

```
// Create the graph. These two calls are always required
$graph = new Graph(260,100,"auto");
$graph->SetScale("linlin");

// Add a drop shadow
```

```

$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar plot
$bplot = new BarPlot($datay);

// Adjust fill color
$bplot->SetFillColor('orange');
$graph->Add($bplot);

// Setup the titles
$graph->title->Set("Bar graph, linear scale");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>

```

7.2.1 バーの幅を調整する

JpGraph ではバーの幅を変更するなど、棒グラフの外見を簡単にカスタマイズできます。バーの幅は各目盛りの幅と比較した割合、あるいは絶対幅（ピクセル単位）で指定できます。

バーの幅を指定するには、[SetWidth\(\)](#) メソッドを使用します。また、ピクセル単位で幅を指定するには、[SetAbsWidth\(\)](#) メソッドを使用します。

以下の例は、先ほどのサンプルを使用して、目盛り幅の 100% をバーの幅に指定したものです。例は以下のようになります。

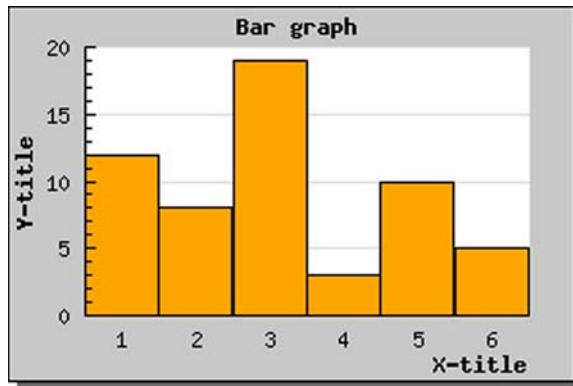


図 34: バーの幅を目盛り幅の 100% に設定する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar plot
$bplot = new BarPlot($datay);

// Adjust fill color
$bplot->SetFillColor('orange');
$bplot->SetWidth(1.0);
$graph->Add($bplot);

// Setup the titles
$graph->title->Set("Bar graph");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>

```

7.2.2 各バーの値を表示する

各バーの上部にそのバーの値を表示できます。これを行うには、バーの `value` プロパティを使用します。例として、以下の行を付け足します。

```
$barplot->value->Show();
```

これにより、以下の結果が得られます。

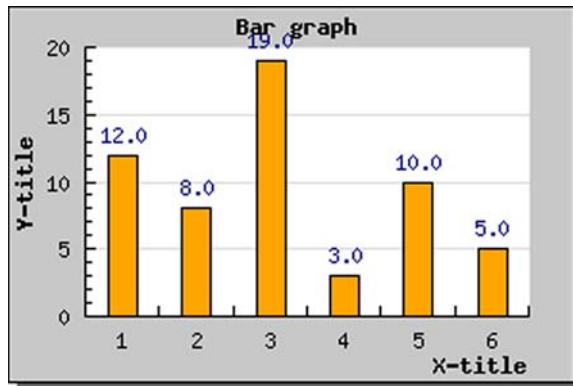


図 35:各バーの値を表示する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar pot
```

```
$bplot = new BarPlot($datay);

// Adjust fill color
$bplot->SetFillColor('orange');
$bplot->value->Show();
$graph->Add($bplot);

// Setup the titles
$graph->title->Set("Bar graph");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>
```

このグラフは少し見づらくなっています。JpGraph の自動スケール機能は、バーが枠内に入り込む最大の大きさで表示します。そのため、バーの上に値を追加すると、グラフの上部にはみ出てしまいます。これを回避するためには、[SetGrace\(\)](#) メソッドを使用して、軸から指定した割合分だけ余白を作成します。今回の場合、上部に 20% 分の余白を作成するには、以下の通りにします。

```
$graph->yaxis->scale->SetGrace(20);
```

これにより、以下の結果が得られます。

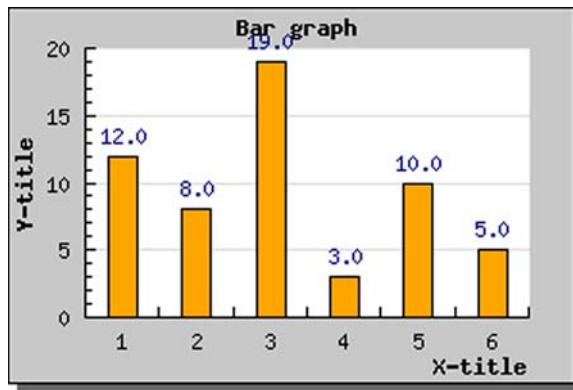


図 35:各バーの値を表示する

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar pot
    
```



```

$bplot = new BarPlot($datay);

// Adjust fill color
$bplot->SetFillColor('orange');
$bplot->value->Show();
$graph->Add($bplot);

// Setup the titles
$graph->title->Set("Bar graph");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>
    
```

このグラフは少し見づらくなっています。JpGraph の自動スケール機能は、バーが枠内に入り込む最大の大きさで表示します。そのため、バーの上に値を追加すると、グラフの上部にはみ出てしまいます。これを回避するためには、[SetGrace\(\)](#) メソッドを使用して、軸から指定した割合分だけ余白を作成します。今回の場合、上部に 20% 分の余白を作成するには、以下の通りにします。

```
$graph->yaxis->scale->SetGrace(20);
```

これにより、以下の結果が得られます。

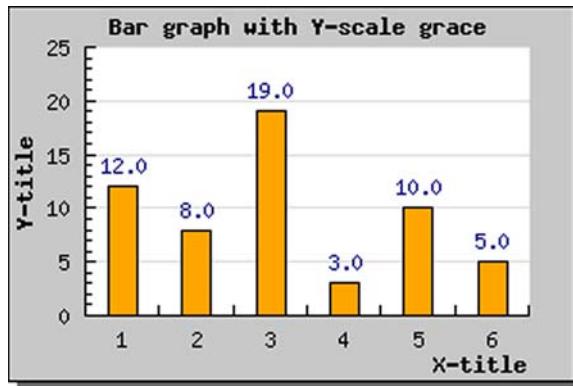


図 36:グラフの上部に余白を作り、値を適切に表示する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");
$graph->yaxis->scale->SetGrace(20);

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar pot

```

```

$bplot = new BarPlot($datay);

// Adjust fill color
$bplot->SetFillColor('orange');
$bplot->value->Show();
$graph->Add($bplot);

// Setup the titles
$graph->title->Set("Bar graph with Y-scale grace");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>

```

また、データ値を表示する位置を [BarPlot::SetValuePos\(\)](#) メソッドで変更できます。位置は、top（上部、デフォルト）、center（中央）、bottom（下部）から選択できます。以下に center を指定した場合のグラフの例を紹介します。また、今回の例では値に小数点がつかないよう、書式の調整も行いました。

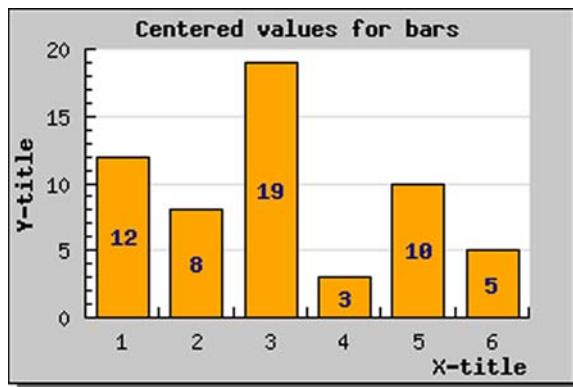


図 37:バーの中央に値を表示する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar pot
$bplot = new BarPlot($datay);

// Adjust fill color
$bplot->SetFillColor('orange');

```

```

// Setup values
$bplot->value->Show();
$bplot->value->SetFormat("%d");
$bplot->value->SetFont(FF_FONT1,FS_BOLD);

// Center the values in the bar
$bplot->SetValuePos('center');

// Make the bar a little bit wider
$bplot->SetWidth(0.7);

$graph->Add($bplot);

// Setup the titles
$graph->title->Set("Centered values for bars");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

```

```
// Display the graph
$graph->Stroke();
```

?>

JpGraph では、データ値の表示形式をより詳細に指定できます。例として、回転やフォントの変更、カラーの変更などが可能です。また、実際に表示する値を printf() 形式の書式で指定することや、書式のコールバック関数を使ってより高度な処理を行えます。例を以下に表します。

何ができるかを示すため、多くの説明無しで調査できる別な例を紹介します。ただし、0 度か 90 度以外の角度をつけたテキストを表示するためには、TrueType フォントを使用します。フォントを指定するには、SetFont() メソッドを使用します。

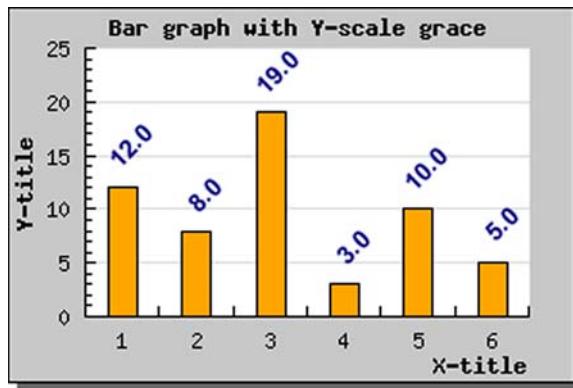


図 38:より詳細に値の書式を設定する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");
$graph->yaxis->scale->SetGrace(20);

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar plot
$bplot = new BarPlot($datay);
```

```
// Adjust fill color
$bplot->SetFillColor('orange');
$bplot->value->Show();
$bplot->value->SetFont(FF_ARIAL,FS_BOLD,10);
$bplot->value->SetAngle(45);
$bplot->value->SetFormat("%0.1f");
$graph->Add($bplot);

// Setup the titles
$graph->title->Set("Bar graph with Y-scale grace");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>
```

7.2.3 バーにドロップ シャドウを追加する

グラフの魅力を向上させるため、各バーにドロップ シャドウをつけてみましょう。この場合は、[SetFont\(\)](#) メソッドを使用します。以下にこの例を表示します。

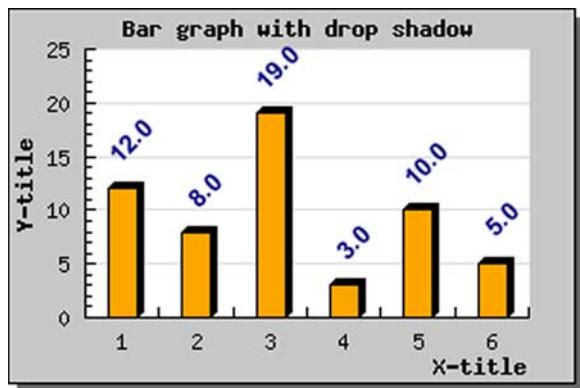


図 39:各バーにドロップ シャドウを追加する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");
$graph->yaxis->scale->SetGrace(20);

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar plot
$bplot = new BarPlot($datay);
// Adjust fill color
$bplot->SetFillColor('orange');
$bplot->SetShadow();
$bplot->value->Show();
$bplot->value->SetFont(FF_ARIAL,FS_BOLD,10);
$bplot->value->SetAngle(45);
$bplot->value->SetFormat('%0.1f');
$graph->Add($bplot);

// Setup the titles
$graph->title->Set("Bar graph with drop shadow");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>

```

7.2.4 テキストスケールのバーの配置を調整する

これまで見てきたとおり、テキストスケールの場合はバー プロットが目盛りの中央に配置されます。この動作は、[BarPlot::SetAlign\(\)](#) メソッドを呼び出すことで変更できます。

7.2.5 グループ化されたバー プロットを使用する

X 軸上の各目盛りに対して、複数のバー プロットをグループ化できます。バーはお互いの横にすぐに配置され、グループとしてそれぞれのチックマークに集中します。グループにするには、それぞれの棒グラフを作成し、それらを元に[GroupBarPlot\(\)](#) オブジェクトを作成します。

```

// バー プロットを作成
$b1plot = new BarPlot($data1y);
$b1plot->SetFillColor('orange');

$b2plot = new BarPlot($data2y);
$b2plot->SetFillColor('blue');

// グループ化されたバー プロットを作成
$gbplot = new GroupBarPlot(array($b1plot,$b2plot));

// ... そして、それをグラフに追加
$graph->Add($gbplot);

```

以下にサンプルを掲載します。

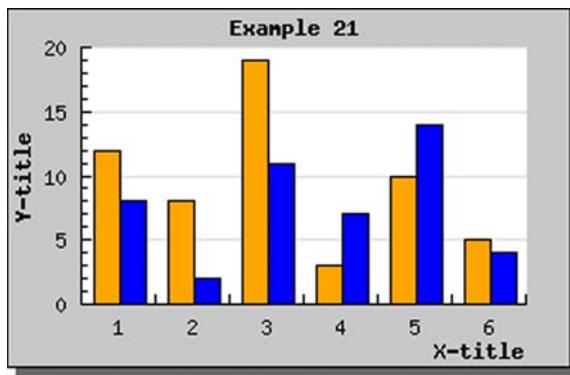


図 40: グループにしたバー プロット

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

$data1y=array(12,8,19,3,10,5);
$data2y=array(8,2,11,7,14,4);

// Create the graph. These two calls are always required
$graph = new Graph(310,200,"auto");
$graph->SetScale("textlin");

$graph->SetShadow();
$graph->img->SetMargin(40,30,20,40);

// Create the bar plots
$b1plot = new BarPlot($data1y);
$b1plot->SetFillColor("orange");
$b2plot = new BarPlot($data2y);

$b2plot->SetFillColor("blue");

// Create the grouped bar plot
$gbplot = new GroupBarPlot(array($b1plot,$b2plot));

// ...and add it to the graph
$graph->Add($gbplot);

$graph->title->Set("Example 21");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>
```

グループ化するグラフの数に制限はありません。

GroupBarPlot() の SetWidth() メソッドを使用すると、追加したプロットを合計した横幅を設定できます。また、それぞれのバーの幅は同一になります。グループ化された棒グラフの幅は、デフォルトで 70% になっています。

グループ化を 0.9 に設定した結果を以下に表示します。

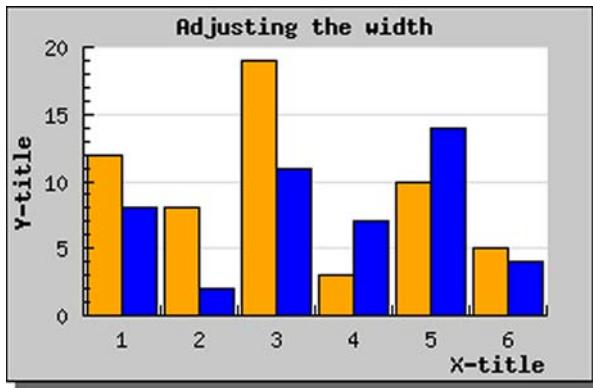


図 41: グループ化バー プロットの幅を調節する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

$data1y=array(12,8,19,3,10,5);
$data2y=array(8,2,11,7,14,4);

// Create the graph. These two calls are always required
$graph = new Graph(310,200,"auto");
$graph->SetScale("textlin");
$graph->SetShadow();

$graph->img->SetMargin(40,30,20,40);

// Create the bar plots
$b1plot = new BarPlot($data1y);
$b1plot->SetFillColor("orange");
$b2plot = new BarPlot($data2y);
$b2plot->SetFillColor("blue");

// Create the grouped bar plot
$gbplot = new GroupBarPlot(array($b1plot,$b2plot));
$gbplot->SetWidth(0.9);

// ...and add it to the graph
$graph->Add($gbplot);

$graph->title->Set("Adjusting the width");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>
```

7.2.6 積み上げバー プロットを使用する

バー プロットをグループ化し、積み上げ式にすることができます。これには、積み上げバー プロットと同様の手順で行います。グラフでは、各プロットの値が積み上げられます。

積み上げバー プロットを作成する場合は、通常のグループ化バー プロットと同様に、まず複数個のバー プロットを作成した後、[AccBarPlot\(\)](#)を呼び出します。

サンプルの積み上げバー プロットを以下に掲載します。前回のサンプルと比較して、変更行は 1 行しかありません。コードはとても似ています(実際 1 つのラインを変えるだけです)

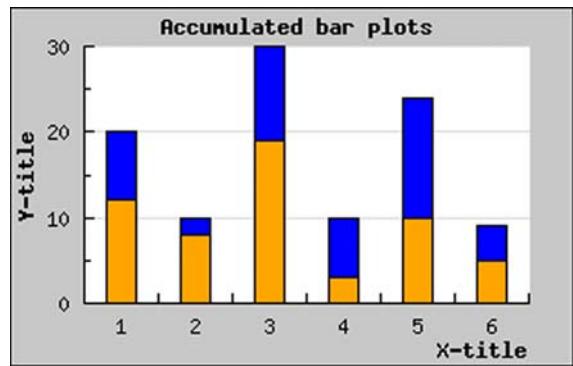


図 42:積み上げバー プロットを作成する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

setlocale (LC_ALL, 'et_EE.ISO-8859-1');

$data1y=array(12,8,19,3,10,5);
$data2y=array(8,2,11,7,14,4);

// Create the graph. These two calls are always required
$graph = new Graph(310,200,"auto");
$graph->SetScale("textlin");

$graph->SetShadow();
$graph->img->SetMargin(40,30,20,40);

// Create the bar plots
$b1plot = new BarPlot($data1y);
$b1plot->SetFillColor("orange");
$b2plot = new BarPlot($data2y);
$b2plot->SetFillColor("blue");

// Create the grouped bar plot
$gbplot = new AccBarPlot(array($b1plot,$b2plot));

// ...and add it to the graph
$graph->Add($gbplot);

$graph->title->Set("Accumulated bar plots");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
```

```
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);
```

```
// Display the graph
$graph->Stroke();
?>
```

7.2.7 グループ化された積み上げバー プロットを使用する

積み上げバー プロットをグループにすることもできます。この場合は複数の積み上げバー プロットを追加します。

```
// 4 つのバー プロットをすべて作成
$b1plot = new BarPlot($data1y);
$b1plot->SetFillColor("orange");
$b2plot = new BarPlot($data2y);
$b2plot->SetFillColor("blue");
$b3plot = new BarPlot($data3y);
$b3plot->SetFillColor("green");
$b4plot = new BarPlot($data4y);
$b4plot->SetFillColor("brown");
```

```
// 蓄積されたバー プロットを作成
$ab1plot = new AccBarPlot(array($b1plot,$b2plot));
$ab2plot = new AccBarPlot(array($b3plot,$b4plot));

// グループ化されたバー プロットを作成
$gbplot = new GroupBarPlot(array($ab1plot,$ab2plot));

// . . . そして、それをグラフに追加
$graph->Add($gbplot);
```

これにより、以下のグラフが得られます。

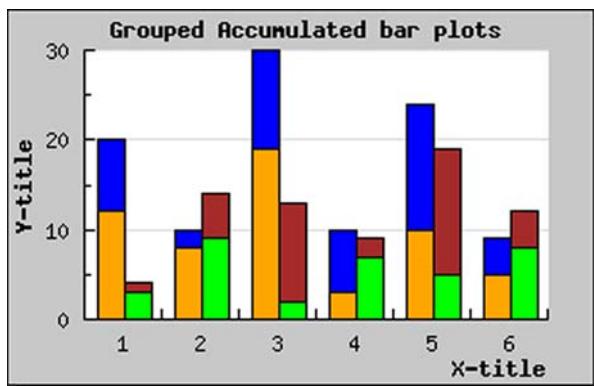


図 43:積み上げ棒グラフをグループ化する [ソース]

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

$data1y=array(12,8,19,3,10,5);
$data2y=array(8,2,11,7,14,4);
$data3y=array(3,9,2,7,5,8);
$data4y=array(1,5,11,2,14,4);

// Create the graph. These two calls are always required
$graph = new Graph(310,200,"auto");
$graph->SetScale("textlin");

$graph->SetShadow();
$graph->img->SetMargin(40,30,20,40);

$b1plot = new BarPlot($data1y);
$b1plot->SetFillColor("orange");
$b2plot = new BarPlot($data2y);
$b2plot->SetFillColor("blue");
$b3plot = new BarPlot($data3y);
$b3plot->SetFillColor("green");
$b4plot = new BarPlot($data4y);
```

```
$b4plot->SetFillColor("brown");

// Create the accumulated bar plots
$ab1plot = new AccBarPlot(array($b1plot,$b2plot));
$ab2plot = new AccBarPlot(array($b3plot,$b4plot));

// Create the grouped bar plot
$gbplot = new GroupBarPlot(array($ab1plot,$ab2plot));

// ...and add it to the graph
$graph->Add($gbplot);

$graph->title->Set("Grouped Accumulated bar plots");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>
```

7.2.8 水平のバー プロット

数多くの値を表示したい時など、バー プロットを水平にした方が適している場合があります。JpGraph では直接水平バー プロットを使用できるわけではありませんが、通常の垂直バー プロットを 90 度回転させることで簡単にこの機能を実現できます。

以下に水平バー プロットのサンプルを掲載します。

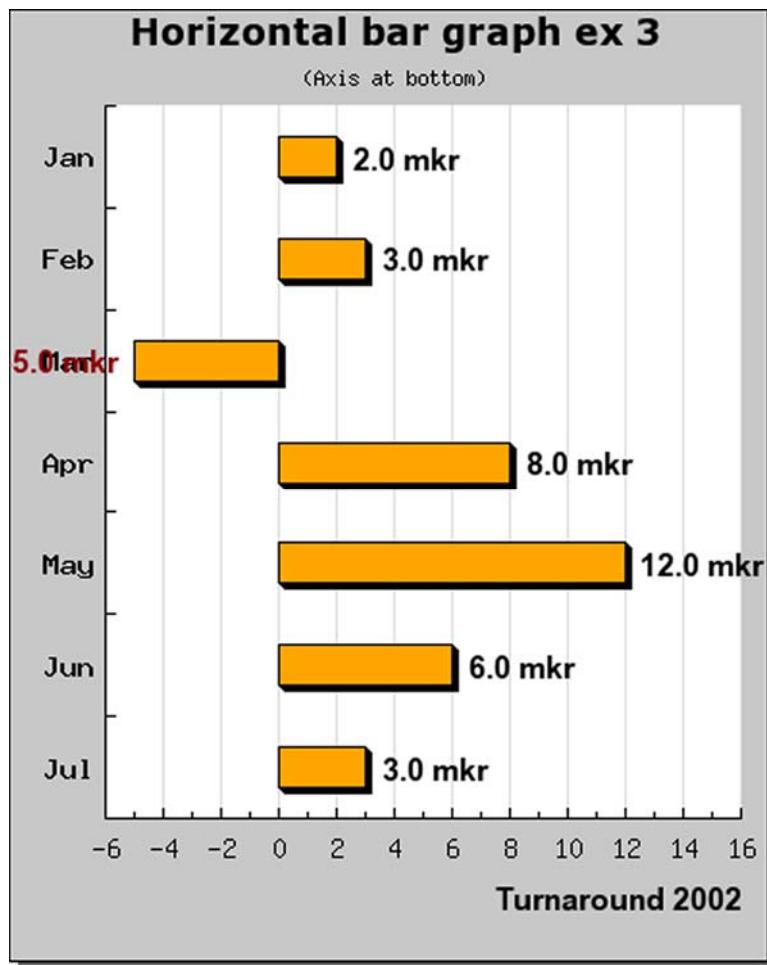


図 44:Y 軸が下部にある、一般的な水平バー プロット

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

$datay=array(2,3,-5,8,12,6,3);
$datax=array("Jan","Feb","Mar","Apr","May","Jun","Jul");

// Size of graph
$width=400;
$height=500;

// Set the basic parameters of the graph
$graph = new Graph($width,$height,'auto');
$graph->SetScale("textlin");

$top = 50;
$bottom = 80;
$left = 50;
$right = 20;
$graph->Set90AndMargin($left,$right,$top,$bottom);

$graph->xaxis->SetPos('min');

// Nice shadow
$graph->SetShadow();

// Setup title
$graph->title->Set("Horizontal bar graph ex 3");
$graph->title->SetFont(FF_VARDANA,FS_BOLD,14);
$graph->subtitle->Set("(Axis at bottom)");

// Setup X-axis
$graph->xaxis->SetTickLabels($datax);
```

```

$graph->xaxis->SetFont(FF_FONT2,FS_BOLD,12);

// Some extra margin looks nicer
$graph->xaxis->SetLabelMargin(5);

// Label align for X-axis
$graph->xaxis->SetLabelAlign('right','center');

// Add some grace to y-axis so the bars doesn't go
// all the way to the end of the plot area
$graph->yaxis->scale->SetGrace(20);

// Setup the Y-
axis to be displayed in the bottom of the
// graph. We also finetune the exact layout of the titl
e,
// ticks and labels to make them look nice.
$graph->yaxis->SetPos('max');

// First make the labels look right
$graph->yaxis->SetLabelAlign('center','top');
$graph->yaxis->SetLabelFormat('%d');
$graph->yaxis->SetLabelSide(SIDE_RIGHT);

// The fix the tick marks
$graph->yaxis->SetTickSide(SIDE_LEFT);

// Finally setup the title
$graph->yaxis->SetTitleSide(SIDE_RIGHT);
$graph->yaxis->SetTitleMargin(35);

// To align the title to the right use :
$graph->yaxis->SetTitle('Turnaround 2002','high');
$graph->yaxis->title->Align('right');

// To center the title use :
// $graph->yaxis->SetTitle('Turnaround 2002','center');
// $graph->yaxis->title->Align('center');

$graph->yaxis->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->yaxis->title->SetAngle(0);

$graph->yaxis->SetFont(FF_FONT2,FS_NORMAL);
// If you want the labels at an angle other than 0 or
90
// you need to use TTF fonts
// $graph->yaxis->SetLabelAngle(0);

// Now create a bar pot
$bplot = new BarPlot($data);
$bplot->SetFillColor("orange");
$bplot->SetShadow();

// You can change the width of the bars if you like
// $bplot->SetWidth(0.5);

// We want to display the value of each bar at the to
p
$bplot->value->Show();
$bplot->value->SetFont(FF_ARIAL,FS_BOLD,12);
$bplot->value->SetAlign('left','center');
$bplot->value->SetColor("black","darkred");
$bplot->value->SetFormat("%.1f mkr");

// Add the bar to the graph
$graph->Add($bplot);

$graph->Stroke();
?>

```

水平バー プロットを作成するためには、このサンプルのコードを詳細まで理解する必要があります。

- 単にグラフを回転しているだけでなく、回転の中心を画像の中央に設定しています。この理由は、標準設定では回転の中心点が プロット エリア の中央に設定されているからです。プロット エリアの中心は、完全なイメージの中心を必要としないので、指定された余白により回転は予言されにくい場合があります。 <
- プロット エリアのサイズは、もとの画像の幅と高さから余白を考慮して決定されます。プロット エリアが 90 度回転されると、左余白は上余白になります。これは回転したプロットの余白を概念的に指定したい場合、微妙な違いがあります。さらに簡単な回転したイメージ付きの予覚の使用には、メソッド “Set90AndMargin()” の使用を勧めます。これは、明白な方法で余白を指定すると同様にイメージの回転を許可します。

最後に、3 種類の水平バー プロットのサンプルを掲載します。1 番目のサンプルでは、Y 軸が隠れています、2 番目のサンプルでは Y 軸を上部に表示しています。

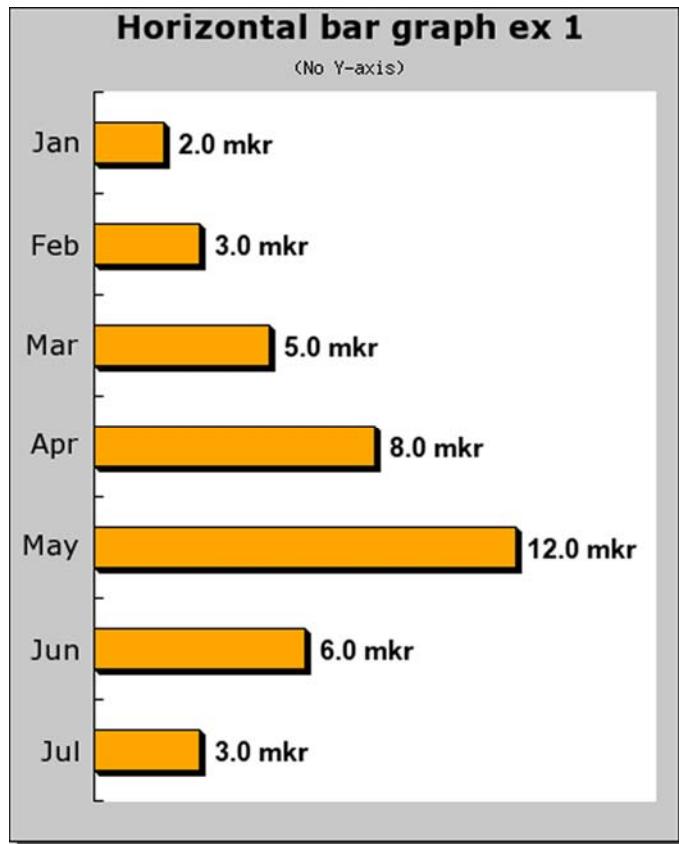


図 45:Y 軸を表示しない水平バープロット

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(2,3,5,8,12,6,3);
$datax=array("Jan","Feb","Mar","Apr","May","Jun","Jul");

// Size of graph
$width=400;
$height=500;

// Set the basic parameters of the graph
$graph = new Graph($width,$height,'auto');
$graph->SetScale("textlin");

// Rotate graph 90 degrees and set margin
$graph->Set90AndMargin(50,20,50,30);

// Nice shadow
$graph->SetShadow();

// Setup title
$graph->title->Set("Horizontal bar graph ex 1");
$graph->title->SetFont(FF_VARDANA,FS_BOLD,14);
$graph->subtitle->Set("(No Y-axis)");

// Setup X-axis
$graph->xaxis->SetTickLabels($datax);
$graph->xaxis->SetFont(FF_VARDANA,FS_NORMAL,12);

// Some extra margin looks nicer
$graph->xaxis->SetLabelMargin(10);

// Label align for X-axis
$graph->xaxis->SetLabelAlign('right','center');

// Add some grace to y-axis so the bars doesn't go
// all the way to the end of the plot area
$graph->yaxis->scale->SetGrace(20);

// We don't want to display Y-axis
$graph->yaxis->Hide();

// Now create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("orange");
$bplot->SetShadow();

// You can change the width of the bars if you like
// $bplot->SetWidth(0.5);

// We want to display the value of each bar at the top
$bplot->value->Show();
$bplot->value->SetFont(FF_ARIAL,FS_BOLD,12);
$bplot->value->SetAlign('left','center');
$bplot->value->SetColor("black","darkred");
$bplot->value->SetFormat("%.1f mkr");

// Add the bar to the graph
$graph->Add($bplot);

// .. and stroke the graph
$graph->Stroke();
?>
```

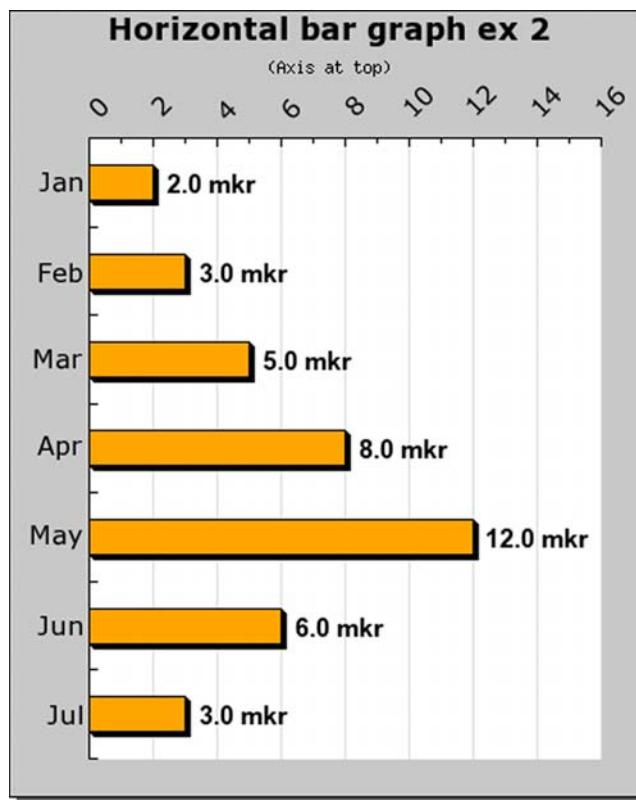


図 46:Y 軸が上部にある水平バープロット

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(2,3,5.8,12,6,3);
$datax=array("Jan","Feb","Mar","Apr","May","Jun","Jul");

$width=400;
$height=500;

// Set the basic parameters of the graph
$graph = new Graph($width,$height,'auto');
$graph->SetScale("textlin");

$top = 80;
$bottom = 30;
$left = 50;
$right = 30;
$graph->Set90AndMargin($left,$right,$top,$bottom);

// Nice shadow
$graph->SetShadow();

// Setup title
$graph->title->Set("Horizontal bar graph ex 2");
$graph->title->SetFont(FF_VERDANA,FS_BOLD,14);
$graph->subtitle->Set("(Axis at top)");

// Setup X-axis
$graph->xaxis->SetTickLabels($datax);
$graph->xaxis->SetFont(FF_VERDANA,FS_NORMAL,12);

// Some extra margin looks nicer
$graph->xaxis->SetLabelMargin(5);

// Label align for X-axis
$graph->xaxis->SetLabelAlign('right','center');

// Add some grace to y-axis so the bars doesn't go
// all the way to the end of the plot area
$graph->yaxis->scale->SetGrace(20);
$graph->yaxis->SetLabelAlign('center','bottom');
$graph->yaxis->SetLabelAngle(45);
$graph->yaxis->SetLabelFormat("%d");
$graph->yaxis->SetFont(FF_VERDANA,FS_NORMAL,12);

// We don't want to display Y-axis
//$graph->yaxis->Hide();

// Now create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("orange");
$bplot->SetShadow();

// You can change the width of the bars if you like
// $bplot->SetWidth(0.5);

// We want to display the value of each bar at the top
$bplot->value->Show();
$bplot->value->SetFont(FF_ARIAL,FS_BOLD,12);
$bplot->value->SetAlign('left','center');
$bplot->value->SetColor("black","darkred");
$bplot->value->SetFormat("%.1f mkr");

// Add the bar to the graph
$graph->Add($bplot);

$graph->Stroke();
?>

```

3番目のサンプルでは、ラベルを複数行に設定した場合の水平バープロットを掲載します。

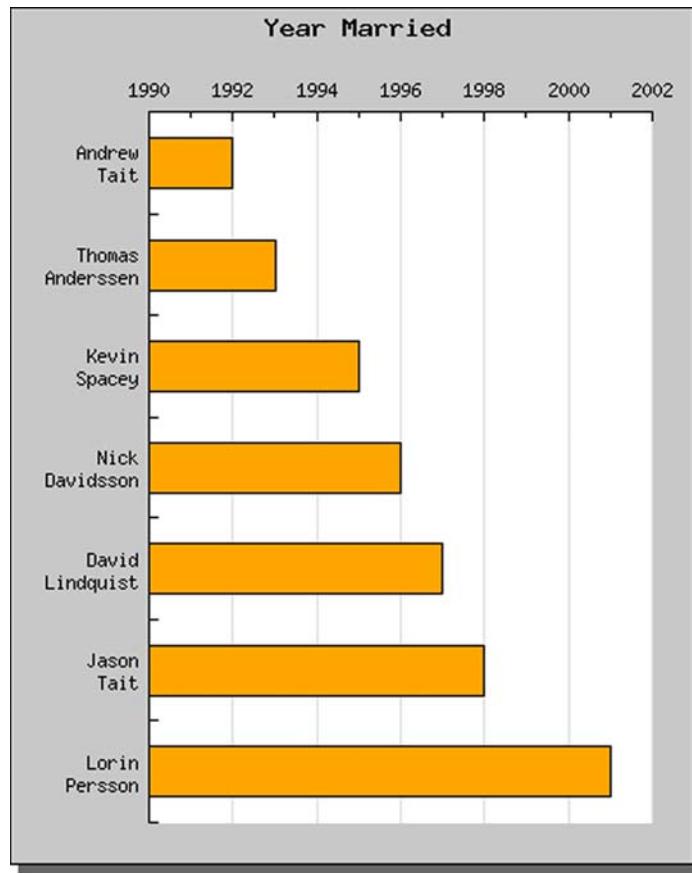


図 47: 手動の整数スケールを持ち、複数行のラベルを持つ水平バープロット

```

<?php
// $Id: horizbarex4.php,v 1.4 2002/11/17 23:59:27 ad
itus Exp $
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(1992,1993,1995,1996,1997,1998,2001);

// Size of graph
$width=400;
$height=500;

// Set the basic parameters of the graph
$graph = new Graph($width,$height,'auto');
$graph->SetScale("textlin");

$top = 60;
$bottom = 30;
$left = 80;
$right = 30;
$graph->Set90AndMargin($left,$right,$top,$bottom);

// Nice shadow
$graph->SetShadow();

// Setup labels
$lbl = array("Andrew\nTait","Thomas\nAnderssen","Kevi
n\nSpacey","Nick\nDavidsson",
"David\nLindquist","Jason\nTait","Lorin\nPersson");
$graph->xaxis->SetTickLabels($lbl);

// Label align for X-axis
$graph->xaxis->SetLabelAlign('right','center','right');

// Label align for Y-axis
$graph->yaxis->SetLabelAlign('center','bottom');

// Titles
$graph->title->Set('Year Married');

// Create a bar plot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("orange");
$bplot->SetWidth(0.5);
$bplot->SetYMin(1990);

$graph->Add($bplot);

$graph->Stroke();
?>

```

7.2.9 バー プロットにグラデーション カラーを使用する

バー プロットの各バーにグラデーション カラーを使用できます。

グラデーション カラーとは、2 色のカラーを連続的につなげたものです。グラデーションの方向は、塗りつぶしのスタイルにより指定できます。JpGraph では、以下の 8 種類のスタイルをサポートしています。サポートされているすべてのスタイルを以下に表示します。

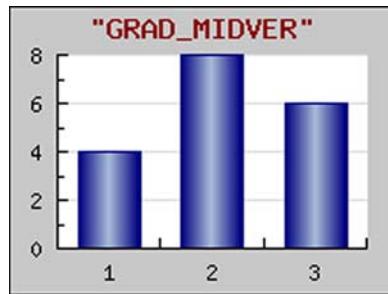


図 48

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

// We need some data
$datay=array(4,8,6);

// Setup the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->img->SetMargin(25,15,25,25);

$graph->title->Set("GRAD_MIDVER");
$graph->title->SetColor('darkred');

// Setup font for axis
$graph->xaxis->SetFont(FF_FONT1);

$graph->yaxis->SetFont(FF_FONT1);

// Create the bar plot
$bplot = new BarPlot($datay);
$bplot->SetWidth(0.6);

// Setup color for gradient fill style
$bplot-
>SetFillGradient("navy","lightsteelblue",GRAD_MIDVER);

// Set color for the frame of each bar
$bplot->SetColor("navy");
$graph->Add($bplot);

// Finally send the graph to the browser
$graph->Stroke();
?>
```

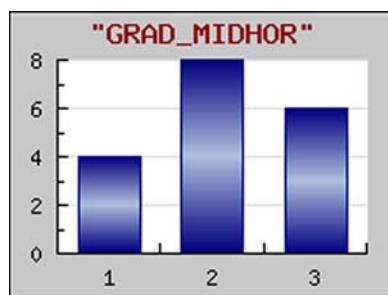


図 49

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

// We need some data
$datay=array(4,8,6);

// Setup the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");

$graph->img->SetMargin(25,15,25,25);

$graph->title->Set("GRAD_MIDHOR");
$graph->title->SetColor('darkred');

// Setup font for axis
$graph->xaxis->SetFont(FF_FONT1);
$graph->yaxis->SetFont(FF_FONT1);

// Create the bar plot
$bplot = new BarPlot($datay);
```

直角のXYプロットを使用する

```
$bplot->SetWidth(0.6);

// Setup color for gradient fill style
$bplot-
>SetFillGradient("navy","lightsteelblue",GRAD_MIDHOR);

// Set color for the frame of each bar
```

```
$bplot->SetColor("navy");
$graph->Add($bplot);

// Finally send the graph to the browser
$graph->Stroke();
?>
```

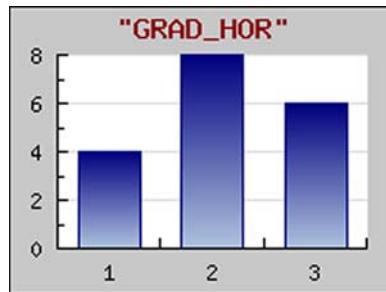


図 50

```
<?php

include ("../jpgraph.php");
include ("../jpgraph_bar.php");

// We need some data
$datay=array(4,8,6);

// Setup the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->img->SetMargin(25,15,25,25);

$graph->title->Set("GRAD_HOR");
$graph->title->SetColor('darkred');

// Setup font for axis
$graph->xaxis->SetFont(FF_FONT1);
$graph->yaxis->SetFont(FF_FONT1);
```

```
$graph->yaxis->SetFont(FF_FONT1);

// Create the bar pot
$bplot = new BarPlot($datay);
$bplot->SetWidth(0.6);

// Setup color for gradient fill style
$bplot-
>SetFillGradient("navy","lightsteelblue",GRAD_HOR);

// Set color for the frame of each bar
$bplot->SetColor("navy");
$graph->Add($bplot);

// Finally send the graph to the browser
$graph->Stroke();
?>
```

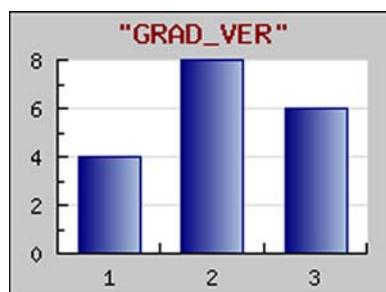


図 51

```
<?php

include ("../jpgraph.php");
include ("../jpgraph_bar.php");

// We need some data
$datay=array(4,8,6);

// Setup the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->img->SetMargin(25,15,25,25);

$graph->title->Set("GRAD_VER");
```

```
$graph->title->SetColor('darkred');

// Setup font for axis
$graph->xaxis->SetFont(FF_FONT1);
$graph->yaxis->SetFont(FF_FONT1);

// Create the bar pot
$bplot = new BarPlot($datay);
$bplot->SetWidth(0.6);

// Setup color for gradient fill style
$bplot-
>SetFillGradient("navy","lightsteelblue",GRAD_VER);
```

```
// Set color for the frame of each bar
$bplot->SetColor("navy");
$graph->Add($bplot);

// Finally send the graph to the browser
$graph->Stroke();
?>
```

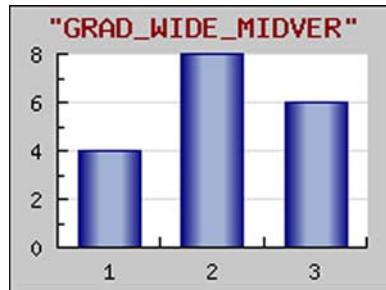


図 53

```
<?php

include ("./jpgraph.php");
include ("./jpgraph_bar.php");

// We need some data
$datay=array(4,8,6);

// Setup the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->img->SetMargin(25,15,25,25);

$graph->title->Set("GRAD_WIDE_MIDVER");
$graph->title->SetColor('darkred');

// Setup font for axis
$graph->xaxis->SetFont(FF_FONT1);
$graph->yaxis->SetFont(FF_FONT1);

$graph->yaxis->SetFont(FF_FONT1);

// Create the bar plot
$bplot = new BarPlot($datay);
$bplot->SetWidth(0.6);

// Setup color for gradient fill style
$bplot-
>SetFillGradient("navy","lightsteelblue",GRAD_WIDE_MIDVER);

// Set color for the frame of each bar
$bplot->SetColor("navy");
$graph->Add($bplot);

// Finally send the graph to the browser
$graph->Stroke();
?>
```

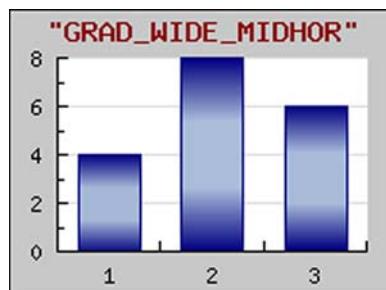


図 53

```
<?php

include ("./jpgraph.php");
include ("./jpgraph_bar.php");

// We need some data
$datay=array(4,8,6);

// Setup the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->img->SetMargin(25,15,25,25);

$graph->title->Set("GRAD_WIDE_MIDHOR");
$graph->title->SetColor('darkred');

// Setup font for axis
$graph->xaxis->SetFont(FF_FONT1);
$graph->yaxis->SetFont(FF_FONT1);

$graph->yaxis->SetFont(FF_FONT1);

// Create the bar plot
$bplot = new BarPlot($datay);
$bplot->SetWidth(0.6);

// Setup color for gradient fill style
$bplot-
>SetFillGradient("navy","lightsteelblue",GRAD_WIDE_MIDHOR);

// Set color for the frame of each bar
$bplot->SetColor("navy");
$graph->Add($bplot);

// Finally send the graph to the browser
$graph->Stroke();
?>
```



図 54

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

// We need some data
$datay=array(4,8,6);

// Setup the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->img->SetMargin(25,15,25,25);

$graph->title->Set("GRAD_CENTER");
$graph->title->SetColor('darkred');

// Setup font for axis
$graph->xaxis->SetFont(FF_FONT1);
```

```
$graph->yaxis->SetFont(FF_FONT1);

// Create the bar pot
$bplot = new BarPlot($datay);
$bplot->SetWidth(0.6);

// Setup color for gradient fill style
$bplot-
>SetFillGradient("navy","lightsteelblue",GRAD_CENTER);

// Set color for the frame of each bar
$bplot->SetColor("navy");
$graph->Add($bplot);

// Finally send the graph to the browser
$graph->Stroke();
?>
```

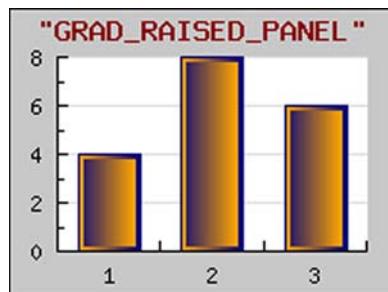


図 55

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

// We need some data
$datay=array(4,8,6);

// Setup the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->img->SetMargin(25,15,25,25);

$graph->title->Set("GRAD_RAISED_PANEL");
$graph->title->SetColor('darkred');

// Setup font for axis
```

```
$graph->xaxis->SetFont(FF_FONT1);
$graph->yaxis->SetFont(FF_FONT1);

// Create the bar pot
$bplot = new BarPlot($datay);
$bplot->SetWidth(0.6);

// Setup color for gradient fill style
$bplot->SetFillGradient('navy','orange',GRAD_RAISED_PANEL);

// Set color for the frame of each bar
$bplot->SetColor("navy");
$graph->Add($bplot);

// Finally send the graph to the browser
$graph->Stroke();
?>
```

グラデーション カラーを指定する場合は、[BarPlot::SetFillGradient\(\)](#) メソッドを使用します。詳細はクラス リファレンスを参照してください。

グラデーション カラーを使用する場合は、以下の点に注意してください。

- グラデーション処理には非常に多くの計算処理が必要です。グラデーションを使った大きなプロットは、1 色の処理の 6 倍かかります。グラフの生成に数分かかるような場合は、JpGraph のキャッシュ機能の利用が役に立つでしょう。
- グラデーションはより多くの色を使用するため、画像のカラーパレットがより大きくなり、その結果画像のサイズが大きくなります。アンチエイリアス処理多くの色を使用する可能性があるので、グラデーション処理された画像にアンチエイリアス処理を使用すると、深刻な影響を受ける可能性があります。そのためカラーパレットは、必要な数より多く用意しておくとよいでしょう。グラデーションカラーを使用する場合は TrueColor を使用してください。

これは、指定した色が別の色で表示されてしまうという現象を回避するためです。これは JpGraph のバグではありません。とりわけアンチエイリアスが使用可能になっている場合や、多くの色を使用している場合に良く見られます。そのため、線の角度にもありますが、アンチエイリアスを使用する場合は実際に使用される色の数を予測することは不可能です。

7.2.10 一部を塗りつぶしたバー プロットの作成

一部だけを塗りつぶしたバー プロット（セミ フィルド バー プロット）は、原則的には通常の塗りつぶされたバープロットと同じですが、X 座標の範囲を指定して塗りつぶすことができる追加機能です。これを使用した図を以下に表示します。

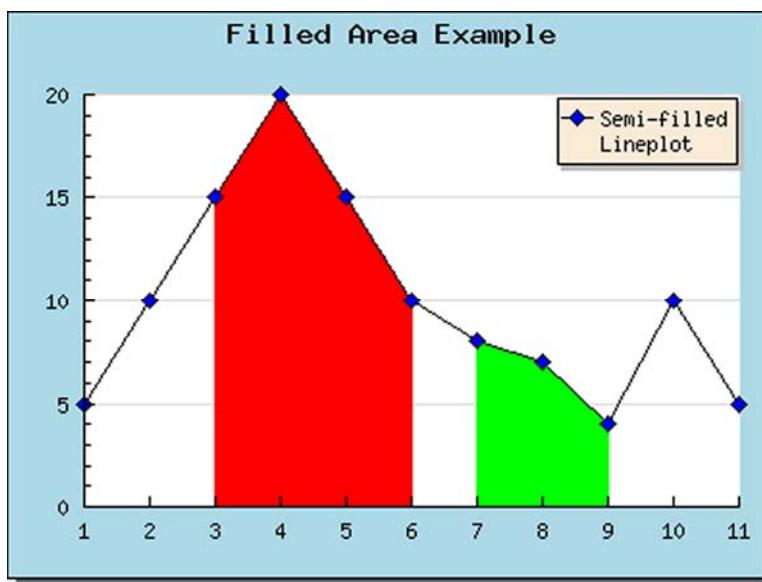


図 56:一部を塗りつぶしたライン プロット

```
<?php
include("../jpgraph.php");
include("../jpgraph_line.php");
// Some data
$ydata = array(5,10,15,20,15,10,8,7,4,10,5);
// Create the graph
$graph= new Graph(400,300,"auto");
$graph->SetScale("textlin");
$graph->SetShadow(true);
```

```

$graph->SetMarginColor("lightblue");

// Setup format for legend
$graph->legend->SetFillColor("antiquewhite");
$graph->legend->SetShadow(true);

// Setup title
$graph->title->Set("Filled Area Example");
$graph->title->SetFont(FF_FONT2,FS_BOLD);

// Setup semi-filled line plot
$lineplot = new LinePlot($ydata);
$lineplot->SetLegend("Semi-filled\nLineplot");

// Set line color
$lineplot->SetColor("black");

```

```

// Setup the two areas to be filled
$lineplot->AddArea(2.5,LP_AREA_FILLED,"red");
$lineplot->AddArea(6.8,LP_AREA_FILLED,"green");

// Display the marks on the lines
$lineplot->mark->SetType(MARK_DIAMOND);
$lineplot->mark->SetSize(8);
$lineplot->mark->Show();

// add plot to the graph
$graph->Add($lineplot);

// display graph
$graph->Stroke();
?>

```

この例では、塗りつぶす範囲を 2 つ指定了しました。[AddArea\(\)](#)メソッドを使用することで塗りつぶすエリアを追加し、そのエリアの色と範囲を指定することができます。

7.3 エラー プロット

エラー プロットは、データ ポイントの値の誤差を表すために使用します。エラー プロットは、各 X 値に対して、最小と最大の Y 値を与えることで実現されます。

エラー プロットを使用する前に、必ず `jpgraph_error.php` をスクリプトにインクルードする必要があります。

以下のサンプルでは、簡単なエラー バーを作成します。このサンプルでは 5 点の X 値があるため、10 個の Y 値が必要です。今回は、エラー バーを赤色にし、2 ピクセルの幅に設定しました。エラー プロットの作成には、[ErrorPlot\(\)](#) オブジェクトを使用します。エラー プロット オブジェクトの使い方はライン プロットと同様です。この実行例を以下に表示します。

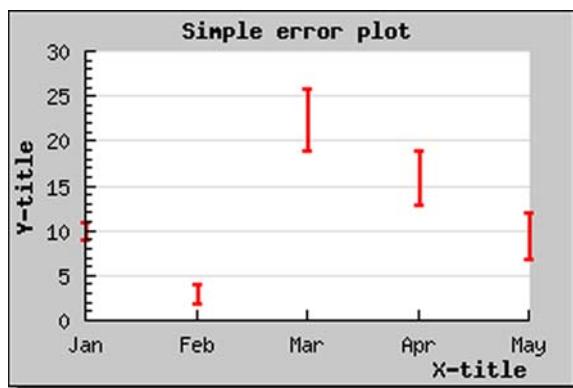


図 57:簡単なエラー バーを作成する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_error.php");

$errdatay = array(11,9,2,4,19,26,13,19,7,12);

// Create the graph. These two calls are always required

```

```

$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

$graph->img->SetMargin(40,30,20,40);
$graph->SetShadow();

// Create the error plot
$errplot=new ErrorPlot($errdatay);
$errplot->SetColor("red");

```

```
$errplot->SetWeight(2);
// Add the plot to the graph
$graph->Add($errplot);

$graph->title->Set("Simple error plot");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
```

```
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$datax = $gDateLocale->GetShortMonth();
$graph->xaxis->SetTickLabels($datax);

// Display the graph
$graph->Stroke();
?>
```

グラフの中に非常に見づらいエラー バーがあることに気づいたでしょうか。グラフの X スケールがエラー バーの数とちょうど同じになっているため、1 番目のエラー バーが Y 軸上に描画されてしまっています。これを調節するためには、[SetCenter\(\)](#) メソッドを使用して、X 軸のスケールを調整します。

以下にこの機能を使って X 軸を調節したときの画面を掲載します。

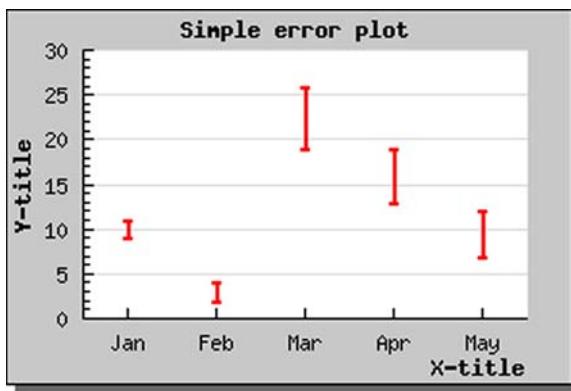


図 58:X 軸を調節して、グラフの縁にエラー バーが重ならないようにする

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_error.php");

$errdatay = array(11,9,2,4,19,26,13,19,7,12);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

$graph->img->SetMargin(40,30,20,40);
$graph->SetShadow();

// Create the error plot
$errplot=new ErrorPlot($errdatay);
$errplot->SetColor("red");
$errplot->SetWeight(2);
```

```
$errplot->SetCenter();

// Add the plot to the graph
$graph->Add($errplot);

$graph->title->Set("Simple error plot");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$datax = $gDateLocale->GetShortMonth();
$graph->xaxis->SetTickLabels($datax);

// Display the graph
$graph->Stroke();
?>
```

7.3.1 ライン エラー プロットを使用する

ライン エラー プロットは、エラー プロットに加えてエラー点の平均値を線で結合したものです。この形式のプロットは、エラープロットとほとんど同じように使用できます。ライン エラー プロットを作成する場合は、jpgraph_line.php をインクルードして、[ErrorLinePlot\(\)](#) インスタンスを作成します。ライン エラー プロットは、LinePlot クラスを使用してラインを描画します。

描画するラインのプロパティを調節するには、ErrorLinePlot オブジェクトの line プロパティを使用します。たとえば、ラインの幅を 2 ピクセルにして、そのカラーを青色に設定する場合は以下の通りになります。

```
$elplot->line->SetWeight(2);
$elplot->line->SetColor("blue");
```

これで、以下のグラフが生成されます。

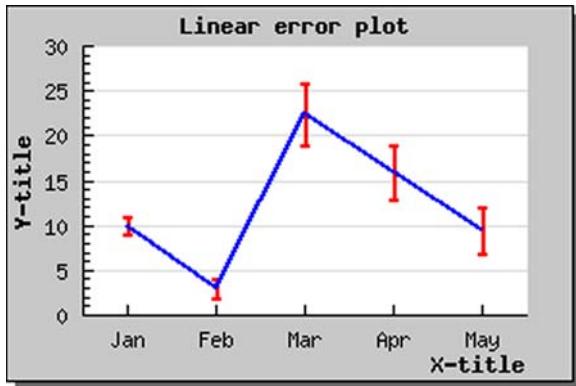


図 59: ライン エラー プロットの例

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");
include ("../jpgraph_error.php");

$errdatay = array(11,9,2,4,19,26,13,19,7,12);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

$graph->img->SetMargin(40,30,20,40);
$graph->SetShadow();

// Create the linear plot
$errplot=new ErrorLinePlot($errdatay);
$errplot->SetColor("red");
$errplot->SetWeight(2);
$errplot->SetCenter();
$errplot->line->SetWeight(2);

$errplot->line->SetColor("blue");
// Add the plot to the graph
$graph->Add($errplot);

$graph->title->Set("Linear error plot");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$datax = $gDateLocale->GetShortMonth();
$graph->xaxis->SetTickLabels($datax);

// Display the graph
$graph->Stroke();
?>
```

もちろん、ライン プロット、エラー プロットそれぞれに対して凡例を追加できます。そこで、以下は、レッド エラー バーの凡例 “Min/Max”と 追加しなければならないブルー ラインの凡例 “Average”が欲しい場合の例です。

```
$errplot->SetLegend("Min/Max");
$errplot->line->SetLegend("Average");
```

これにより、以下のグラフが作成されます。今回はデフォルトの位置に凡例を設置しました。

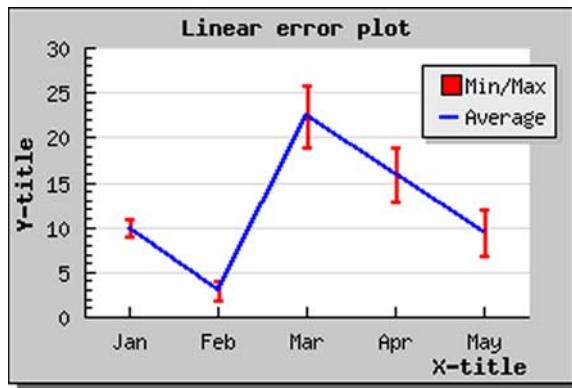


図 60: ライン エラー プロットに凡例を追加する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");
include ("../jpgraph_error.php");

$errdatay = array(11,9,2,4,19,26,13,19,7,12);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

$graph->img->SetMargin(40,30,20,40);
$graph->SetShadow();

// Create the linear plot
$errplot=new ErrorLinePlot($errdatay);
$errplot->SetColor("red");
$errplot->SetWeight(2);
$errplot->SetCenter();
$errplot->line->SetWeight(2);
$errplot->line->SetColor("blue");

// Setup the legends
$errplot->SetLegend("Min/Max");
$errplot->line->SetLegend("Average");

// Add the plot to the graph
$graph->Add($errplot);

$graph->title->Set("Linear error plot");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$datax = $gDateLocale->GetShortMonth();
$graph->xaxis->SetTickLabels($datax);

// Display the graph
$graph->Stroke();
?>

```

7.4 散布プロット

散布プロットは非常に簡単に作成できます。X 座標と Y 座標で指定された点をプロットします。ライン プロットと同様、イメージ上には各ポイントが描画されます。また、それぞれのポイントを線分で結ぶこともできます。

注意: X、Y プロットとして利用できるのは散布プロットだけですが、バープロットとライン プロットでも X、Y 座標を使用して位置を決定できます。

通常の使用では X 座標を指定しますが、X スケールにテキスト スケールを使用することもできます。これは最後に説明する、インパルス形式の散布プロットを使用する際に便利です。

散布プロットは `jpgraph_scatter.php` をインクルードして、`ScatterPlot()` のインスタンスを作成することで作成できます。また、マーカーを指定するには `mark` 変数を使用します。デフォルトでは、小さな円が使用されます。

散布プロットを作成するには、インスタンスを生成する必要があります。

デフォルトの値を使用する、簡単な例を下に表示します。

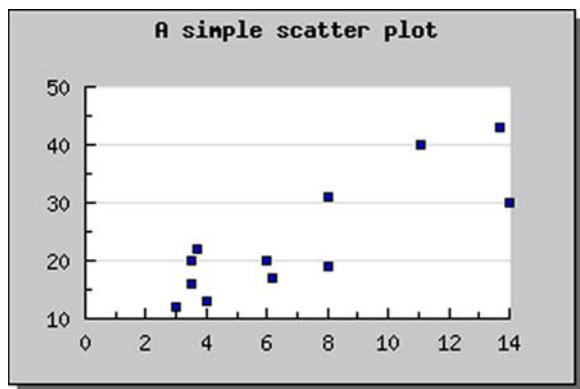


図 61:簡単な散布プロット

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_scatter.php");

$datax = array(3.5,3.7,3.4,6.2,6,3.5,8,14,8,11.1,13.7);
$datay = array(20,22,12,13,17,20,16,19,30,31,40,43);

$graph = new Graph(300,200,"auto");
$graph->SetScale("linlin");

$graph->img->SetMargin(40,40,40,40);

$graph->SetShadow();

$graph->title->Set("A simple scatter plot");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

$sp1 = new ScatterPlot($datay,$datax);

$graph->Add($sp1);
$graph->Stroke();

?>
```

マーカーのサイズやカラーを設定することで、異なった効果を得られます。

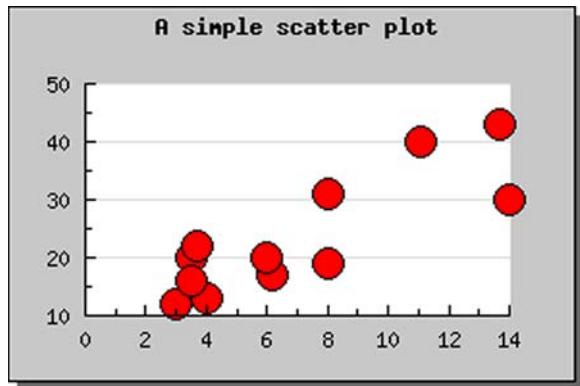


図 62:マーカーを独自に設定した散布プロット

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_scatter.php");

$datax = array(3.5,3.7,3.4,6.2,6,3.5,8,14,8,11.1,13.7);
$datay = array(20,22,12,13,17,20,16,19,30,31,40,43);

$graph = new Graph(300,200,"auto");
$graph->SetScale("linlin");

$graph->img->SetMargin(40,40,40,40);
$graph->SetShadow();

$graph->title->Set("A simple scatter plot");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

$sp1 = new ScatterPlot($datay,$datax);
$sp1->mark->SetType(MARK_FILLEDIRCLE);
$sp1->mark->SetFillColor("red");
$sp1->mark->SetWidth(8);

$graph->Add($sp1);
$graph->Stroke();

?>
```

また、インパルス形式の散布プロットを作成できます。これは、通常の散布プロットとは異なり、Y = 0 のベース ラインからラインが引かれます。散布プロットをインパルス形式に変更する場合は、[SetImpuls\(\)](#) メソッドを使用します。

インパルス形式のプロットは、デジタル信号処理の際、信号を描画する時に使用されます。以下に、簡単なインパルス形式のプロットを紹介します。

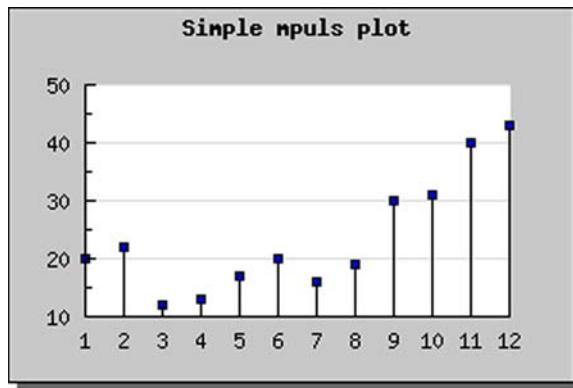


図 63:簡単なインパルス形式のプロット

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_scatter.php");

$datay = array(20,22,12,13,17,20,16,19,30,31,40,43);

$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

$graph->SetShadow();
$graph->img->SetMargin(40,40,40,40);
```

```
$graph->title->Set("Simple mpuls plot");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

$sp1 = new ScatterPlot($datay);
$sp1->mark->SetType(MARK_SQUARE);
$sp1->SetImpuls();

$graph->Add($sp1);
$graph->Stroke();

?>
```

次に、インパルス形式のプロットのカラーや線幅を設定する方法を紹介します。

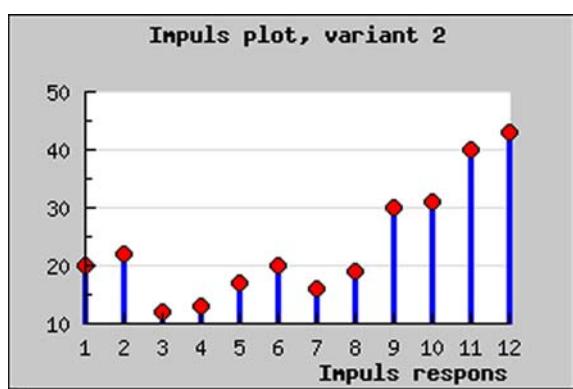


図 64:カスタマイズしたインパルス形式のプロット

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_scatter.php");

$datay = array(20,22,12,13,17,20,16,19,30,31,40,43);

$graph = new Graph(300,200,"auto");
```

```
$graph->SetScale("textlin");

$graph->SetShadow();
$graph->img->SetMargin(40,40,40,40);

$graph->title->Set("Impuls plot, variant 2");
$graph->title->SetFont(FF_FONT1,FS_BOLD);
```

直角の XY プロットを使用する

```
$graph->xaxis->title->Set("Impuls respons");
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$sp1 = new ScatterPlot($datay); //,$datax);
$sp1->mark->SetType(MARK_FILLEDIRCLE);
$sp1->mark->SetFillColor("red");
$sp1->mark->SetWidth(4);
$sp1->SetImpuls();

$sp1->SetColor("blue");
$sp1->SetWeight(3);

$graph->Add($sp1);
$graph->Stroke();

?>
```

注意:インパルス形式のプロットでマークを使用しない場合、マークの形式として -1 を指定することで、インパルスのラインだけが描画されます。この方法でのみ、インパルス線を描くことができます。

最後に、より高度なインパルス プロットを掲載します。このグラフでは、Y 軸のラベルの書式を指定し、軸の位置も調節しています。

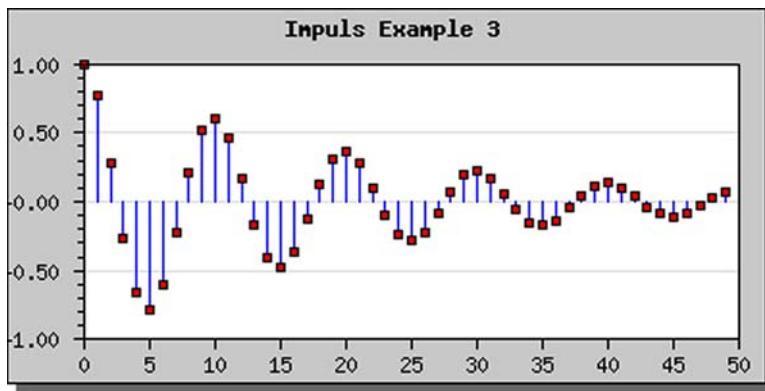


図 65:このインパルス プロットでは、X 軸を下部に配置し、Y 軸の書式を変更しています

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_scatter.php");

$numpoints=50;
$k=0.05;

// Create some data points
for($i=0; $i<$numpoints; ++$i) {
    $datay[$i]=exp(-$k*$i)*cos(2*M_PI/10*$i);
}

// A format callback function
function mycallback($i) {
    return sprintf("%02.2f",$i);
}

// Setup the basic parameters for the graph
$graph = new Graph(400,200,"auto");
$graph->SetScale("intlin");
$graph->SetShadow();
$graph->SetBox();

$graph->title->Set("Impuls Example 3");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Set format callback for labels
$graph->yaxis->SetLabelFormatCallback("mycallback");

// Set X-axis at the minimum value of Y-axis (default will be at 0)
$graph->xaxis->SetPos("min"); // "min" will position the x-axis at the minimum value of the Y-axis

// Extend the margin for the labels on the Y-axis and reverse the direction
// of the ticks on the Y-axis
$graph->yaxis->SetLabelMargin(12);
$graph->xaxis->SetLabelMargin(6);
$graph->yaxis->SetTickSide(SIDE_LEFT);
$graph->xaxis->SetTickSide(SIDE_DOWN);

// Create a new impuls type scatter plot
$sp1 = new ScatterPlot($datay);
$sp1->mark->SetType(MARK_SQUARE);
$sp1->mark->SetFillColor("red");
$sp1->SetImpuls();
$sp1->SetColor("blue");
$sp1->SetWeight(1);
$sp1->mark->SetWidth(3);

$graph->Add($sp1);

$graph->Stroke();

?>
```

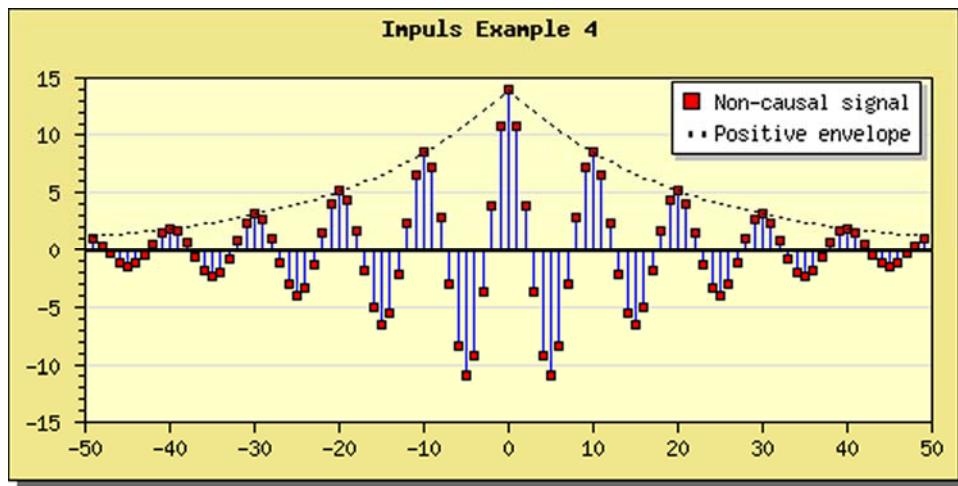


図 66:このインパルス プロットでは、ドットライン形式のライン プロットも追加しています

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_scatter.php");
include ("../jpgraph_line.php");

$numpoints=50;
$k=0.05;

// Create some data points
for($i=-$numpoints+1; $i<0; ++$i) {
    $datay[$i+$numpoints-1]=exp($k*$i)*cos(2*M_PI/10*$i)*14;
    $datayenv[$i+$numpoints-1]=exp($k*$i)*14;
    $datax[$i+$numpoints-1]=$i;
}

for($i=0; $i<$numpoints; ++$i) {
    $datay[$i+$numpoints-1]=exp(-$k*$i)*cos(2*M_PI/10*$i)*14;
    $datayenv[$i+$numpoints-1]=exp(-$k*$i)*14;
    $datax[$i+$numpoints-1]=$i;
}

// Setup the basic parameters for the graph
$graph = new Graph(500,250,"auto");
$graph->SetScale("intlin");

$graph->SetShadow();
$graph->SetBox();
$graph->title->Set("Impuls Example 4");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Set some other color then the boring default
$graph->SetColor("lightyellow");
$graph->SetMarginColor("khaki");

// Set legend box specification
$graph->legend->SetFillColor("white");
$graph->legend->SetLineWeight(2);

// Set X-axis at the minimum value of Y-
axis (default will be at 0)
$graph->xaxis->SetPos("min"); // "min" will position the x-axis at the minimum value of the Y-axis

// Extend the margin for the labels on the Y-axis and reverse the direction
// of the ticks on the Y-axis
$graph->yaxis->SetLabelMargin(12);
$graph->xaxis->SetLabelMargin(6);
$graph->yaxis->SetTickSide(SIDE_LEFT);
$graph->xaxis->SetTickSide(SIDE_DOWN);

// Add mark graph with static lines
$line = new PlotLine(HORIZONTAL,0,"black",2);
$graph->AddLine($line);

// Create a new impuls type scatter plot
$sp1 = new ScatterPlot($datay,$datax);
$sp1->mark->SetType(MARK_SQUARE);
$sp1->mark->SetFillColor("red");
$sp1->mark->SetWidth(3);

$sp1->SetImpuls();
$sp1->SetColor("blue");
$sp1->SetWeight(1);
$sp1->SetLegend("Non-causal signal");

$graph->Add($sp1);

// Create the envelope plot
$ep1 = new LinePlot($datayenv,$datax);
$ep1->SetStyle("dotted");
$ep1->SetLegend("Positive envelope");

$graph->Add($ep1);

$graph->Stroke();

?>

```

7.5 フィールド プロット

散布プロットのうち、フィールド プロットと呼ばれるものは、0 から 359 度までの方向を矢印で持つ散布ポイントを有しています。これにより、各点において 3 個の情報 (X, Y, 角度) を表現できます。さらに、各散布プロットにコールバック関数を定義でき、各ポイントのカラーも指定できます。

フィールド プロットを作成するには、通常の散布プロットと同様、[FieldPlot](#) インスタンスを作成します。このメソッドへの引数は、Y 座標、Y 座標、そして角度です。コールバック関数を指定するには、[FieldPlot::SetCallback\(\)](#) メソッドを使用します。

以下に、フィールド プロット形式の散布プロットの例を紹介します。

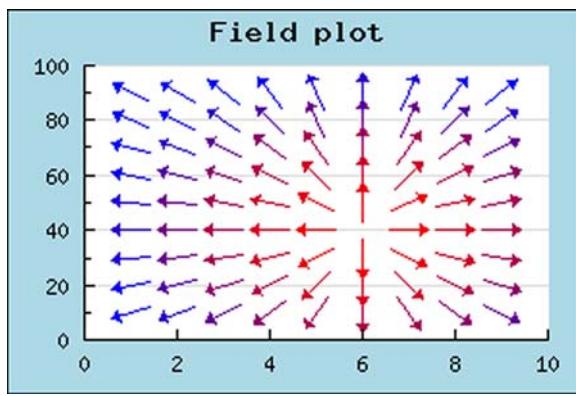


図 67:フィールド プロット形式の例

```
<?php
include ("../jgraph.php");
include ("../jgraph_scatter.php");

$polex = 6;
$poley = 40;

function FldCallback($x,$y,$a) {
    GLOBAL $polex, $poley;
    $maxr = 3000;

    // Size and arrow size is constant
    $size="";
    $arrowsize="";

    // Since we have different scales we need the data points
    // to be of the same magnitude to give it a distance
    // interpretation.
    $x *= 10;

    // Colors gets colder the further out we go from the center
    $r = ($x-$polex*10)*($x-$polex*10)+($y-$poley)*($y-$poley);
    $f = $r/$maxr;
    if( $f > 1 ) $f=1;
    $red = floor((1-$f)*255);
    $blue = floor($f*255);
    $color = array($red,0,$blue);
}

//echo "x=$x, y=$y, blue=$blue, red=$red<br>";
return array($color,$size,$arrowsize);
}

// Create data for a simulated pseudo-magnetic radient field
$datax = array();
$datay = array();
$angle = array();
for($x=1; $x < 10; ++$x ) {
    for($y=10; $y<100; $y += 10) {
        $a = -1;
        if( $x==$polex && $y==$poley ) continue;
        if( $x==$polex ) {
            if( $y > $poley ) $a=90;
            else $a = 270;
        }
        if( $y==$poley ) {
            if( $x > $polex ) $a=0;
            else $a=180;
        }
        if( $a == -1 ) {
            $d1 = $y-$poley;
            $d2 = ($polex-$x)*20;
            if( $y < $poley ) $d2 *= -1;
            $h = sqrt($d1*$d1+$d2*$d2);
            $t = -$d2/$h;
            $ac = acos($t);
            if( $y < $poley ) $ac += M_PI;
            $a = $ac * 180/M_PI;
        }
    }
}
```

```

        $datax[] = $x;
        $datay[] = $y;
        $angle[] = $a;
    }

// Setup the graph
$graph = new Graph(300,200);
$graph->SetScale("intlin",0,100,0,10);
$graph->SetMarginColor('lightblue');

// ..and titles
$graph->title->Set("Field plot");

// Setup the field plot
$fp = new FieldPlot($datay,$datax,$angle);

// Setup formatting callback
$fp->SetCallback('FldCallback');

// First size argument is length (in pixels of arrow)
// Second size argument is roughly size of arrow. Arrow size is specified as
// an integer in the range [0,9]
$fp->arrow->SetSize(20,2);
$fp->arrow->SetColor('navy');

$graph->Add($fp);

// .. and output
$graph->Stroke();
?>

```

また、フィールド プロットでは矢印のサイズを調節したり、矢印の頭の部分のサイズのみを設定することもできます。矢印のサイズはピクセル単位で指定します。矢印の頭の部分のサイズは 0 から 10 までの整数値で指定します。これらのサイズは FieldPlot::arrow::SetSize() メソッドを呼び出しこれで調節できます。

7.6 ボックスとストック チャート

JpGraph は、ストック チャート（キャンドル チャートやボックス プロットとも呼ばれています）を 2 種類の形式で作成できます。

7.6.1 ストック チャート

ストック チャートは、1 つのデータ点に対して 4 個の異なった値を表す時に使用されます。たとえば、毎日の株価の始値、高値、安値、終値を表す時に使用されます。そのためこのプロットはストック（株価）チャート、もしくはストック プロットと呼ばれます。

以下に、ストック チャートのサンプルを掲載します。

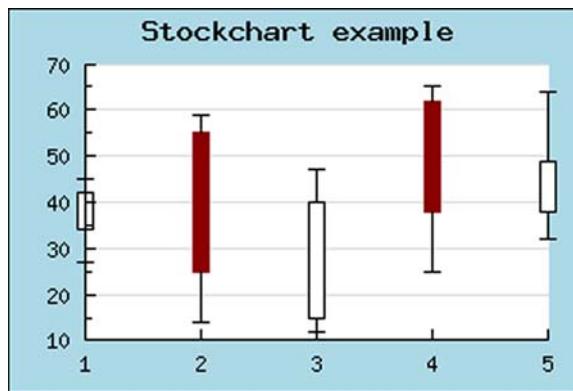


図 68:簡単なストック チャートのサンプル

```

<?php
// Example of a stock chart
include ("./jpgraph.php");
include ("./jpgraph_stock.php");

// Data must be in the format : open,close,min,max

```

```

$datay = array(
    34,42,27,45,
    55,25,14,59,
    15,40,12,47,
    62,38,25,65,
    38,49,32,64);

```

```
// Setup a simple graph
$graph = new Graph(300,200);
$graph->SetScale("texlin");
$graph->SetMarginColor('lightblue');
$graph->title->Set('Stockchart example');

// Create a new stock plot
$p1 = new StockPlot($datay);

// Width of the bars (in pixels)
$p1->SetWidth(9);

// Uncomment the following line to hide the horizontal
// end lines
// $p1->HideEndLines();

// Add the plot to the graph and send it back to the
// browser
$graph->Add($p1);
$graph->Stroke();

?>
```

ストック プロットの場合、Y 座標のデータは 4 個の値（最大値、開始値、終了値、最小値）を保持する必要があります。中央のバーが高値と安値を表しており、バーの両端の線分が出来高と終値を表しています。

データは以下の規則に従う必要があります。

- 最小値 < 最大値
- 最小値 < 開始値、終了値のうち最小の値
- 最大値 > 開始値、終了値のうち最大の値

開始値と終了値は、どちらが大きても構いません。そのため、開始値 > 終了値の場合と開始値 < 終了値の 2 種類の場合が考えられ、それぞれ別のカラーが使用できます。これらのカラーは [SetColor\(\)](#) メソッドを使用して設定できます。デフォルトでは、バーが終了値 > 開始値の場合は白色で上塗りされ、逆の場合は赤色で上塗りされます。

また、[SetWidth\(\)](#) メソッドを使用して、バーの幅をピクセル単位で設定できます。

ストック チャートの最後のバリエーションは、最小値、最大値の終値に水平ラインを表示させるかどうかを決定することです。さらに、[HideEndLine\(\)](#) メソッドを使うと、バーと最大値、最小値を結ぶ線分の表示、非表示を切り替えることができます。

7.6.2 ボックスプロット：中央線の入ったストック チャート

ストック チャートの外見を少し変更したものがボックス プロットです。ボックス プロットは、StockPlot() と同様、BoxPlot() インスタンスを用いて作成します。ボックス プロットは、ストック プロットで必要だった最大値、開始値、終了値、最小値に加え、中央値が必要になります。中央値は開始値と終了値の範囲で設定され、バーの内部に水平線が描画されます。

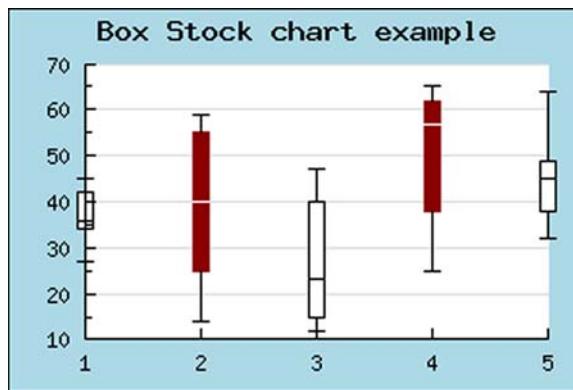


図 69:ボックス形式のストックプロットのサンプル

```
<?php
// Example of a stock chart
include ("../jpgraph.php");
include ("../jpgraph_stock.php");

// Data must be in the format : open,close,min,max,median
$datay = array(
    34,42,27,45,36,
    55,25,14,59,40,
    15,40,12,47,23,
    62,38,25,65,57,
    38,49,32,64,45);

// Setup a simple graph
$graph = new Graph(300,200);
$graph->SetScale("textlin");
$graph->SetMarginColor('lightblue');

$graph->title->Set('Box Stock chart example');

// Create a new stock plot
$p1 = new BoxPlot($datay);

// Width of the bars (in pixels)
$p1->SetWidth(9);

// Uncomment the following line to hide the horizontal end lines
// $p1->HideEndLines();

// Add the plot to the graph and send it back to the browser
$graph->Add($p1);
$graph->Stroke();

?>
```

中央線の色は [SetMedianColor\(\)](#) メソッドで変更できます。

7.6.3 ボックスとストックチャートのイメージマップ

他のプロットと同様、ストックチャートやボックスチャートでもイメージマップを使用できます。各プロットにおいて、クリックできる場所は中央にあるバーの部分になります。他のグラフと同様、[SetCSIMTargets\(\)](#) メソッドを用いて URL を設定できます。

これらのプロット形式では通常、厳密に X 値を定めることはありません。ただし、X 座標を明示的に指定することも可能です。

7.7 複数のプロット形式を組み合わせる

複数のプロットを 1 つのグラフに追加できます。この場合、プロットの種類は別のものであっても構わないでの、ラインプロットとバープロットを重ね合わせることも可能です。ただし、これらのプロットは画像として描画されるため、後に描画したものが上塗りしてしまうことに注意が必要です。プロットを指定する場合は、追加した順番に描画されます。例、一番最初に追加されたプロットは一番最初に描画されます。つまり、どのプロットが後列に表示され、どのプロットが前列に表示されるのかはプロットを追加する順番を変更することで調整できます。

まずは、フィルド ラインプロットと通常のラインプロットを組み合わせるサンプルを紹介します。

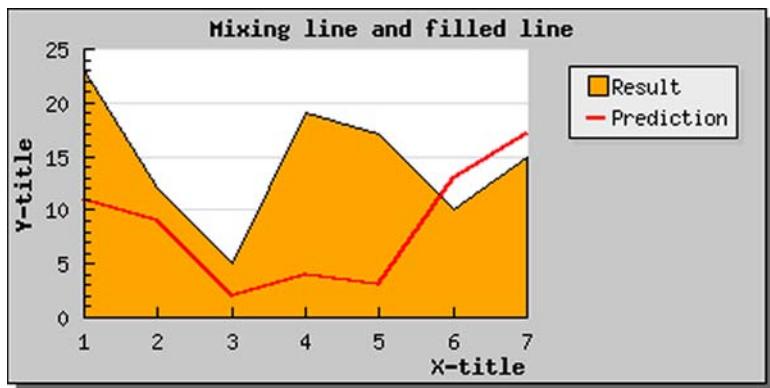


図 70:フィルド ライン プロットとライン プロットを 1 つのグラフで描画する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");
include ("./jpgraph_error.php");
include ("./jpgraph_bar.php");

$line1datay = array(11,9,2,4,3,13,17);
$line2datay = array(23,12,5,19,17,10,15);
$datax=array("Jan","Feb","Mar","Apr","May");

// Create the graph.
$graph = new Graph(400,200,"auto");
$graph->SetScale("textlin");

$graph->img->SetMargin(40,130,20,40);
$graph->SetShadow();

// Create the linear error plot
$line1plot=new LinePlot($line1datay);
$line1plot->SetColor("red");
$line1plot->SetWeight(2);
$line1plot->SetLegend("Prediction");

// Create the bar plot
$line2plot = new LinePlot($line2datay);
$line2plot->SetFillColor("orange");
$line2plot->SetLegend("Result");

// Add the plots to the graph
$graph->Add($line2plot);
$graph->Add($line1plot);

$graph->title->Set("Mixing line and filled line");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// $graph->xaxis->SetTickLabels($datax);
// $graph->xaxis->SetTextTickInterval(2);

// Display the graph
$graph->Stroke();
?>
```

もう少し複雑にしましょう。次に、バー プロットとライン プロットを組み合わせてみましょう。先ほどのバー グラフをライン プロットに変更して、外見がどのように変化するか確認してみましょう。

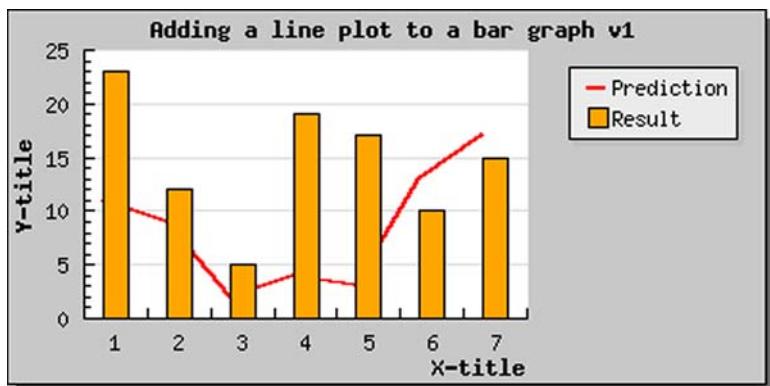


図 71:ライン プロットをバー グラフに追加する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");
include ("./jpgraph_bar.php");

$line1datay = array(23,12,5,19,17,10,15);
$datax=array("Jan","Feb","Mar","Apr","May");

// Create the graph.
$graph = new Graph(400,200,"auto");
$graph->SetScale("textlin");
```

```
$graph->img->SetMargin(40,130,20,40);
$graph->SetShadow();

// Create the linear error plot
$l1plot=new LinePlot($l1datay);
$l1plot->SetColor("red");
$l1plot->SetWeight(2);
$l1plot->SetLegend("Prediction");

// Create the bar plot
$bplot = new BarPlot($l2datay);
$bplot->SetFillColor("orange");
$bplot->SetLegend("Result");

// Add the plots to the graph
$graph->Add($l1plot);
$graph->Add($bplot);
```

```
$graph->title-
>Set("Adding a line plot to a bar graph v1");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// $graph->xaxis->SetTickLabels($datax);
// $graph->xaxis->SetTextTickInterval(2);

// Display the graph
$graph->Stroke();
?>
```

悪くありません。ライン プロットがバー プロットの裏側にきてしまったので、グラフにライン プロットを追加する順番を変更してみましょう。

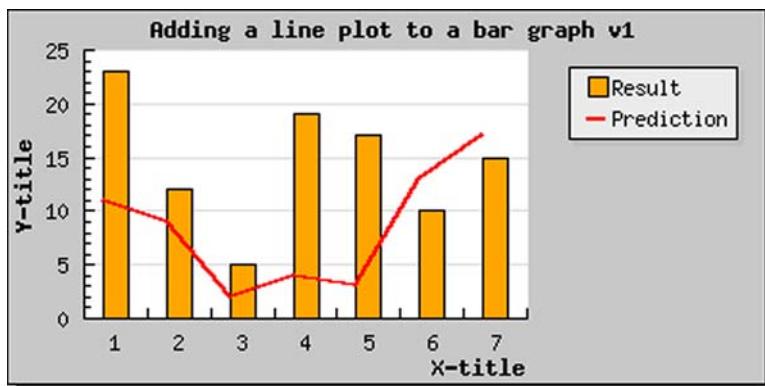


図 72:逆の順番でライン プロットをバー プロットに追加する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");
include ("./jpgraph_bar.php");

$l1datay = array(11,9,2,4,3,13,17);
$l2datay = array(23,12,5,19,17,10,15);
$datax=array("Jan","Feb","Mar","Apr","May");

// Create the graph.
$graph = new Graph(400,200,"auto");
$graph->SetScale("textlin");

$graph->img->SetMargin(40,130,20,40);
$graph->SetShadow();

// Create the linear error plot
$l1plot=new LinePlot($l1datay);
$l1plot->SetColor("red");
$l1plot->SetWeight(2);
$l1plot->SetLegend("Prediction");

// Create the bar plot
```

```
$bplot = new BarPlot($l2datay);
$bplot->SetFillColor("orange");
$bplot->SetLegend("Result");

// Add the plots to the graph
$graph->Add($bplot);
$graph->Add($l1plot);

$graph->title-
>Set("Adding a line plot to a bar graph v1");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// $graph->xaxis->SetTickLabels($datax);
// $graph->xaxis->SetTextTickInterval(2);

// Display the graph
$graph->Stroke();
?>
```

ライン プロットをバーの中央に配置したい場合は、以下の 2 通りの方法があります。

テキスト スケールを使用している場合は、[LinePlot::SetBarCenter\(\)](#)を呼び出す必要があります。

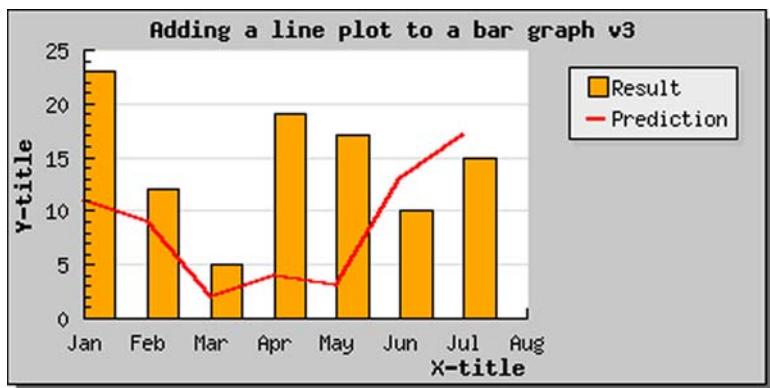


図 74: リニア スケールを使用した場合

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");
include ("../jpgraph_bar.php");

$datay = array(11,9,2,4,3,13,17);
$datay = array(23,12,5,19,17,10,15);

// Create the graph.
$graph = new Graph(400,200,"auto");
$graph->SetScale("intlin");

$graph->img->SetMargin(40,130,20,40);
$graph->SetShadow();

// Create the linear error plot
$lplot=new LinePlot($datay);
$lplot->SetColor("red");
$lplot->SetWeight(2);
$lplot->SetLegend("Prediction");

// Create the bar plot
$bplot = new BarPlot($datay);

$bplot->SetFillColor("orange");
$bplot->SetLegend("Result");

// Add the plots to the graph
$graph->Add($bplot);
$graph->Add($lplot);

$graph->title->Set("Adding a line plot to a bar graph v3");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$datax = $gDateLocale->GetShortMonth();
$graph->xaxis->SetTickLabels($datax);

// Display the graph
$graph->Stroke();
?>
```

注意:今回の例では、テキスト ラベルをリニア スケール(整数スケール)に使用するサンプルも兼ねています。

後に、散布プロットとライン プロットを組み合わせ、回帰直線を描画するサンプルを掲載します。

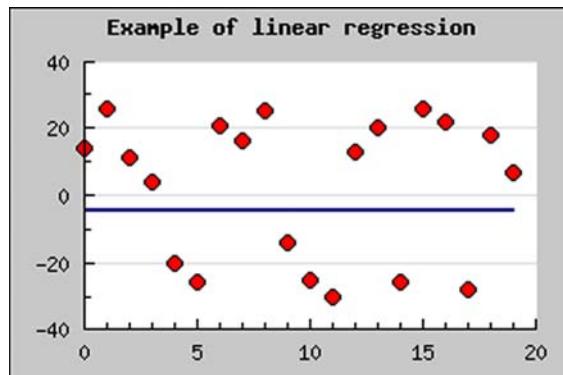


図 75: 散布プロットとライン プロットを組み合わせる

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_scatter.php");
include ("./jpgraph_line.php");

// Create some "fake" regression data
$datay = array();
$datay2 = array();
$datax = array();
$a=rand(-3,3);
$b=rand(-5,5);
for($x=0; $x<20; ++$x) {
    $datay[] = $a*$x + $b;
    $datay2[] = $a*$x + $b + rand(-30,30);
    $datax[] = $x;
}

// Create the graph
$graph = new Graph(300,200,'auto');
$graph->SetScale("linlin");

// Setup title
$graph->title->Set("Example of linear regression");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// make sure that the X-axis is always at the
// bottom at the plot and not just at Y=0 which is
// the default position
$graph->xaxis->SetPos('min');

// Create the scatter plot with some nice colors
$sp1 = new ScatterPlot($datay2,$datax);
$sp1->mark->SetType(MARK_FILLEDIRCLE);
$sp1->mark->SetFillColor("red");
$sp1->SetColor("blue");
$sp1->SetWeight(3);
$sp1->mark->SetWidth(4);

// Create the regression line
$lplot = new LinePlot($datay);
$lplot->SetWeight(2);
$lplot->SetColor('navy');

// Add the plots to the line
$graph->Add($sp1);
$graph->Add($lplot);

// ... and stroke
$graph->Stroke();

?>

```

7.8 スケールを手動で指定する

JpGraph の自動スケール機能は、ほとんどの場合において有効に働きます。しかし、手動でスケールを設定することが必要な場合もあります。たとえば、複数のグラフの比較を簡単にするために、スケールを合わせる場合などです。

手動でスケールを設定するには、[Graph::SetScale\(\)](#) メソッドに引数を追加で指定します。たとえば、Y スケールを 0 から 100 までの間で指定する場合は、以下の通りになります。

```
$graph->SetScale("textlin",0,100);
```

手動でスケールを設定した場合、チック記号の配置の仕方も指示する必要があります。チェックマークを配置するには、以下の 3 通りを選択できます。

1. 指定したスケールに基づいて、最適な目盛りが決定されます。他に何もしない場合、これがデフォルトの動作です。
2. 指定した最小値、最大値がわずかに調整されます。デフォルトの動作では最小値や最大値を基準に目盛りが設置されますが、目盛りを下端と上端に重ねたいという場合があります。視覚的な理由から、最初と最後の目盛りをスケールの端に表示したいと思うかもしれません。そういう場合は、[LinearScale::SetAutoTicks\(\)](#) メソッドを呼び出すことで、スケールの最小値と最大値の調節を行います。
3. [LinearTicks::Set\(\)](#) メソッドを呼び出すことで、目盛りを指定できます。

```
$graph->SetScale("textlin",0,100);
$graph->yscale->ticks->Set(10,5);
```

上記の設定では、目盛りを 10 ごとに表示しています。そして補助目盛りは 5 ごとに描画してあります。

これら3種類の違いを以下に掲載します。

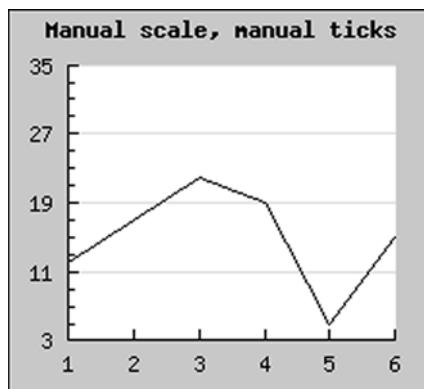


図 76:手動でスケールと目盛りを設定した場合

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(12,17,22,19,5,15);

$graph = new Graph(220,200);
$graph->SetScale("textlin",3,35);
$graph->yscale->ticks->Set(7,2);

$graph->title->Set('Manual scale, manual ticks');
$graph->title->SetFont(FF_FONT1,FS_BOLD);

$line = new LinePlot($ydata);
$graph->Add($line);

// Output graph
$graph->Stroke();

?>
```

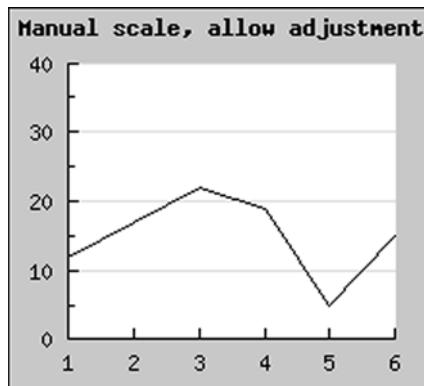


図 77:手動でスケールを設定し、自動で目盛りを調節した場合

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(12, 17, 22, 19, 5, 15);

$graph = new Graph(220, 200);
$graph->SetScale("textlin", 3, 35);
$graph->yscale->SetAutoTicks();

$graph->title->Set('Manual scale, allow adjustment');
$graph->title->SetFont(FF_FONT1, FS_BOLD);

$line = new LinePlot($ydata);
$graph->Add($line);

// Output graph
$graph->Stroke();

?>
```



図 78: 手動でスケールを設定し、自動で目盛りを調節した場合。ただし、最小値・最大値の調節を許可

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(12,17,22,19,5,15);
$graph = new Graph(220,200);
$graph->SetScale("textlin",3,35);

$graph->title->Set("Manual scale, exact limits");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

$line = new LinePlot($ydata);
$graph->Add($line);

// Output graph
$graph->Stroke();

?>

```

7.9 目盛りの間隔を自動で調節する

JpGraph では、目盛りの頻度を指定することで、目盛りの間隔を自動で設定できます。[Graph::SetTickDensity\(\)](#) メソッドを使用してください。目盛りの間隔は、以下の 4 つの手順で調節することができます。

- TICKD_VERYSPARSE — 目盛りの数は非常に少ない
- TICKD_SPARSE — 目盛りの数は少ない
- TICKD_NORMAL — 一般的な目盛り数
- TICKD_DENSE — 目盛りの数は多い

先ほどの例において、目盛りの数を増やした場合は以下の通りになります。

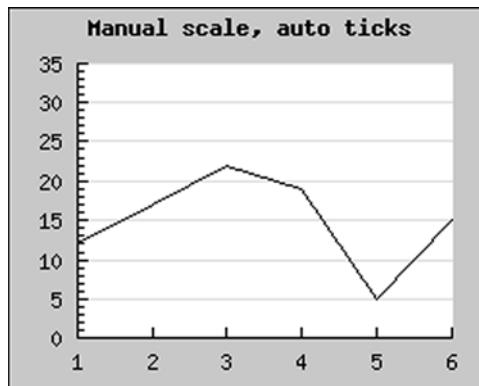


図 79: 自動チック、しかしそれほど濃厚なチックの設定での手動スケール

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(12,17,22,19,5,15);

$graph = new Graph(250,200);
$graph->SetScale("textlin",3,35);
$graph->SetTickDensity(TICKD_DENSE);
$graph->yscale->SetAutoTicks();

$graph->title->Set("Manual scale, auto ticks");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

$line = new LinePlot($ydata);
$graph->Add($line);

// Output graph
$graph->Stroke();

?>
```

7.10 日付・時間スケールを使用する

日付・時間スケールでは、独立変数(X 軸)が時間変数を指名するデータのプロットを説明します。これは、たとえば、一日の間隔で測定された温度などです。

バージョン 1.18 と前のバージョンには、日付・時間スケールを扱う 2 つの主要な方法があります。手動方法と完全に特別な “dat” スケールを使用する自動方法。ほとんどの場合、自動 “dat” スケールが好ましい選択ですが、まれな状況で、全体的なスケールの取り扱いにおいて少し多くの制限を与えるので(しかし、多くではない)、手動方法の使用がより適切な場合があります。

日付・時間スケールの使用を可能にするため、データ ポイントは X 軸(時間値)と Y 値によって表示されます。まずは、“dat” スケールの使用定義から始めます。

7.10.1 自動日付・時間スケールを使用する

X 軸の日付・時間スケールを取得する一番簡単な方法は、あらかじめ定義された “dat” スケールを使用することです。その使用を可能にするには、初め、モジュール “jpgraph_date.php” を含みスケールを指定する必要があります。たとえば、以下のコードの断片が示すように、 SetScale() を呼び出す “datlin”。

```
require_once("./jpgraph.php");
require_once("./jpgraph_line.php");
require_once("./jpgraph_date.php");
...
$graph = new Graph(540,300);
$graph->SetScale('datlin');
...
```

上記のコードはグラフに日付・線スケールを作成します。最初、日付スケールの基本的な使用を説明して、次に日付軸の正確なフォーマットの調整がどのように可能かということを例証します。

すべての自動フォーマットを使用する場合、ライブラリは日付・時間を特定するだけ短い文字列を使用します。たとえば、同じ日に全間隔である場合、時間値だけが表示され、間隔が複数日以上の場合も、日付がラベルに追加されます。

通常では、以下に例証されているように、使用されるラベルを手動にフォーマットさせることも可能です。

7.10.1.1 入力データを指定する

以下の最初の基本例は、毎 S 秒にサンプルの割り足を匹敵し、2 日間の合計周期を超えるデータセットを作成します。X 軸の入力データがタイムスタンプのデータフォームであるべきということは重要です、たとえば、システムからの秒数。PHP で、現在のタイムスタンプ値は関数 `time()` によって返されます。

```
$line = new LinePlot($data,$xdata);
```

以下の例は実践用です。

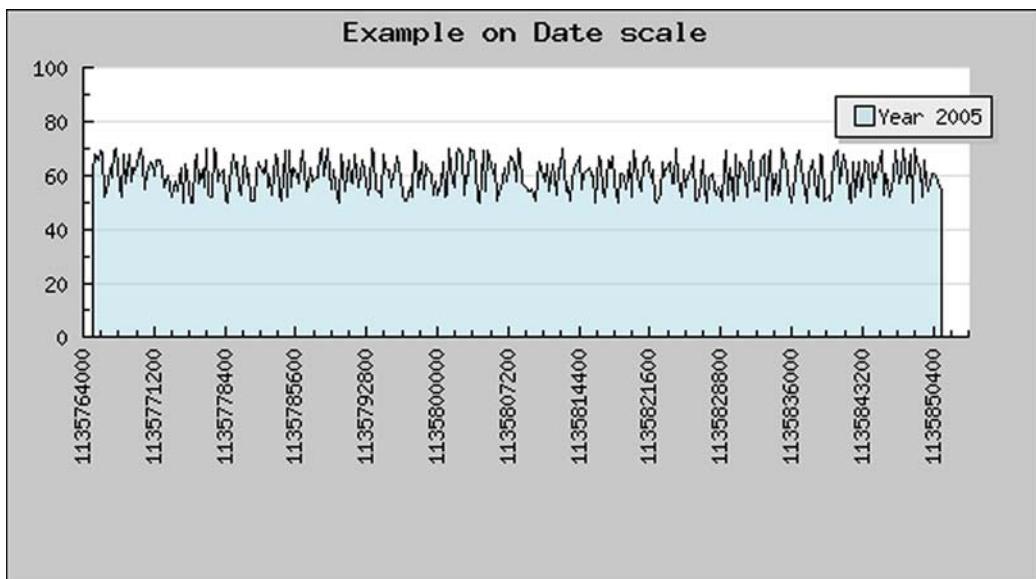


図 80:すべてがデフォルト値である特別なデータ X 軸を使用します

```
<?php
require_once("../jpgraph.php");
require_once("../jpgraph_line.php");
require_once("../jpgraph_date.php");

// Create a data set in range (50,70) and X-positions
DEFINE('NDATAPOINTS',360);
DEFINE('SAMPLERATE',240);
$start = time();
$end = $start+NDATAPOINTS*SAMPLERATE;
$data = array();
$xdata = array();
for( $i=0; $i < NDATAPOINTS; ++$i ) {
    $data[$i] = rand(50,70);
    $xdata[$i] = $start + $i * SAMPLERATE;
}

// Create the new graph
$graph = new Graph(540,300);

// Slightly larger than normal margins at the bottom to
// have room for
// the x-axis labels
$graph->SetMargin(40,40,30,130);

// Fix the Y-
// scale to go between [0,100] and use date for the x-
// axis
$graph->SetScale('datlin',0,100);
$graph->title->Set("Example on Date scale");

// Set the angle for the labels to 90 degrees
$graph->xaxis->SetLabelAngle(90);

$line = new LinePlot($data,$xdata);
$line->SetLegend('Year 2005');
$line->SetFillColor('lightblue@0.5');
$graph->Add($line);
$graph->Stroke();
?>
```

こらからの例は、上記の例をベースにしていくので、続ける前にこのグラフを作成するスクリプト(上記ソースコードにそって)を復習してください。

7.10.1.2 最初と最後の直線を調整する

上記の例のように、スケールは最初のデータ ポイントの前にわずかに起動します。それは、なぜでしょうか？これは、当然一時間の場合に“偶数”値で起動する最初の時間ラベルを作るためです。グラフの全感覚によって、起動値は常に最大意の“偶数”的時間値のために選択されます。これは、たとえば、偶数の 1 分、30 分、1 時間、1 日、1 週間などということになります。

最初(および最後)の時間線の直線は 2 つのメソッドの使用によって手動で調整することもできます。

- SetTimeAlign(\$aStartAlign,\$aEndAlign)
- SetDateAlign(\$aStartAlign,\$aEndAlign)

これらのメソッドは以下のコンスタントのうち 1 つを指定することによって作られる直線を許可します。

SetTimeAlign() では、以下のシンボルのコンスタントを使用することができます。

1. 秒単位の直線
 - MINADJ_1、1 秒の直線(これは最低のレゾルーションです)
 - MINADJ_5、5 秒に最も近い直線
 - MINADJ_10、10 秒に最も近い直線
 - MINADJ_15、15 秒に最も近い直線
 - MINADJ_30、30 秒に最も近い直線
2. 分単位の直線
 - MINADJ_1、1 分に最も近い直線
 - MINADJ_5、5 分に最も近い直線
 - MINADJ_10、10 分に最も近い直線
 - MINADJ_15、15 分に最も近い直線
 - MINADJ_30、30 分に最も近い直線
3. 時間単位の直線
 - HOURADJ_1、1 時間に最も近い直線
 - HOURADJ_2、2 時間に最も近い直線
 - HOURADJ_3、3 時間に最も近い直線
 - HOURADJ_4、4 時間に最も近い直線
 - HOURADJ_6、6 時間に最も近い直線
 - HOURADJ_12、12 時間に最も近い直線

SetDateAlign() では、以下のシンボルのコンスタントを使用することができます。

1. 日単位の直線
 - DAYADJ_1、1 日のスタートの直線
 - DAYADJ_7、1 週のスタートの直線
 - DAYADJ_WEEK、DAYADJ_7 の類義語

2. 月単位の直線
 - MONTHADJ_1、月スタートの直線
 - MONTHADJ_6、半年のスタートの直線
3. 年単位の直線
 - YEARADJ_1、1 年の直線
 - YEARADJ_2、2 年に 1 度の直線
 - YEARADJ_5、5 年の直線

いくつかの例はこれらのメソッドの使用を明確にします。

例 1. 1 時間の偶数の 1/4 のスタートを調整した時間が欲しい場合、例、偶数の 15 分間周期

```
$graph->xaxis->scale->SetTimeAlign(MINADJ_15);
```

例 2. 偶数の 2 時間で時間をスタートさせたい場合

```
$graph->xaxis->scale->SetTimeAlign(HOURADJ_2);
```

例 3. 偶数の 1 日で時間をスタートさせたい場合

```
$graph->xaxis->scale->SetDateAlign(DAYADJ_1);
```

7.10.1.3 ラベル フォーマットを調整する

デフォルトのラベル フォーマットは常にできるだけ短いユニークな文字列を使用しようとします。手動スケールを手動で設定するには、以下の例のメソッド SetDateFormat() を使用します。

```
$graph->xaxis->scale->SetDateFormat('H:i');
```

上記の例はラベルに時間(24 時間)と分を表示させます。

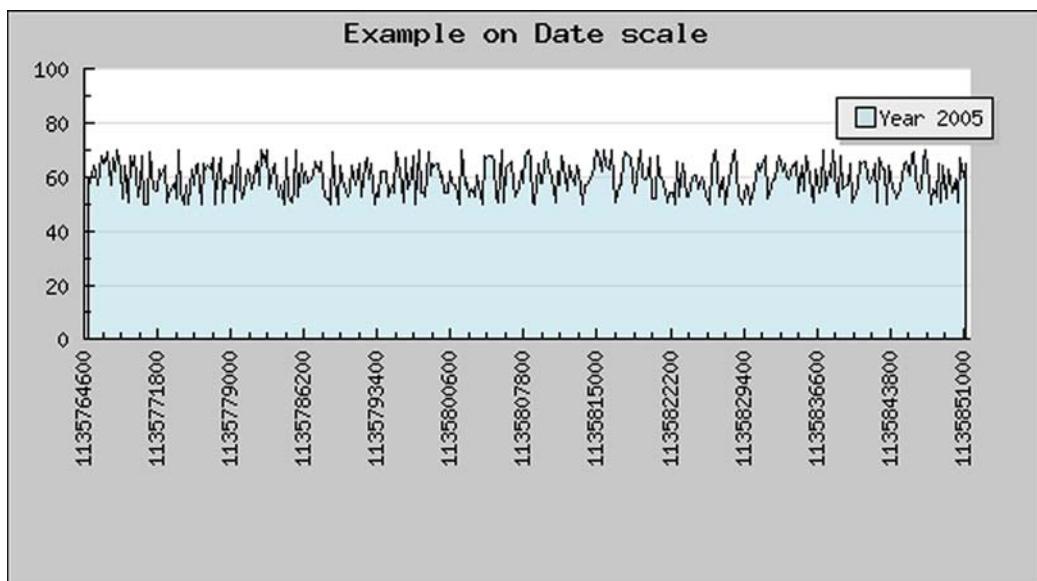


図 81:偶数の 10 踏んで最初と最後の特定のラベル フォーマットと直線の時間軸を使用します

```
<?php
require_once("../jpgraph.php");
require_once("../jpgraph_line.php");
require_once("../jpgraph_date.php");

// Create a data set in range (50,70) and X-positions
DEFINE('NDATAPOINTS',360);
DEFINE('SAMPLERATE',240);
$start = time();

$end = $start+NDATAPOINTS*SAMPLERATE;
$data = array();
$xdata = array();
for( $i=0; $i < NDATAPOINTS; ++$i ) {
    $data[$i] = rand(50,70);
    $xdata[$i] = $start + $i * SAMPLERATE;
}
```

```
// Create the new graph
$graph = new Graph(540,300);

// Slightly larger than normal margins at the bottom to
// have room for
// the x-axis labels
$graph->SetMargin(40,40,30,130);

// Fix the Y-
scale to go between [0,100] and use date for the x-
axis
$graph->SetScale('datlin',0,100);
$graph->title->Set("Example on Date scale");

// Set the angle for the labels to 90 degrees
$graph->xaxis->SetLabelAngle(90);

// The automatic format string for dates can be over-
ridden
$graph->xaxis->scale->SetDateFormat('Hi');

// Adjust the start/end to a specific alignment
$graph->xaxis->scale->SetTimeAlign(MINADJ_10);

$line = new LinePlot($data,$xdata);
$line->SetLegend('Year 2005');
$line->SetFillColor('lightblue@0.5');
$graph->Add($line);
$graph->Stroke();
?>
```

7.10.1.4 密集したデータ スケールを調整する

ライナー スケールで、どの濃度なスケール チックが必要か示すことができます。Graph::SetTickDensity() の呼び出して通常指定されます。

```
$graph->SetTickDensity(TICKD_DENSE);
```

7.10.2 手動コールバックで日付・時間スケールを指定する

以下では、PHP の time() 関数と同様の形式でタイムスタンプ値が表現されていることを前提としています。

コールバック関数を用いて、スケールに表示するラベルの書式を整形する方法を述べます。X スケールは一般的な整数スケールで指定します（タイムスタンプ値は整数値として表されています）。次に、カスタムなラベル用のコールバック関数を作成し、[SetLabelFormatCallback\(\)](#) メソッドで指定します。このコールバック関数では、タイムスタンプ値を一定の書式に変換するものです。今回の例では、PHP の Date() 関数を使用しています。

コールバック関数の例は以下のようになります。

```
// タイムスタンプを分、秒に変換するコールバック
function TimeCallback($aVal) {
    return Date('Hi:s',$aVal);
}
```

これにより、以下のようなグラフを得られます。

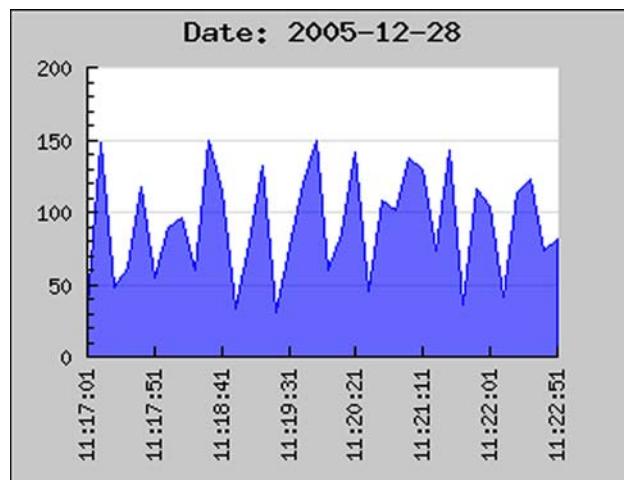


図 82: 整数スケールとコールバック関数を使用して日付の書式の整形を行う例

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

// The callback that converts timestamp to minutes and seconds
function TimeCallback($aVal) {
    return Date('H:i:'.$aVal);
}

// Fake some suitable random data
$now = time();
$datax = array($now);
for( $i=0; $i < 360; $i += 10 ) {
    $datax[] = $now + $i;
}
$n = count($datax);
$datay=array();
for( $i=0; $i < $n; ++$i ) {
    $datay[] = rand(30,150);
}

// Setup the basic graph
$graph = new Graph(324,250);
$graph->SetMargin(40,40,30,70);
$graph->title->Set('Date: '.date('Y-m-d',$now));
$graph->SetAlphaBlending();

// Setup a manual x-
// scale (We leave the sentinels for the
// Y-axis at 0 which will then autoscale the Y-axis.)
// We could also use autoscaling for the x-
// axis but then it
// probably will start a little bit earlier than the first value
// to make the first value an even number as it sees
// the timestamp
// as an normal integer value.
$graph->SetScale("intlin",0,200,$now,$datax[$n-1]);

// Setup the x-
// axis with a format callback to convert the timestamp
// to a user readable time
$graph->xaxis->SetLabelFormatCallback('TimeCallback');
$graph->xaxis->SetLabelAngle(90);

// Create the line
$p1 = new LinePlot($datay,$datax);
$p1->SetColor("blue");

// Set the fill color partly transparent
$p1->SetFillColor("blue@0.4");

// Add lineplot to the graph
$graph->Add($p1);

// Output line
$graph->Stroke();
?>

```

上記の例では、手動で X スケールを指定しています。そうすることで、X 軸の最小値と最大値がデータの最小値と最大値と合わさっています。

それぞれ表示されるラベルで SetLabelFormatCallback() が呼ばれ、関数で渡される引数はライブラリがその特定のラベルを割り当てているデフォルトのラベルです。

これは、データが、たとえば数秒で与えられたタイムスタンプを基準に表示された場合、表示された値は時間・日付の値の通常の表示方法を調整しない場合があるということです。

整数のスケールの使用では、ライブラリはデータは整数でタイムスタンプの値ではないと仮定するため、自動のスケールにあつた偶数の位置になるようにラベルのポジションを決めるので、これは作動しません。

解決のための一番良い方法は手動の特定スケールでコールバックの関数と一緒に整数の X スケールを使用することです。スケールを設定するには、少し手動の作業が必要です。データによって、1 つを表示させるため適切な時間でスケールが始まり終わることと、選択されたチックの間隔が偶数の数分、数時間、数日、または表示するための最適な時間範囲と一致することを確認する必要があります。

以下に、その例を掲載します。データを基準にした時間をサンプルにすると仮定した“偽”的データを作り、いくつかの適切なスケールとチックの間隔を設定します。このスクリプトは、タイム データの扱いをもっと促進するため基準として使用される場合があります。

```
(File: timestampex01.php)
<?php
// 人間が読めるラベルのタイムスタンプの使い方と形成の仕
// 方の例
```

```

require_once("../jpgraph.php");
require_once("../jpgraph_line.php");

// "fake" データ ポイントの番号
```

```

DEFINE('NDATAPOINTS',500);

// データ ポイントが毎 10 秒のサンプルと仮定する
DEFINE('SAMPLERATE',10);

// タイムスタンプを時間と分に変換する X スケールの
// フォーマット関数をコールバックする。
function TimeCallback($aVal) {
    return Date('H:i', $aVal);
}

// スタート時間を取得する
$start = time();
// “スタート” 時間ちょうど前に、一番近い分のスタート時間を
設定する
$adjstart = floor($start / 60);

// 範囲 (20,100) と X 位置のデータセットを作成
// ランダムをより少なくし、少ないスマーザーを作るため、
// データのシンプルな低バスを適用します
$data = array();
$xdata = array();
$data[0] = rand(20,100);
$xdata[0] = $adjstart;
for( $i=1; $i < NDATAPoints; ++$i ) {
    $data[$i] = rand(20,100)*0.2 + $data[$i-1]*0.8;
    $xdata[$i] = $adjstart + $i * SAMPLERATE;
}

```



```

// スケールのエンド値を判断する場合、
// データ ポイントが毎 10 秒でサンプル化されてデータを表
記すると仮定する。
// また、偶数のラベル チックの終わりに長さを追加する。
$adjend = $adjstart + (NDATAPoints+10)*10;

$graph = new Graph(500,250);
$graph->SetMargin(40,20,30,50);

// そして、X スケールを明確に指定するが、自動的に Y ス
ケールを測るようにする
$graph->SetScale("intlin",0,0,$adjstart,$adjend);
$graph->title->Set("Example onTimeStamp Callback");

// コールバックを設定し、ラベルの角を調整する
$graph->xaxis->SetLabelFormatCallback('TimeCallback');
$graph->xaxis->SetLabelAngle(90);

// ラベルを 5 分ごと(例、300 秒)に設定し、マイナー チック
を 1 分後とに設定する
$graph->xaxis->scale->ticks->Set(300,60);

$line = new LinePlot($data,$xdata);
$line->SetColor('lightblue');
$graph->Add($line);

$graph->Stroke();
?>

```

7.11 テキスト スケールのラベルを調整する

次に、テキスト スケールにおいてラベルを操作する方法を紹介します。主に、多数の値がある場合にテキスト スケールを扱う場合を想定します。

テキスト スケールは、X 軸の値が数値ではない場合（要するにデータの順番だけが大切な場合）に使用するためのものです。もしラベルを手動で指定しない場合、下のサンプルの通り 1 からの連番で自動でラベルが作成されます。

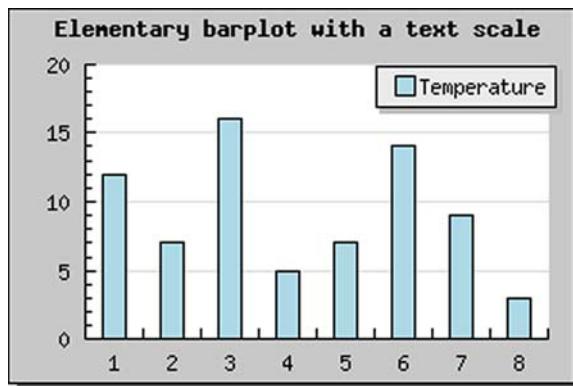


図 83: 自動のテキストスケールを用いたバー プロットの例

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

// Some data
$databary=array(12,7,16,5,7,14,9,3);

```



```

// New graph with a drop shadow
$graph = new Graph(300,200,'auto');
$graph->SetShadow();

// Use a "text" X-scale
$graph->SetScale("textlin");

```

```
// Set title and subtitle
$graph->title-
>Set("Elementary barplot with a text scale");

// Use built in font
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Create the bar plot
$b1 = new BarPlot($databary);
$b1->SetLegend("Temperature");
// $b1->SetAbsWidth(6);

// $b1->SetShadow();

// The order the plots are added determines who's on top
$graph->Add($b1);

// Finally output the image
$graph->Stroke();

?>
```

X 軸のラベルを文字列で指定するには、[Axis::SetTickLabels\(\)](#) メソッドを使用して、テキスト ラベルが格納されている配列を手渡します。指定されたラベル数よりもデータの数の方が多い場合は、デフォルトの数値も使用されます。先ほどの例に月の名前を使用した例を下記に掲載します。

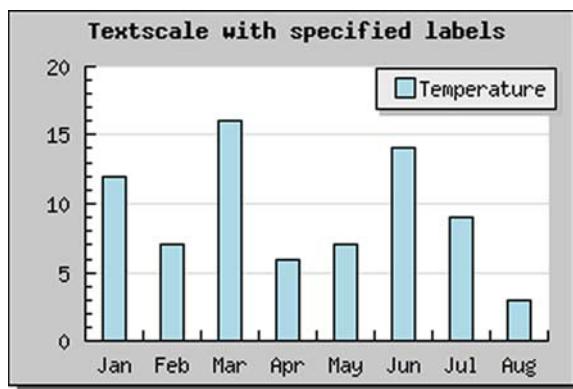


図 84: 手動でテキストスケール ラベルを指定する

```
<?php
include ("../jraph.php");
include ("../jraph_bar.php");

// Some data
$databary=array(12,7,16,6,7,14,9,3);
$months=$gDateLocale->GetShortMonth();

// New graph with a drop shadow
$graph = new Graph(300,200,'auto');
$graph->SetShadow();

// Use a "text" X-scale
$graph->SetScale("textlin");

// Specify X-labels
$graph->xaxis->SetTickLabels($months);

// Set title and subtitle
$graph->title->Set("Textscale with specified labels");

// Use built in font
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Create the bar plot
$b1 = new BarPlot($databary);
$b1->SetLegend("Temperature");

// $b1->SetAbsWidth(6);
// $b1->SetShadow();

// The order the plots are added determines who's on top
$graph->Add($b1);

// Finally output the image
$graph->Stroke();

?>
```

ヒント: 月の名前と曜日の名前を地域別に扱うため、JpGraph では [DateLocal](#) クラスを提供しています。ロケールが指定されていない場合は、システムのデフォルトのロケールが使用されます。

では、バーの数が大量にあった場合はどうなるでしょうか。25 個のバーがある場合の結果を見てみましょう。

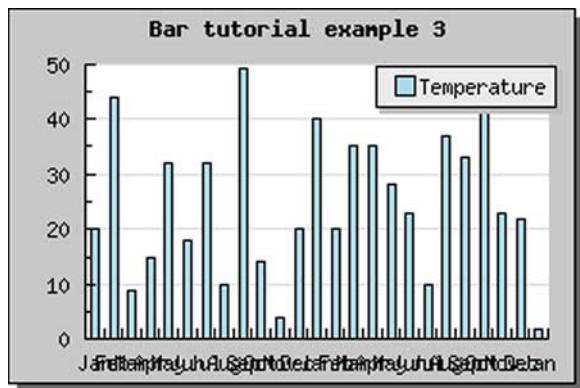


図 85:バーの数が多い場合

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

// Some data
$months=$gDateLocale->GetShortMonth();

srand ((double) microtime() * 1000000);
for( $i=0; $i<25; ++$i) {
    $databary[]=rand(1,50);
    $databarx[]=$months[$i%12];
}

// New graph with a drop shadow
$graph = new Graph(300,200,'auto');
$graph->SetShadow();

// Use a "text" X-scale
$graph->SetScale("textlin");

// Specify X-labels
$graph->xaxis->SetTickLabels($databarx);

// Set title and subtitle
$graph->title->Set("Bar tutorial example 3");

// Use built in font
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Create the bar plot
$b1 = new BarPlot($databary);
$b1->SetLegend("Temperature");
// $b1->SetAbsWidth(6);
// $b1->SetShadow();

// The order the plots are added determines who's on top
$graph->Add($b1);

// Finally output the image
$graph->Stroke();

?>

```

あまり見栄えがよくありません。ラベルの数が増えると、お互いが接近してしまいすべてのラベルを表示できなくなってしまいます。従って、すべてのラベルを表示するのは良い考えではありません。表示するラベルの数を調節するには、[SetTextLabelInterval\(\)](#) メソッドを使用します。このメソッドの引数には、表示するテキストラベルの間隔を指定します。3ヶ月ごとにラベルを表示するには、以下のようにします。

```
$graph->xaxis->SetTextLabelInterval(3)
```

これにより、以下のような外見になります。

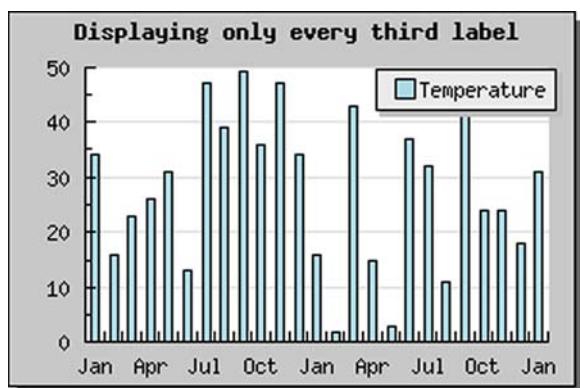


図 86:ラベルを3個ごとに表示する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

// Some data
$months=$gDateLocale->GetShortMonth();

srand ((double) microtime() * 1000000);
for( $i=0; $i<25; ++$i) {
    $databary[]=rand(1,50);
    $databarx[]=$months[$i%12];
}

// New graph with a drop shadow
$graph = new Graph(300,200,'auto');
$graph->SetShadow();

// Use a "text" X-scale
$graph->SetScale("textlin");

// Specify X-labels
// $databarx = array('tXi','','xxx','','iXii','',''000','','t00');
$graph->xaxis->SetFont(FF_FONT1,FS_NORMAL);

```

```

$graph->xaxis->SetTickLabels($databarx);
$graph->xaxis->SetTextLabelInterval(3);

// Set title and subtitle
$graph->title->Set("Displaying only every third label");

// Use built in font
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Create the bar plot
$b1 = new BarPlot($databary);
$b1->SetLegend("Temperature");
// $b1->SetAbsWidth(6);
// $b1->SetShadow();

// The order the plots are added determines who's on top
$graph->Add($b1);

// Finally output the image
$graph->Stroke();

?>

```

これにより、かなり外見が改善されました。

さらにデータの数が多くなった場合は、すべての目盛りを表示するのも難しくなってきます。そのため、JpGraph では目盛りを表示する間隔も変更できます（[SetTextTickInterval\(\)](#) 関数を使用します）。しかし、テキスト スケールを使用しているグラフの場合、目盛りはバーの間に表示されるため、下記のグラフのようにあまり見栄えはよくありません。下記の例のように不可能というわけではありませんが、それほど見栄えがよいわけではありません。

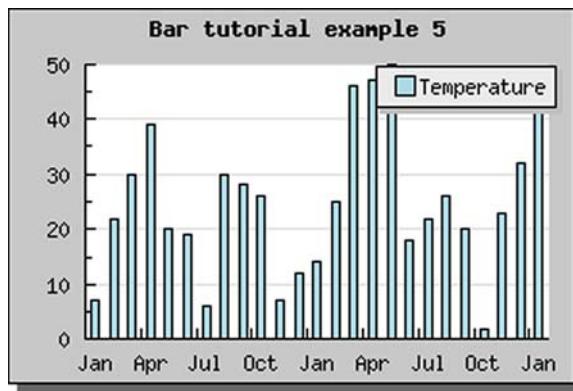


図 87: 目盛りを 3 個ごとに表示する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

// Some data
$months=$gDateLocale->GetShortMonth();

srand ((double) microtime() * 1000000);
for( $i=0; $i<25; ++$i) {
    $databary[]=rand(1,50);
    $databarx[]=$months[$i%12];
}

// New graph with a drop shadow

```

```

$graph = new Graph(300,200,'auto');
$graph->SetShadow();

// Use a "text" X-scale
$graph->SetScale("textlin");

// Specify X-labels
$graph->xaxis->SetTickLabels($databarx);
$graph->xaxis->SetTextLabelInterval(1);
$graph->xaxis->SetTextTickInterval(3);

// Set title and subtitle
$graph->title->Set("Bar tutorial example 5");

```

直角のXYプロットを使用する

```
// Use built in font
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Create the bar plot
$b1 = new BarPlot($databary);
$b1->SetLegend("Temperature");
$b1->SetWidth(0.4);

// The order the plots are added determines who's on top
$graph->Add($b1);

// Finally output the image
$graph->Stroke();

?>
```

目盛りを非表示にすることもできます。目盛りを隠すには、[Axis::HideTicks\(\)](#) メソッドを使用します。X 軸にある目盛りを隠した場合は、以下のような結果が得られます。

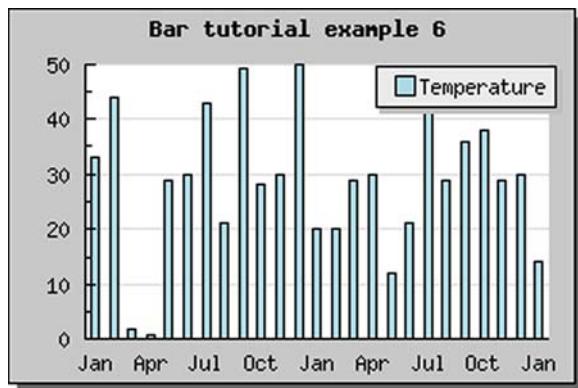


図 88:すべての目盛りを非表示にする

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

// Some data
$months=$gDateLocale->GetShortMonth();
 srand ((double) microtime() * 1000000);
for( $i=0; $i<25; ++$i) {
    $databary[]=rand(1,50);
    $databarx[]=$months[$i%12];
}

// New graph with a drop shadow
$graph = new Graph(300,200,'auto');
$graph->SetShadow();

// Use a "text" X-scale
$graph->SetScale("textlin");

// Specify X-labels
$graph->xaxis->SetTickLabels($databarx);
$graph->xaxis->SetTextLabelInterval(3);

// Hide the tick marks
$graph->xaxis->HideTicks();

// Set title and subtitle
$graph->title->Set("Bar tutorial example 6");

// Use built in font
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Create the bar plot
$b1 = new BarPlot($databary);
$b1->SetLegend("Temperature");
$b1->SetWidth(0.4);

// The order the plots are added determines who's on top
$graph->Add($b1);

// Finally output the image
$graph->Stroke();

?>
```

7.12 グラフにテキスト文字列を追加する

JpGraph では、グラフの任意の位置にテキスト文字列を追加できます。テキストは複数行に渡ることが可能で、段落揃えも指定できます。

テキストを追加するには、[Text\(\)](#) オブジェクトのインスタンスを作成し、[AddText\(\)](#) メソッドを使ってテキストオブジェクトをグラフに追加します。

テキスト ボックスの位置はグラフの横と縦に対する割合で指定します。また、テキスト ボックスを配置する時には、テキスト ボックスのどの部分が指定した位置に対応しているのかを指定する必要があります。

デフォルトでは、テキスト ボックスの左上部が使用されます。

テキストの位置を変更する方法を説明するために、とてもシンプルなバー プロットを使用します。まずは、ほとんどの値をデフォルトのままにして、1 行のテキストをグラフに追加してみましょう。

```
$txt=new Text("This is a text");
$txt->Pos(0,0);
$txt->SetColor("red");
$graph->AddText($txt);
```

この結果を以下に表します。

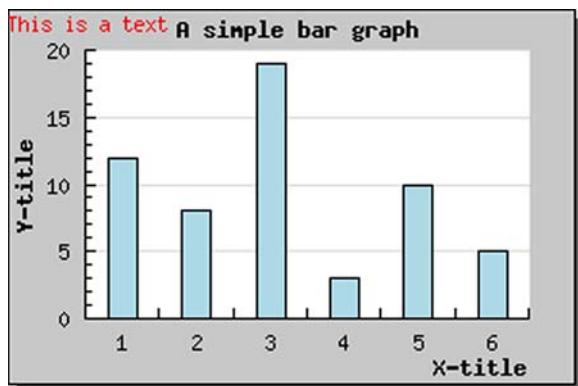


図 89:テキスト文字列を左上部に追加する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar plot
$bplot = new BarPlot($datay);
$graph->Add($bplot);

// Create and add a new text
$txt=new Text("This is a text");
$txt->Pos(0,0);
$txt->SetColor("red");
$graph->AddText($txt);

// Setup the titles
$graph->title->Set("A simple bar graph");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>
```

簡単なことです。次に、背景色を加えてフォントを大きくしたテキスト ボックスを作成しましょう。また、テキスト ボックスの周囲にはフレームを設置し、ドロップシャドウを追加してみましょう。テキスト ボックスの調整には、[SetBox\(\)](#) メソッドと [SetBox\(\)](#) メソッドを使用します。

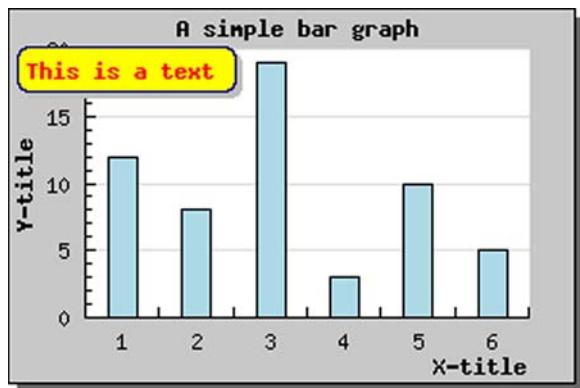


図 90:テキストをより効果的にする

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar plot
$bplot = new BarPlot($datay);
$graph->Add($bplot);

// Create and add a new text
$txt=new Text("This is a text");
$txt->Pos(10,25);
$txt->SetFont(FF_FONT1,FS_BOLD);
$txt->SetBox('yellow','navy','gray');
$txt->SetColor("red");
$graph->AddText($txt);

// Setup the titles
$graph->title->Set("A simple bar graph");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>

```

もっと良くなりました。ここで、注目します。複数行のテキストを追加する場合は、各行の後に改行文字（¥n）を付け加えます。デフォルトでは段落は左揃えですが、右揃えや中央揃えを指定することもできます。

今回は、先ほどのテキストに数行を追加して、テキストボックスをグラフの中央に配置してみましょう。同時に、段落揃えを中央揃えに設定します。段落揃えを設定するには、[Text::ParagraphAlign\(\)](#)メソッドを使用します。

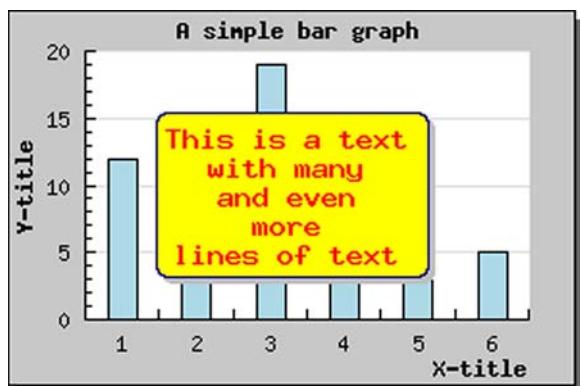


図 91:複数の行があり、段落が中央揃えになっている場合

```

<?php
include ("../jgraph.php");
include ("../jgraph_bar.php");

$datay=array(12,8,19,3,10,5);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Add a drop shadow
$graph->SetShadow();

// Adjust the margin a bit to make more room for titles
$graph->img->SetMargin(40,30,20,40);

// Create a bar plot
$bplot = new BarPlot($datay);
$graph->Add($bplot);

// Create and add a new text

```

```

$txt=new Text("This is a text\nwith many\nand even\nmore\nlines of\ntext");
$txt->Pos(0.5,0.5,"center","center");
$txt->SetFont(FF_FONT2,FS_BOLD);
$txt->ParagraphAlign('centered');
$txt->SetBox('yellow','navy','gray');
$txt->SetColor("red");
$graph->AddText($txt);

// Setup the titles
$graph->title->Set("A simple bar graph");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

// Display the graph
$graph->Stroke();
?>

```

もちろん、グラフに追加できるテキスト文字列に制限はありません。

JpGraph のバージョン 1.12 からは、スケール座標を用いてテキスト文字列を追加できるようになりました。これは、[Text::SetScalePos\(\)](#) メソッドを使います。

7.13 グラフにタイトルとフッターを追加する

各グラフは 3 種類のタイトルを持つことができます。それぞれ以下の通りです。

1. タイトル (title オブジェクト)
2. サブタイトル (subtitle オブジェクト)
3. サブサブタイトル (subsubtitle オブジェクト)

これらの 3 種類のプロパティは標準的なテキスト オブジェクトで、異なったフォントやカラー、サイズなどを設定できます。

また、グラフの周りに余白を設定したい場合は[Graph::SetMargin\(\)](#) メソッドを使用します。

これらのタイトルは自動で配置され、使用されているフォント サイズに調節されます。

もし、タイトルの周りに余白を設定したい場合は、[Text::SetMargin\(\)](#) メソッドを使用できます。たとえば、以下のようにした場合

```
$graph->title->SetMargin(20);
```

タイトル文字列の上端と グラフの上端の間には 20 ピクセル分の余白ができます。同様に、サブタイトルに対して SetMargin() メソッドを呼び出した場合は、タイトルの下端とサブタイトルの上端の間の余白が変更されます。

タイトルはグラフの上部に配置され、中央揃えで表示されます。各タイトルは改行文字（¥n）を用いて複数行指定できます。デフォルトでは各タイトルは中央揃えになっていますが、ParagraphAlign()メソッドにより変更が可能です。

各グラフではフッターも使用できます。実際には、フッターには 3 種類のフッターがあり、left、center、right フッターが提供されています。left フッターは左揃えのもので、center フッターは中央揃え、right フッターは右揃えです。

これらのフッターは標準的なテキストオブジェクトなので、任意にカラーやフォント、サイズなどを変更できます。

フッターには、以下の通り Graph::footer プロパティを使用してアクセスできます。

```
$graph->footer->left->Set("(C) 2002 KXY");
$graph->footer->center->Set("CONFIDENTIAL");
$graph->footer->center->SetColor("red");
```

```
$graph->footer->center->SetFont(FF_FONT2,FS_BOLD);
$graph->footer->right->Set("19 Aug 2002");
```

注意: ブランドタイミング引数を使用する場合は、left フッターを空にします。

7.14 タブ形式のタイトルを追加する

タイトルの種類の 1 つにタブ タイトルがあります。これは、グラフに名前を表示させるもう一つの方法です。タブ タイトルを使用すると、プロットエリアの上部にタブが表示されます。

タブ タイトルを使用するには、グラフの tabtitle プロパティにアクセスします。

以下に、タブ タイトルの効果を掲載します。

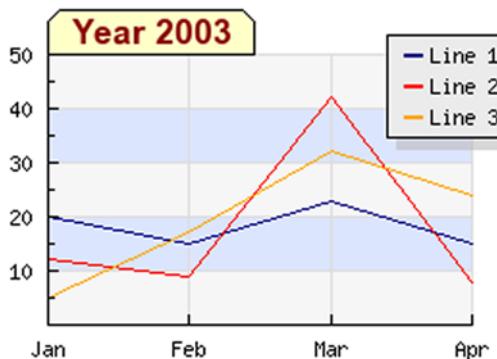


図 92:タブ タイトル形式を使用する

```
<?php
include ("../jgraph.php");
include ("../jgraph_line.php");

$datay1 = array(20,15,23,15);
$datay2 = array(12,9,42,8);
$datay3 = array(5,17,32,24);

// Setup the graph
$graph = new Graph(300,200);
$graph->SetMarginColor('white');
$graph->SetScale("textin");
$graph->SetFrame(false);
```

```
$graph->SetMargin(30,50,30,30);

$graph->tabtitle->Set(' Year 2003 ');
$graph->tabtitle->SetFont(FF_ARIAL,FS_BOLD,13);

$graph->yaxis->HideZeroLabel();
$graph->ygrid-
>SetFill(true,'#EFEFEF@0.5','#BBCCFF@0.5');
$graph->xgrid->Show();

$graph->xaxis->SetTickLabels($gDateLocale-
>GetShortMonth());
```

```
// Create the first line
$p1 = new LinePlot($datay1);
$p1->SetColor("navy");
$p1->SetLegend('Line 1');
$graph->Add($p1);

// Create the second line
$p2 = new LinePlot($datay2);
$p2->SetColor("red");
$p2->SetLegend('Line 2');
$graph->Add($p2);
```

```
// Create the third line
$p3 = new LinePlot($datay3);
$p3->SetColor("orange");
$p3->SetLegend('Line 3');
$graph->Add($p3);

$graph->legend->SetShadow('gray@0.4',5);
$graph->legend->SetPos(0.1,0.1,'right','top');
// Output line
$graph->Stroke();

?>
```

他のタイトルと同様、タブ タイトルのフォントやカラーは自由に設定できます。詳細は、[GraphTabTitle\(\)](#) クラス リファレンスを参照してください。

7.15 背景イメージを使用する

単色の背景カラーではなく、任意のイメージ画像を背景に指定できます。指定できる形式は PNG、JPEG、あるいは GIF 形式で、インストール状況によって異なります。

GD に関する注意事項: GD2.x を使う場合は、USE_TRUECOLOR オプションを `true` に設定する必要があります。これはデフォルトの動作です。この設定を行わないと、画像が黒一色で塗りつぶされてしまいます。

特定のイメージを背景に使用する場合は、[Graph::SetBackgroundImage\(\)](#) メソッドを使用します。引数では、ファイル名とグラフへの配置方法、画像の形式 (JPG、PNG、GIF 形式の場合) を指定します。画像の形式が `auto` として指定された場合、ファイルの拡張子から適切な画像形式が判断されます。

当然、ファイル名は明白ですが、2 つ目の引数はそうでない場合があります。2 番目の引数は、ファイルにあるイメージをグラフにコピーする方法を指定しています。以下の 4 種類の値を指定できます。

1. `BGIMG_COPY` — イメージを変更することなくグラフにコピーします。イメージはグラフの左上隅に配置されます。
2. `BGIMG_CENTER` — イメージを変更することなくグラフにコピーします。イメージはグラフの中央に配置されます。
3. `BGIMG_FILLFRAME` — イメージを伸縮して、グラフの大きさに合わせます。
4. `BGIMG_FILLPLOT` — イメージを伸縮して、グラフのプロット エリアの大きさに合わせます。

以下では、画像に基づいたパレットのみを取り扱います。この方法では **True color** の画像は取り扱えません。**True Color** の画像を使った場合には、効果が表示されません。

背景イメージを、透かしとして使用したい方もいるでしょう。透かしに利用するには、元々のイメージを取得して、そのコントラストや明るさを下げる必要があります。最終的に背景画像として使用する修正した画像を保存します。

JpGraph では、[Graph::AdjBackgroundImage\(\)](#) メソッドを使うことで自動的に透かしを作成できます。このメソッドは、画像の彩度と明るさ、コントラストを調節するものです。

```
$graph->AdjBackgroundImage(...)
```

グラフに侵入しすぎるイメージを回避する“ウォーターカラー”効果の遂行。

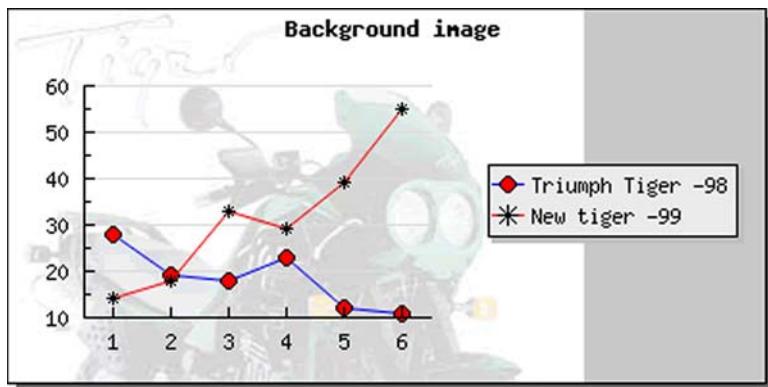


図 93:透かしとして背景イメージを使用する例

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

// Some data
$datay = array(28,19,18,23,12,11);
$data2y = array(14,18,33,29,39,55);

// A nice graph with anti-aliasing
$graph = new Graph(400,200,"auto");
$graph->img->SetMargin(40,180,40,40);
$graph-
>SetBackgroundImage("tiger_bkg.png",BGIMG_COPY);

$graph->img->SetAntiAliasing("white");
$graph->SetScale("textlin");
$graph->SetShadow();
$graph->title->Set("Background image");

// Use built in font
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Slightly adjust the legend from it's default position in the
// top right corner.
$graph->legend->Pos(0.05,0.5,"right","center");

// Create the first line
$p1 = new LinePlot($datay);
$p1->mark->SetType(MARK_FILLEDCIRCLE);
$p1->mark->SetFillColor("red");
$p1->mark->SetWidth(4);
$p1->SetColor("blue");
$p1->SetCenter();
$p1->SetLegend("Triumph Tiger -98");
$graph->Add($p1);

// ... and the second
$p2 = new LinePlot($data2y);
$p2->mark->SetType(MARK_STAR);
$p2->mark->SetFillColor("red");
$p2->mark->SetWidth(4);
$p2->SetColor("red");
$p2->SetCenter();
$p2->SetLegend("New tiger -99");
$graph->Add($p2);

// Output line
$graph->Stroke();

?>
```

注意: 背景イメージは 1998 年のトライアンフ タイガーです。このバイクは 900 cc の公道以外を走るためのバイクです。

7.16 背景グラデーション カラーを使用する

背景イメージだけでなく、背景にグラデーション カラーも使用できます。グラデーション カラーは、グラフ全体、プロット エリアのみ、あるいは余白部分のみ、の 3 通りから選択できます。こうすることで、背景カラーとイメージの両方を組み合わせることができます。たとえば、プロットエリアで背景画像を使用し、その余白でグラデーションカラーを使用するということもできます。

背景にグラデーション カラーを使用するには、[Graph::SetBackgroundGradient\(\)](#) メソッドを使用します。メソッドを使用します。詳細はクラス リファレンスを参照してください。これにより得られる効果を下記に掲載します。



図 94:背景にグラデーション カラーを使用する例

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$datay1 = array(4,26,12,18,8,22);
$datay2 = array(12,9,42,8,20,19);

// Setup the graph
$graph = new Graph(300,200);
$graph->SetMarginColor('white');
$graph->SetScale("textin",0.50);
$graph->SetMargin(30,50,30,30);

// We must have the frame enabled to get the gradient
// However, we don't want the frame line so we set it to
// white color which makes it invisible.
$graph->SetFrame(true,'white');

// Setup a background gradient image
$graph-
>SetBackgroundGradient('blue','navy:0.5',GRAD_HOR,BGRA_D_PLOT);

// Setup the tab title
$graph->tabtitle->Set(' 3rd Division ' );
$graph->tabtitle->SetFont(FF_ARIAL,FS_BOLD,13);

// Setup x,Y grid
$graph->xgrid->Show();
$graph->xgrid->SetColor('gray@0.5');
$graph->xaxis->SetTickLabels($gDateLocale-
>GetShortMonth());
$graph->ygrid->SetColor('gray@0.5');

// Setup color for axis and labels on axis
$graph->xaxis->SetColor('orange','black');
$graph->yaxis->SetColor('orange','black');

// Ticks on the outside
$graph->xaxis->SetTickSide(SIDE_DOWN);
$graph->yaxis->SetTickSide(SIDE_LEFT);

// Setup the legend box colors and font
$graph->legend->SetColor('white','navy');
$graph->legend->SetFillColor('navy@0.25');
$graph->legend->SetFont(FF_ARIAL,FS_BOLD,8);
$graph->legend->SetShadow('darkgray@0.4',3);
$graph->legend->SetPos(0.05,0.05,'right','top');

// Create the first line
$p1 = new LinePlot($datay1);
$p1->SetColor("red");
$p1->SetWeight(2);
$p1->SetLegend('2002');
$graph->Add($p1);

// Create the second line
$p2 = new LinePlot($datay2);
$p2->SetColor("lightyellow");
$p2->SetLegend('2001');
$p2->SetWeight(2);
$graph->Add($p2);

// Output line
```

```
$graph->Stroke();
```

?>

また、/utils/misc/ ディレクトリに mkgrad.php というスクリプトがあります。このスクリプトを実行すると、グラデーション カラーを作成できます。

下の例では、より面白いプロットエリアを作成するためにこのユーティリティを使用しました。



図 95:グラデーション カラーを使用した例

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

// Callback to negate the argument
function _cb_negate($aVal) {
    return round(-$aVal);
}

// A fake depth curve
$ydata = array(0,1,4,5,8,9,10,14,16,16,16,18,20,20,20,22,25,22,19,19,15,15,15,10,10,10,6,5,5,4,2,1,0);

// Negate all data
$n = count($ydata);
for($i=0; $i<$n; ++$i) {
    $ydata[$i] = round(-$ydata[$i]);
}

// Basic graph setup
$graph = new Graph(400,300,"auto");
$graph->SetScale("linlin");
$graph->img->SetMargin(50,50,60,40);
$graph->SetMarginColor('darkblue');
$graph->SetColor('darkblue');
$graph->SetAxisStyle(AXSTYLE_BOXOUT);
$graph-
>SetBackgroundImage("blueblack400x300grad.png",1);

// $graph-
>SetBackgroundImage("lightbluedarkblue400x300grad.png
",1);

$graph->title->Set("Depth curve. Dive #2");
$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->title->SetColor("white");

$graph->subtitle->Set("(Negated Y-axis)");
$graph->subtitle->SetFont(FF_FONT1,FS_NORMAL);
$graph->subtitle->SetColor("white");

// Setup axis
$graph->yaxis->SetLabelFormatCallback("_cb_negate");
$graph->xaxis->SetColor("lightblue","white");
$graph->yaxis->SetColor("lightblue","white");
$graph->ygrid->SetColor("blue");

$lp1 = new LinePlot($ydata);
$lp1->SetColor("yellow");
$lp1->SetWeight(2);

$graph->Add($lp1);
$graph->Stroke();
?>
```

7.17 プロット マークにコールバック関数を使用する

JpGraph では、プロット マークの描画にコールバック関数を用いることができます。そうすることで、各プロット マークの大きさとカラーを制御できます。

コールバック関数では、現在の Y 座標を引数として受け取ります。コールバック関数は、以下の 3 種類の値を格納した配列を戻り値として返却する必要があります。戻り値は以下の値でなければなりません。

1. プロット マークの大きさ
2. プロット マークのカラー
3. プロット マークの塗りつぶしカラー

これらの意味は、使用するプロット マークによって異なります。

コールバック関数はグローバル関数で、[PlotMark::SetCallback\(\)](#) メソッドを通じて使用します。

たとえば、Y 値が 90 より上の場合のプロット マークにのみカラーを上塗りしたい場合は、以下のようないコードになります。

```
function MarkCallback($aVal) {
    if( $aVal > 90)
        $fcolor="red"
    else
        $fcolor="";
    return array("", "", $fcolor);
}
...
$plot->mark->SetCallback("MarkCallback");
...
```

上記のサンプルでは、値を返却しているものもあります。こうすることで、それらに関しては変更が反映されず、プロット マークのデフォルト設定が使用されます。

また、プロット マークの大きさを Y 値に対応したものにすることで、バルーン プロット形式のプロットを作成できます。以下にその例を掲載します。format コールバックは、それぞれのプロットの Y 値に応じて色とサイズを変更するために使用されます。

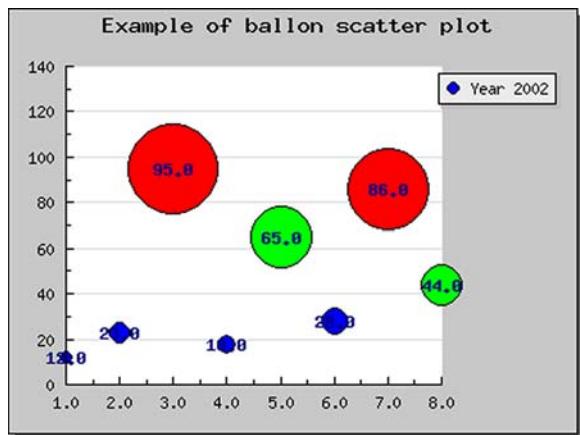


図 96:コールバック関数を使ってバルーンプロットを作成する

```
<?php
// $Id: balloonex1.php,v 1.5 2002/12/15 16:08:51 adit
us Exp $
include ("./jpgraph.php");
include ("./jpgraph_scatter.php");

// Some data
$datax = array(1,2,3,4,5,6,7,8);
$datay = array(12,23,95,18,65,28,86,44);
// Callback for markers
// Must return array(width,color,fill_color)
// If any of the returned values are "" then the
// default value for that parameter will be used.
function FCallback($aVal) {
    // This callback will adjust the fill color and size o
f
    // the datapoint according to the data value accord
ing to
    if( $aVal < 30 ) $c = "blue";
    elseif( $aVal < 70 ) $c = "green";
    else $c="red";
    return array(floor($aVal/3),"",$c);
}

// Setup a basic graph
$graph = new Graph(400,300,'auto');
$graph->SetScale("linlin");
$graph->img->SetMargin(40,100,40,40);
$graph->SetShadow();
```

```
$graph->title->Set("Example of balloon scatter plot");
// Use a lot of grace to get large scales
$graph->yaxis->scale->SetGrace(50,10);

// Make sure X-axis as at the bottom of the graph
$graph->xaxis->SetPos('min');

// Create the scatter plot
$sp1 = new ScatterPlot($datay,$datax);
$sp1->mark->SetType(MARK_FILLEDIRCLE);

// Uncomment the following two lines to display the va
lues
$sp1->value->Show();
$sp1->value->SetFont(FF_FONT1,FS_BOLD);

// Specify the callback
$sp1->mark->SetCallback("FCallback");

// Setup the legend for plot
$sp1->SetLegend('Year 2002');

// Add the scatter plot to the graph
$graph->Add($sp1);

// ... and send to browser
$graph->Stroke();

?>
```

7.18 グラフを 90 度回転する

セクション [10.2](#) で、任意の角度でグラフを回転させる方法を説明していますが、実用的には 90 度の回転がよく用いられます。この機能を使うことで、水平バー プロットを作成することができます。

グラフを回転させた場合、余白も回転が行われます。また、デフォルトでは回転の中心はプロット エリアの中心で、画像全体の中心ではありません。そのため、設定した余白により、回転の中心が移動してしまいます。回転の中心を手動で設定する方法は、セクション [10.2](#) に記述されています。

[Graph::SetMargin\(\)](#) メソッドを使用して明確な余白を設定する必要があります。

しかし、グラフを 90 度回転して、適切な余白を設定する場合は、[Graph::Set90AndMargin\(\)](#) メソッドを使用できます。このメソッドを使うことで、回転した後のグラフの余白を手動で調整する必要がなくなります。

このメソッドを呼び出すだけで、グラフが 90 度回転されます。

7.19 軸をフォーマットする

JpGraph では軸の自由度が高く、数多くの設定ができるようになっています。カラーやサイズ、位置、外見などを変更できます。

7.19.1 一般的な 2 軸グラフ

X 軸と Y 軸が 1 つずつの一般的な 2 軸グラフの例から始めましょう。各軸の位置は、[Axis::SetPos\(\\$aPosition\)](#) メソッドを呼び出すことで指定できます。軸を指定する際には、別軸での位置を指定する必要があることに注意してください。デフォルトでは、軸は別軸の 0 点に配置されています。そのため、軸は 0,0 点で交わります。

また、軸の指定には min および max という特殊な 2 点を指定することもできます。これらを指定すると、軸はもう一方の軸の最小値、あるいは最大値に配置されます。

たとえば、常に X 軸をグラフの最下部に表示したい場合は、以下のようにします。

```
$graph->xaxis->SetPos("min");
```

軸のカラーと線の幅を指定するには、[Axis::SetColor\(\)](#) メソッドと [Axis::SetWeight\(\)](#) メソッドを使用します。

ヒント: JpGraph では、ラベルが描画されたまま軸のみを非表示にすることを直接サポートしていませんが、この効果は軸のカラーを背景色と同じにすることで簡単に実現できます。詳細は Example ディレクトリにある、barintex2.php を参照してください。完全に軸を非表示にするには、[Hide\(\)](#) メソッドを使用します。

また、軸にタイトルを設定することも可能です。これは、[Axis::SetTitle\(\)](#) メソッドを使用します。これは実際はタイトルプロパティに直接接続するための、ただのショートカットです。また、[Axis::title::Set\(\)](#) メソッドを使用すると、行削除を行えます。

デフォルトでは、X 軸のタイトルは右揃えで、Y 軸のタイトルは中央揃え（90 度回転します）に設定されています。

また、タイトルの位置は [Axis::SetTitle\(\)](#) メソッドの 2 番目の引数で調節できます。引数には、high、middle、そして low のいずれかを指定できます。

また、軸とタイトルの間の余白を調節することもできます。特にラベルが長い場合は、Y 軸のタイトルの余白を増やす必要があります。軸とタイトルの距離は、[Axis::SetTitleMargin\(\)](#) メソッドでピクセル単位で指定できます。

例として、Y 軸の余白を増やす場合は、以下のように記述します。

```
$graph->yaxis->SetTitleMargin(40);
```

次に、軸の上に目盛りやラベルを配置する場合の説明を行います。軸、目盛り、およびラベルをどの位置に配置するかを設定できます。X 軸は、グラフの上部（プロットエリアの内側）、あるいはグラフの下部（プロットエリアの外側）のいずれかに配置できます。同様に、Y 軸もグラフの左側、あるいは右側に配置できます。

ラベルの配置を修正するには、[Axis::SetTitleSide\(\)](#) メソッドを使用します。また、目盛りの位置を修正するには、[SetTickSide\(\)](#) メソッドを使用します。

注意: メソッドには、以前のバージョンとの互換性のため、同様の SetTickDirection() メソッドがありますが、現在ではこのメソッドの使用は推奨されておりません。

これらのメソッドには、以下の引数を指定できます。

- SIDE_UP
- SIDE_DOWN
- SIDE_LEFT
- SIDE_RIGHT

たとえば、下記の通りに記述することで、X 軸のラベルと目盛りの場所を変更できます。

```
$graph->xaxis->SetLabelPos(SIDE_UP);
$graph->xaxis->SetTickSide(SIDE_DOWN);
```

この方法を用いると、以下のようにグラフの上部に X 軸を配置できます。

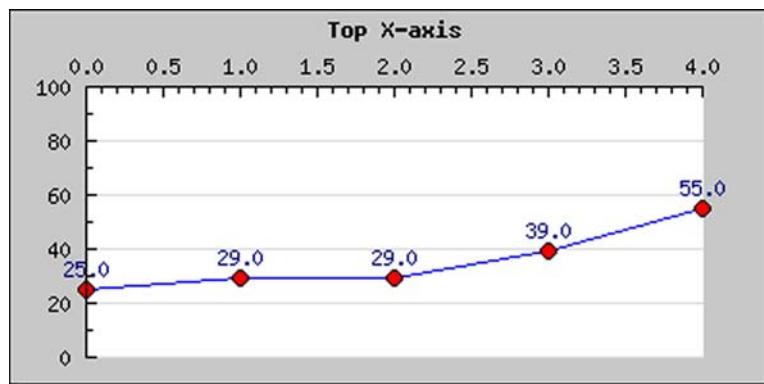


図 97:X 軸の位置を調節する例

```
<?php
include ("../jpgraph.php");
```

```
include ("../jpgraph_line.php");
```

```
// Some data
$datay = array(25,29,29,39,55);

$graph = new Graph(400,200,'auto');
$graph->img->SetMargin(40,40,40,20);

$graph->SetScale("linlin");
$graph->SetShadow();
$graph->title->Set("Top X-axis");

// Start at 0
$graph->yscale->SetAutoMin(0);

// Add some air around the Y-scale
$graph->yscale->SetGrace(100);

// Use built in font
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Adjust the X-axis
$graph->xaxis->SetPos("max");
$graph->xaxis->SetLabelSide(SIDE_UP);

$graph->xaxis->SetTickSide(SIDE_DOWN);

// Create the line plot
$p1 = new LinePlot($datay);
$p1->SetColor("blue");

// Specify marks for the line plots
$p1->mark->SetType(MARK_FILLEDIRCLE);
$p1->mark->SetFillColor("red");
$p1->mark->SetWidth(4);

// Show values
$p1->value->Show();

// Add lineplot to graph
$graph->Add($p1);

// Output line
$graph->Stroke();

?>
```

7.19.2 関数形式の軸

関数形式のプロットではそれぞれの軸を複写することがよくあるので、グラフのすべての端にはラベル形式の軸が設定されています。JpGraph では関数形式のプロットをサポートしています。

関数形式のプロットの例は下記を参照してください。

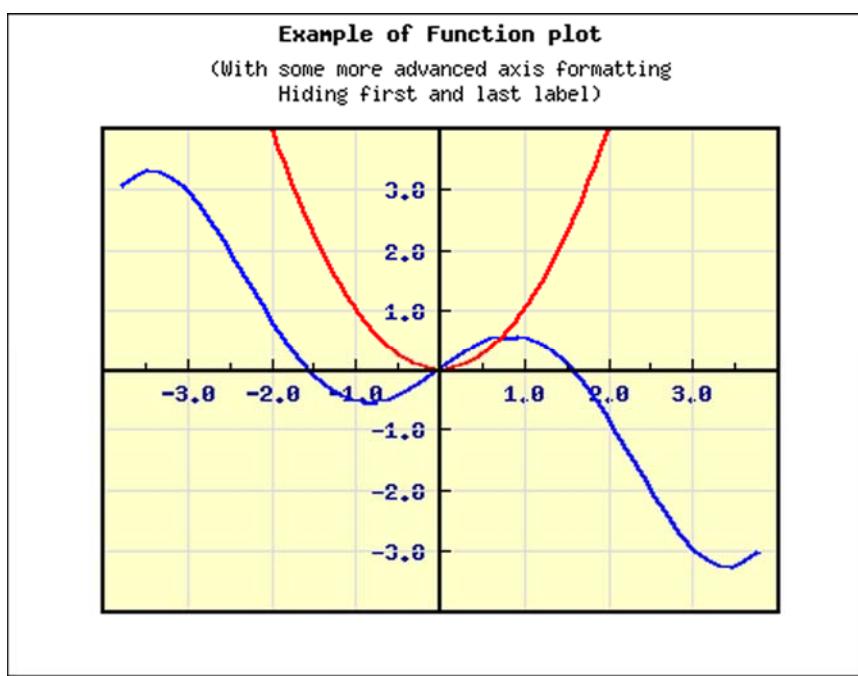


図 98:関数形式の軸

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");
include ("./jpgraph_utils.inc");

$f = new FuncGenerator('cos($x)*$x');
list($xdata,$ydata) = $f->E(-1.2*M_PI,1.2*M_PI);

$f = new FuncGenerator('($x*$x)');
list($x2data,$y2data) = $f->E(-2,2);

// Setup the basic graph
$graph = new Graph(450,350,"auto");
$graph->SetScale("linlin");
$graph->SetShadow();
$graph->img->SetMargin(50,50,60,40);
```

```
$graph->SetBox(true,'black',2);
$graph->SetMarginColor('white');
$graph->SetColor('lightyellow');

// ... and titles
$graph->title->Set('Example of Function plot');
$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->subtitle-
>Set("(With some more advanced axis formatting\nhidin
g first and last label)");
$graph->subtitle->SetFont(FF_FONT1,FS_NORMAL);
$graph->xgrid->Show();

$graph->yaxis->SetPos(0);
$graph->yaxis->SetWeight(2);
$graph->yaxis->HideZeroLabel();
$graph->yaxis->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->SetColor('black','darkblue');
$graph->yaxis->HideTicks(true,false);
$graph->yaxis->HideFirstLastLabel();

$graph->xaxis->SetWeight(2);
$graph->xaxis->HideZeroLabel();
$graph->xaxis->HideFirstLastLabel();
$graph->xaxis->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->SetColor('black','darkblue');

$lp1 = new LinePlot($ydata,$xdata);
$lp1->SetColor('blue');
$lp1->SetWeight(2);

$lp2 = new LinePlot($y2data,$x2data);
list($xm,$ym)=$lp2->Max();
$lp2->SetColor('red');
$lp2->SetWeight(2);

$graph->Add($lp1);
$graph->Add($lp2);
$graph->Stroke();

?>
```

上記のサンプルは基本的な設定を表しています。X 軸と Y 軸には修正を加えることができます。たとえば、以下の通りです。

- ラベルをプロット エリアの内側、あるいは外側に配置する
- 目盛りをプロット エリアの内側、あるいは外側に配置する

軸の形式は、[Graph::SetAxisSize\(\)](#) メソッドで指定できます。指定できる軸は以下の通りです。

- AXSTYLE_SIMPLE — 標準的な 2 軸グラフ
- AXSTYLE_BOXIN — ラベルと目盛りが内側にある 4 軸の関数形式のグラフ
- AXSTYLE_BOXOUT — ラベルと目盛りが外側にある 4 軸の関数形式のグラフ

7.19.3 スケール ラベルの位置を調整する

実際のラベルを調整する方法は、このマニュアルの「7.11 テキスト スケールのラベルを調整する」を参照してください。ラベルの余白（軸とラベルの間隔）を調節するには、[Axis::SetLabelMargin\(\)](#) メソッドを使用します。

また、ラベルの書式（フォントやカラー、角度など）を調節するには、[Axis::SetFont\(\)](#) メソッドと [Axis::SetColor\(\)](#) メソッドを使用します。詳細は、[Axis](#) クラス リファレンスを参照すると、様々な設定方法を確認できます。

[Axis::SetLabelAlign\(\)](#) メソッドを使用すると、ラベルの配置方法を設定できます。デフォルトでは、ラベルは各目盛りの中央に配置されます。また、[Axis::SetLabelAngle\(\)](#) メソッドによりラベルの角度を調節できます。しかしラベルの位置揃えをしたい場合もあるでしょう。デフォルトでは各目盛りの中央に配置されます。メソッドを使うと、位置を調節できます。

2 つ目は、軸に対してラベルの角度を調整します。このメソッドは、X 軸に長いラベルを使用する場合に便利です。

7.19.4 スケール ラベルをフォーマットする

表示されるラベルを完全に制御するために、すべてのラベルに適用されるコールバック関数を定義することができます。コールバック関数はラベルを唯一の引数として呼び出され、グラフに表示された値が返されます。これは、例えば目盛りに整形した時間と分を表示する場合などに便利です。詳細なリファレンスについては以下の 2 つのメソッドを参照してください。[Graph::SetLabelFormatCallback\(\)](#) と [Graph::SetLabelFormat\(\)](#);

7.19.5 Y 軸を反転する

Y 軸を反転することで、グラフをより魅力的に表現できる事があります

Y 軸を反転すると、最小値が最上部に表示され、軸を下がるにつれて値が増加します。

JpGraph ではこの機能を直接サポートするわけではありませんが、コードに数行記述するだけでこの機能を実現できます。

Y 軸を反転したグラフの例を掲載します。

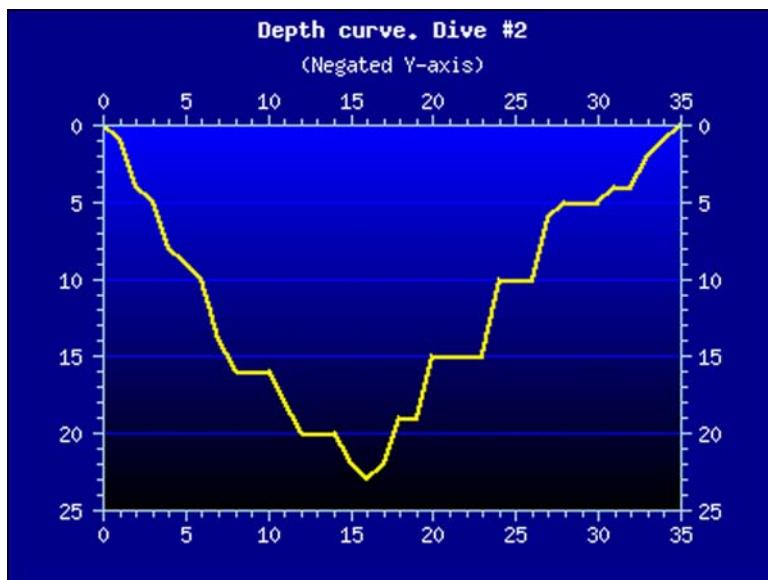


図 99:Y 軸を反転した例

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

// Callback to negate the argument
function _cb_negate($aVal) {
    return round(-$aVal);
}

// A fake depth curve
$ydata = array(0,1,4,5,8,9,10,14,16,16,16,16,18,20,20,20,2
2,22,5,22,19,19,15,15,15,15,10,10,10,6,5,5,4,4,2,1,0);

// Negate all data
$n = count($ydata);
for($i=0; $i<$n; ++$i) {
    $ydata[$i] = round(-$ydata[$i]);
}

// Basic graph setup
$graph = new Graph(400,300,"auto");
$graph->SetScale("linlin");
$graph->img->SetMargin(50,50,60,40);
$graph->SetMarginColor('darkblue');
$graph->SetColor('darkblue');
$graph->SetAxisStyle(AXSTYLE_BOXOUT);
$graph-
>SetBackgroundImage("blueblack400x300grad.png",1);
// $graph-
>SetBackgroundImage("lightbluedarkblue400x300grad.png
",1);

$graph->title->Set("Depth curve. Dive #2");
$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->title->SetColor("white");

$graph->subtitle->Set("(Negated Y-axis)");
$graph->subtitle->SetFont(FF_FONT1,FS_NORMAL);
```

```
$graph->subtitle->SetColor("white");
// Setup axis
$graph->yaxis->SetLabelFormatCallback("_cb_negate");
$graph->xaxis->SetColor("lightblue", "white");
$graph->yaxis->SetColor("lightblue", "white");
$graph->ygrid->SetColor("blue");

$lp1 = new LinePlot($ydata);
$lp1->SetColor("yellow");
$lp1->SetWeight(2);

$graph->Add($lp1);
$graph->Stroke();
?>
```

この効果を得るためにには、以下の 2 ステップを踏みます。

1. データの Y 値の正負を逆にする。
2. スケール ラベルの正負を逆にするコールバック関数を作成する。

それだけです。こうすることで、上記のような効果が得られます。

7.20 自動スケールの上限を設定する

自動スケール機能を用いると、デフォルトでは最大値と最小値をグラフの上端と下端に合わせます。そのため、自動スケール機能を有効にしてグラフの描画を行うと、以下のような外見になります。

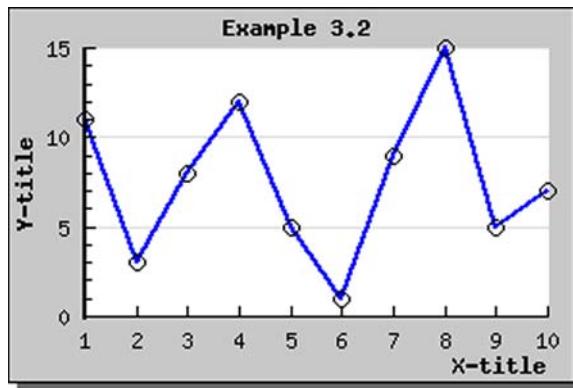


図 100: 自動スケール機能が有効なグラフの例

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,15,5,7);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot->mark->SetType(MARK_CIRCLE);

// Add the plot to the graph
$graph->Add($lineplot);

$graph->img->SetMargin(40,20,20,40);
$graph->title->Set("Example 3.2");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);
$graph->yaxis->SetWeight(2);
$graph->SetShadow();

// Display the graph
$graph->Stroke();
?>
```

しかし、場合として最高値の上部、あるいは最小値の下部に余白を持たしたい場合があります。これは grace の割合を目盛りに加えることで可能になります。しかし、場合として最高値の上部、ある

いは最小値の下部に余白を持たせたい場合があります。これを実現するため、たとえば Y 軸に 10% の空白を持たせるには、以下の通り [SetGrace\(\)](#) メソッドを使用します。

```
$graph->yaxis->scale->SetGrace(10,10);
```

こうすることで、スケールの上部と下部に最小で 10% の余白を追加できます。なぜ“最小”という表現をしたかというと、特定のチック値によっては、偶数のチックマーク上にスケール フォールの終わりを作るために余白が少し多くなることがあるからです。

これにより、グラフは以下のような外見になります。

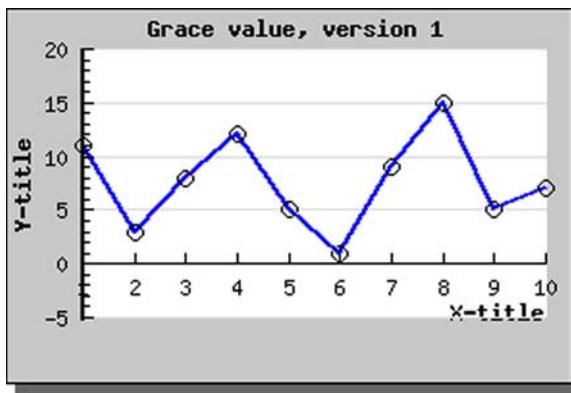


図 101:Y 軸の上部と下部に 10% の余白を追加する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,15,5,7);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");
$graph->yaxis->scale->SetGrace(10,10);

// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot->mark->SetType(MARK_CIRCLE);

// Add the plot to the graph
$graph->Add($lineplot);

$graph->xaxis->SetPos("min");
$graph->img->SetMargin(40,20,20,40);
$graph->title->Set("Grace value, version 1");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);
$graph->yaxis->SetWeight(2);
$graph->SetShadow();

// Display the graph
$graph->Stroke();
?>
```

X 軸の位置を調節していないので、X 軸が Y=0 の場所に表示されたままになっています。X 軸を最下部に表示するためには、以下の行を追加します。

```
$graph->xaxis->SetPos("min");
```

そのため X 軸は常に Y 値の最小の値を保持します。これにより、下記のような外見が得られます。

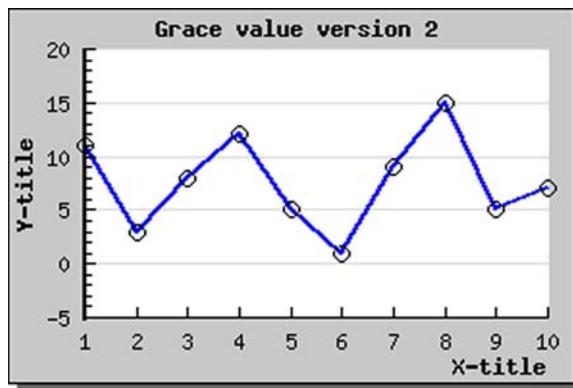


図 102:X 軸を最下部に持ってきた例

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,15,5,7);

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");
$graph->yaxis->scale->SetGrace(10,10);

// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot->mark->SetType(MARK_CIRCLE);

// Add the plot to the graph
$graph->Add($lineplot);

$graph->img->SetMargin(40,20,20,40);
$graph->title->Set("Grace value version 2");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->xaxis->SetPos('min');

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);
$graph->yaxis->SetWeight(2);
$graph->SetShadow();

// Display the graph
$graph->Stroke();
?>

```

7.21 バンドパターンとグラフのカラーを追加する

グラフをより分かりやすくするため、JpGraph では X-Y 座標系のすべてのグラフに対して、バンドを縦方向、あるいは横方向に追加できます。バンドとは、横あるいは縦方向に伸びた長方形のエリアのことです。これは、バンドを X 座標 3 と 5 で定義した場合、バンドのエリアは X 軸の 2 つの値の間とすべての Y 軸の値で構成される範囲になる、ということです。

この機能を使用するには、あらかじめ `jpgraph_plotband.php` ファイルをインクルードしておく必要があります。

現状では、以下の 8 種類のバンドを使用できます。また、バンドは独自にカスタマイズもできます。分かりやすくするために、各図には 1 種類のパターンしか用いていません。

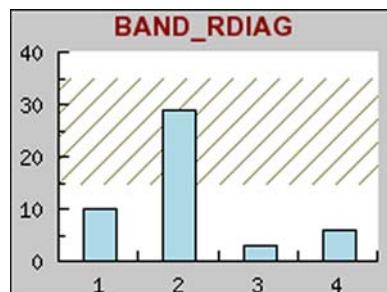


図 103

```

<?php
// Illustration of the different patterns for bands
// $Id: smallstaticbandsex1.php,v 1.1 2002/09/01 21:
51:08 aditus Exp $
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(10,29,3,6);

// Create the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->SetMargin(25,10,20,20);

// Add 10% grace ("space") at top and bottom of Y-
scale.
$graph->yscale->SetGrace(10);

// Create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("lightblue");

// Position the X-axis at the bottom of the plotare
$graph->xaxis->SetPos("min");

$graph->ygrid->Show(false);

// .. and add the plot to the graph
$graph->Add($bplot);

// Add band
$band = new PlotBand(HORIZONTAL,BAND_RDIAG,15,35,'khaki4');
$band->ShowFrame(false);
$graph->AddBand($band);

// Set title
$graph->title->SetFont(FF_ARIAL,FS_BOLD,10);
$graph->title->SetColor('darkred');
$graph->title->Set('BAND_RDIAG');

$graph->Stroke();
?>

```

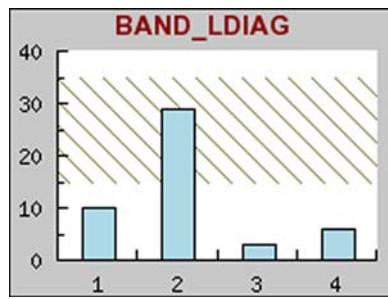


図 104:

```

<?php
// Illustration of the different patterns for bands
// $Id: smallstaticbandsex2.php,v 1.1 2002/09/01 21:
51:08 aditus Exp $
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(10,29,3,6);

// Create the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->SetMargin(25,10,20,20);

// Add 10% grace ("space") at top and bottom of Y-
scale.
$graph->yscale->SetGrace(10);

// Create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("lightblue");

// Position the X-axis at the bottom of the plotare
$graph->xaxis->SetPos("min");

$graph->ygrid->Show(false);

// .. and add the plot to the graph
$graph->Add($bplot);

// Add band
$band = new PlotBand(HORIZONTAL,BAND_LDIAG,15,35,'khaki4');
$band->ShowFrame(false);
$graph->AddBand($band);

// Set title
$graph->title->SetFont(FF_ARIAL,FS_BOLD,10);
$graph->title->SetColor('darkred');
$graph->title->Set('BAND_LDIAG');

$graph->Stroke();
?>

```

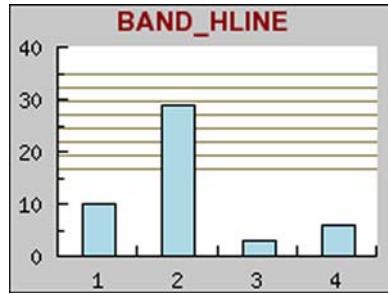


図 105

```
<?php
// Illustration of the different patterns for bands
// $Id: smallstaticbandsex7.php,v 1.1 2002/09/01 21:
51:08 aditus Exp $
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

$datay=array(10,29,3,6);

// Create the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->SetMargin(25,10,20,20);

// Add 10% grace ("space") at top and bottom of Y-
scale.
$graph->yscale->SetGrace(10);

// Create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("lightblue");

// Position the X-axis at the bottom of the plotare
$graph->xaxis->SetPos("min");

$graph->ygrid->Show(false);

// .. and add the plot to the graph
$graph->Add($bplot);

// Add band
$band = new PlotBand(HORIZONTAL,BAND_HLINE,15,35,'k
haki4');
$band->ShowFrame(false);
$graph->AddBand($band);

// Set title
$graph->title->SetFont(FF_ARIAL,FS_BOLD,10);
$graph->title->SetColor('darkred');
$graph->title->Set('BAND_HLINE');

$graph->Stroke();
?>
```

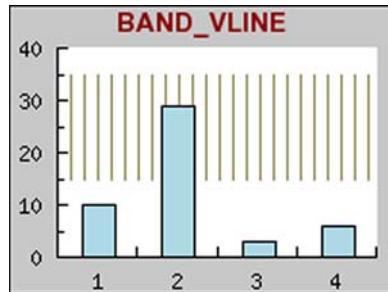


図 106

```
<?php
// Illustration of the different patterns for bands
// $Id: smallstaticbandsex6.php,v 1.1 2002/09/01 21:
51:08 aditus Exp $
include ("./jpgraph.php");
include ("./jpgraph_bar.php");

$datay=array(10,29,3,6);

// Create the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->SetMargin(25,10,20,20);

// Add 10% grace ("space") at top and bottom of Y-
scale.
$graph->yscale->SetGrace(10);

// Create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("lightblue");

// Position the X-axis at the bottom of the plotare
$graph->xaxis->SetPos("min");

$graph->ygrid->Show(false);

// .. and add the plot to the graph
$graph->Add($bplot);

// Add band
$band = new PlotBand(HORIZONTAL,BAND_VLINE,15,35,'k
haki4');
$band->ShowFrame(false);
$graph->AddBand($band);
```

```
// Set title
$graph->title->SetFont(FF_ARIAL,FS_BOLD,10);
$graph->title->SetColor('darkred');
$graph->title->Set('BAND_VLINE');
```

```
$graph->Stroke();
?>
```

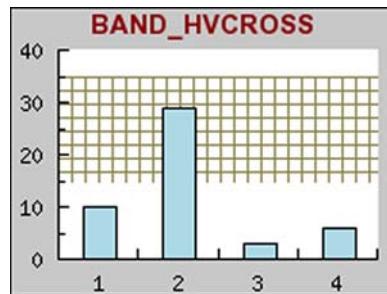


図 107

```
<?php
// Illustration of the different patterns for bands
// $Id: smallstaticbandsex9.php,v 1.1 2002/09/01 21:
51:08 aditus Exp $
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(10,29,3,6);

// Create the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->SetMargin(25,10,20,20);

// Add 10% grace ("space") at top and bottom of Y-scale.
$graph->yscale->SetGrace(10);

// Create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("lightblue");
```

```
// Position the X-axis at the bottom of the plotare
$graph->xaxis->SetPos("min");

$graph->ygrid->Show(false);

// .. and add the plot to the graph
$graph->Add($bplot);

// Add band
$band = new PlotBand(HORIZONTAL,BAND_HVCROSS,15,
35,'khaki4');
$band->ShowFrame(false);
$graph->AddBand($band);

// Set title
$graph->title->SetFont(FF_ARIAL,FS_BOLD,10);
$graph->title->SetColor('darkred');
$graph->title->Set('BAND_HVCROSS');

$graph->Stroke();
?>
```

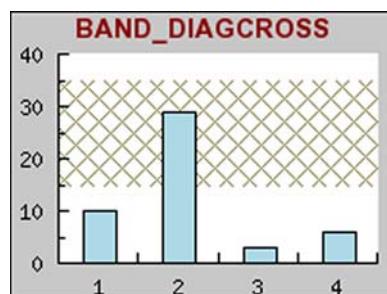


図 108

```
<?php
// Illustration of the different patterns for bands
// $Id: smallstaticbandsex10.php,v 1.1 2002/09/01 21:
51:08 aditus Exp $
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(10,29,3,6);

// Create the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->SetMargin(25,10,20,20);
```

```
// Add 10% grace ("space") at top and bottom of Y-scale.
$graph->yscale->SetGrace(10);

// Create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("lightblue");

// Position the X-axis at the bottom of the plotare
$graph->xaxis->SetPos("min");

$graph->ygrid->Show(false);

// .. and add the plot to the graph
```

```

$graph->Add($bplot);

// Add band
$band = new PlotBand(HORIZONTAL,BAND_DIAGCROSS,1
5,35,'khaki4');
$band->ShowFrame(false);
$graph->AddBand($band);

// Set title
$graph->title->SetFont(FF_ARIAL,FS_BOLD,10);
$graph->title->SetColor('darkred');
$graph->title->Set('BAND_DIAGCROSS');

$graph->Stroke();
?>

```

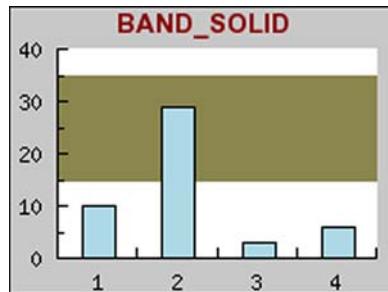


図 109

```

<?php
// Illustration of the different patterns for bands
// $Id: smallstaticbandsex3.php,v 1.1 2002/09/01 21:
51:08 aditus Exp $
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(10,29,3,6);

// Create the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->SetMargin(25,10,20,20);

// Add 10% grace ("space") at top and bottom of Y-
scale.
$graph->yscale->SetGrace(10);

// Create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("lightblue");

// Position the X-axis at the bottom of the plotare
$graph->xaxis->SetPos("min");

$graph->ygrid->Show(false);

// .. and add the plot to the graph
$graph->Add($bplot);

// Add band
$band = new PlotBand(HORIZONTAL,BAND_SOLID,15,35,'k
haki4');
$band->ShowFrame(false);
$graph->AddBand($band);

// Set title
$graph->title->SetFont(FF_ARIAL,FS_BOLD,10);
$graph->title->SetColor('darkred');
$graph->title->Set('BAND_SOLID');

$graph->Stroke();
?>

```

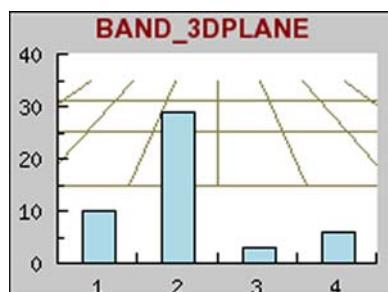


図 110

```

<?php
// Illustration of the different patterns for bands
// $Id: smallstaticbandsex8.php,v 1.1 2002/09/01 21:
51:08 aditus Exp $
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(10,29,3,6);

// Create the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->SetMargin(25,10,20,20);

// Add 10% grace ("space") at top and bottom of Y-
scale.
$graph->yscale->SetGrace(10);

// Create a bar pot
$bplot = new BarPlot($datay);

```

```
$bplot->SetFillColor("lightblue");

// Position the X-axis at the bottom of the plotare
$graph->xaxis->SetPos("min");

$graph->ygrid->Show(false);

// .. and add the plot to the graph
$graph->Add($bplot);

// Add band
$band = new PlotBand(HORIZONTAL,BAND_3DPLANE,15,3
5,"khaki4");
$band->ShowFrame(false);
$graph->AddBand($band);

// Set title
$graph->title->SetFont(FF_ARIAL,FS_BOLD,10);
$graph->title->SetColor('darkred');
$graph->title->Set('BAND_3DPLANE');

$graph->Stroke();
?>
```

これらのパターンをグラフに追加するには、[PlotBand::PlotBand\(\)](#) メソッドを使用します。使用したいパターンは定数を指定することで選択できます。上記の図で正確な定数を確認してください。バンドの座標の最小値と最大値だけではなく、バンドを垂直方向や水平方向に伸ばすかどうかも指定する必要があります。最小値、最大値が自動的に設定される min と max という特別な値も使用することができます。この特別な値はプロットエリアに収まるように自動的に修正されます。

7.21.1 パターンをカスタマイズする

すべてのパターンは以下の通りカスタマイズが可能です。

- … バンドのカラーを変更する。
- … [PlotBand::SetDensity\(\)](#) メソッドを使用してパターンの密度を変更する。密度は 1 から 100 までの整数値で指定でき、値が高くなるほど密度が高くなります。たとえば、3DPLANE 形式で密度を 60 に設定した場合、下記のような外見が得られます。

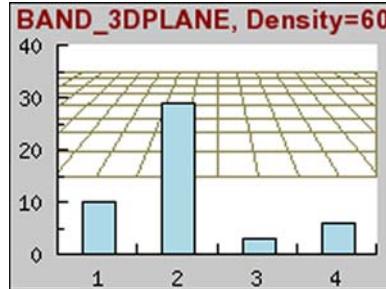


図 111:パターンの密度を高くした例

```
<?php
// Illustration of the different patterns for bands
// $Id: smallstaticbandsex4.php,v 1.1 2002/09/01 21:
51:08 aditus Exp $
include ("../jgraph.php");
include ("../jgraph_bar.php");

$datay=array(10,29,3,6);

// Create the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->SetMargin(25,10,20,20);

// Add 10% grace ("space") at top and bottom of Y-
scale.
$graph->yscale->SetGrace(10);

// Create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("lightblue");

// Position the X-axis at the bottom of the plotare
$graph->xaxis->SetPos("min");

$graph->ygrid->Show(false);

// .. and add the plot to the graph
$graph->Add($bplot);

// Add band
$band = new PlotBand(HORIZONTAL,BAND_3DPLANE,15,3
5,"khaki4");
$band->SetDensity(60);
$band->ShowFrame(true);
$graph->AddBand($band);

// Set title
$graph->title->SetFont(FF_ARIAL,FS_BOLD,10);
$graph->title->SetColor('darkred');
$graph->title->Set('BAND_3DPLANE, Density=60');
```

```
$graph->Stroke();
```

?>

- ... フレームをパターンの周囲に表示、非表示するには、[PlotBand::ShowFrame\(\)](#) メソッドを使用します。フレームはバンドと同じカラーで表示されます。
- ... バンドとプロットの前後関係を [PlotBand::SetOrder\(\)](#) メソッドで指定できます。デフォルトではバンドはプロットの下位にあります。

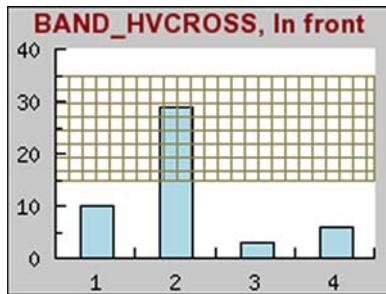


図 112:パターンをプロットの上部に描画する

```
<?php
// Illustration of the different patterns for bands
// $Id: smallstaticbandsx5.php,v 1.1 2002/09/01 21:51:08 aditus Exp $
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(10,29,3,6);

// Create the graph.
$graph = new Graph(200,150,"auto");
$graph->SetScale("textlin");
$graph->SetMargin(25,10,20,20);

// Add 10% grace ("space") at top and bottom of Y-scale.
$graph->yscale->SetGrace(10);

// Create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("lightblue");

// Position the X-axis at the bottom of the plotare
```

```
$graph->xaxis->SetPos("min");

$graph->ygrid->Show(false);

// .. and add the plot to the graph
$graph->Add($bplot);

// Add band
$band = new PlotBand(HORIZONTAL,BAND_HVCROSS,15,35,'khaki4');
$band->ShowFrame(true);
$band->SetOrder(DEPTH_FRONT);
$graph->AddBand($band);

// Set title
$graph->title->SetFont(FF_ARIAL,FS_BOLD,10);
$graph->title->SetColor('darkred');
$graph->title->Set('BAND_HVCROSS, In front');

$graph->Stroke();
?>
```

注意: 3DPLANE 形式のパターンでは、奥行きの指定が可能です。以下の通りに指定します。水平の値を変更するために関数を直接呼び出すことはできませんが、下のようにすれば可能です。

```
$band->prect->SetHorizon($aHorizon)
```

バンドは 3D の面と考えられます。

最後に、バンドを組み合わせる方法を紹介します。

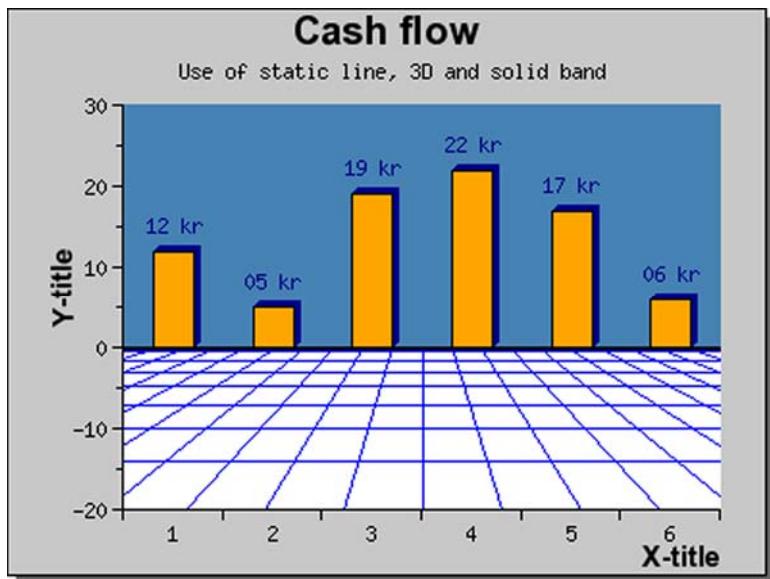


図 113: ラインと3DPLANE、SOLID バンドを組み合わせる

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_bar.php");

$datay=array(12,5,19,22,17,6);

// Create the graph.
$graph = new Graph(400,300,"auto");
$graph->img->SetMargin(60,30,50,40);
$graph->SetScale("textlin");
$graph->SetShadow();

$graph->title->SetFont(FF_ARIAL,FS_BOLD,15);
$graph->title->Set("Cash flow ");
$graph->subtitle-
>Set("Use of static line, 3D and solid band");

// Turn off Y-grid (it's on by default)
$graph->ygrid->Show(false);

// Add 10% grace ("space") at top of Y-scale.
$graph->yscale->SetGrace(10);
$graph->yscale->SetAutoMin(-20);

// Turn the tick mark out from the plot area
$graph->xaxis->SetTickSide(SIDE_DOWN);
$graph->yaxis->SetTickSide(SIDE_LEFT);

// Create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor("orange");
$bplot->SetShadow("darkblue");
    
```

```

// Show the actual value for each bar on top/bottom
$bplot->value->Show(true);
$bplot->value->SetFormat("%02d kr");

// Position the X-axis at the bottom of the plotare
$graph->xaxis->SetPos("min");

// .. and add the plot to the graph
$graph->Add($bplot);

// Add upper and lower band and use no frames
$band[0]=new PlotBand(HORIZONTAL,BAND_3DPLANE,"mi
n",0,"blue");
$band[0]->ShowFrame(false);
$band[0]->SetDensity(20);
$band[1]=new PlotBand(HORIZONTAL,BAND_SOLID,0,"max
","steelblue");
$band[1]->ShowFrame(false);
$graph->Add($band);

$graph->Add(new PlotLine(HORIZONTAL,0,"black",2));

// $graph->title->Set("Test of bar gradient fill");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->yaxis->title->SetFont(FF_ARIAL,FS_BOLD,11);
$graph->xaxis->title->SetFont(FF_ARIAL,FS_BOLD,11);

$graph->Stroke();
?>
    
```

7.22 静止ラインをプロットに追加する

JpGraph ではバンドに加えてラインをグラフに追加できます。その例は上記の 11 で実際に示しています。このラインは、[PlotLine](#) クラスのインスタンスとして作成できます。以下の通りに記述すると、2 ピクセル幅の水平線を $Y=0$ の位置に描画できます。

```

$line = new PlotLine(HORIZONTAL,0,"black",2);
$graph->Add($line);
    
```

8 X、Y プロット以外を使用する

非 XY プロットには以下の種類が含まれます。

- 円プロット (2D および 3D)
- レーダー プロット
- 極プロット
- ガント チャート

基本的な違いは、これらのプロットは Graph クラスの拡張バージョンのクラスを使用するということです。そのため、XY プロットと非 XY プロットを組み合わせることができません。たとえば、ライン プロットと極プロットを同時に表示させることはできません。

- 2D および 3D 円プロットを作成するには、PieGraph クラスを使用します。
- レーダー プロットを作成するには RadarGraph クラスを使用します。
- 極 プロットを作成するには PolarGraph クラスを使用します。
- ガント チャートを作成するには GanttGraph クラスを使用します。

8.1 レーダー プロット

レーダー プロットは、複数の対象について数値を比較する場合に良く用いられます。それらは、個人の能力やスキルなどを対称的に表示させたい場合によく使用されます。レーダー プロットの例を以下に掲載します(ウェブ プロット、スパイダー プロットとも呼ばれます)。ただ、レーダー プロットはグラフから正確な値を読み取ることには適していません。

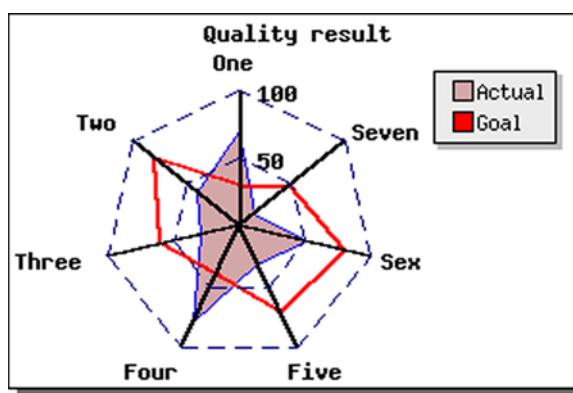


図 114:2 個のプロットを持つ一般的なレーダー プロットの例

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_radar.php");

// Create the basic rtadar graph
$graph = new RadarGraph(300,200,"auto");
```

```
// Set background color and shadow
$graph->SetColor("white");
$graph->SetShadow();

// Position the graph
$graph->SetCenter(0.4,0.55);
```

```
// Setup the axis formatting
$graph->axis->SetFont(FF_FONT1,FS_BOLD);
$graph->axis->SetWeight(2);

// Setup the grid lines
$graph->grid->SetLineStyle("longdashed");
$graph->grid->SetColor("navy");
$graph->grid->Show();
$graph->HideTickMarks();

// Setup graph titles
$graph->title->Set("Quality result");
$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph-
>SetTitles(array("One","Two","Three","Four","Five","Six",
,"Seven","Eight","Nine","Ten"));
// Create the first radar plot
$pplot = new RadarPlot(array(30,80,60,40,71,81,47));

```

```
$plot->SetLegend("Goal");
$pplot->SetColor("red","lightred");
$pplot->SetFill(false);
$pplot->SetLineWeight(2);

// Create the second radar plot
$pplot2 = new RadarPlot(array(70,40,30,80,31,51,14));
$pplot2->SetLegend("Actual");
$pplot2->SetColor("blue","lightred");

// Add the plots to the graph
$graph->Add($pplot2);
$graph->Add($plot);

// And output the graph
$graph->Stroke();

?>
```

-
- 各データ ポイントに軸がある。
- 各軸には任意のタイトルを付加でき、自動的に配置される。
- レーダー プロットは塗りつぶしにできる。
- 他のグラフと同様の形式でカラー や ライン の幅を制御できる。
- タイトルや凡例をつけることができる。
- 1 番目の軸は必ず垂直になっており、ラベルが付加されている。
- グリッドを使用できる(上記ではダッシュ線で描画)。
- 目盛りを描画できる(上記では省略)。
- グラフのサイズや位置を制御できる。
- 複数のプロットを 1 つのグラフ内に描画できる。

以下の例では、簡単なレーダー プロットと複雑なレーダー プロットの描画方法を解説します。これまでと同様の手順でグラフを作成できます。

8.1.1 簡単なレーダー プロットを作成する

5 個のデータ ポイントを持つシンプルなレーダー プロットを作成してみましょう。

まず初めに、レーダー プロットを含む拡張モジュールををインクルードしておく必要があります。
"jpgraph_radar.php".

非常に簡単なレーダー プロットは以下のようにして作成できます。

```
(File:radarex1.php)
<?php
include ("../jpgraph.php");
include ("../jpgraph_radar.php");

// プロットを行うデータ
$data = array(55,80,46,71,95);

// グラフとプロットを作成する
```

```
$graph = new RadarGraph(250,200,"auto");
$plot = new RadarPlot($data);

// プロットをグラフに追加
$graph->Add($plot);
$graph->Stroke();
?>
```

これにより、以下の結果が出力されます。

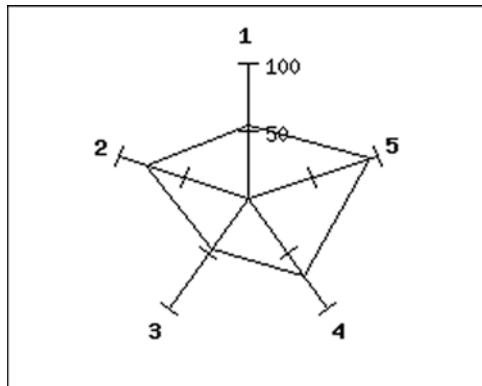


図 115: デフォルトの設定で作成した簡単なレーダー プロット

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_radar.php");

// Some data to plot
$data = array(55,80,46,71,95);

// Create the graph and the plot
$graph = new RadarGraph(250,200,"auto");
$plot = new RadarPlot($data);

// Add the plot and display the graph
$graph->Add($plot);
$graph->Stroke();
?>
```

レーダー プロットの位置やサイズを変更するには、[SetSize\(\)](#) メソッドや[SetCenter\(\)](#) メソッドを使用します。

また、レーダー プロットの内側を塗りつぶしたい場合は、[SetFillColor\(\)](#) メソッドを使用します。このメソッドの使用方法は次のようにになります。



図 116: レーダー プロットのサイズ、位置を変更して、内側を塗りつぶす例

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_radar.php");

// Some data to plot
$data = array(55,80,46,71,95);

// Create the graph and the plot
$graph = new RadarGraph(300,200,"auto");

$graph->title->Set('Weekly goals');
$graph->subtitle->Set('Year 2003');
```

```
$plot = new RadarPlot($data);
$plot->SetFillColor('lightred');
$graph->SetSize(0.6);
$graph->SetPos(0.5,0.6);
```

```
// Add the plot and display the graph
$graph->Add($plot);
$graph->Stroke();
?>
```

8.1.2 プロットの軸と凡例にタイトルを指定する

それぞれの軸に対して、値の数値だけではなく文字列を表示させることもできます。タイトルを指定するには、`SetTitles()` メソッドを使用します。たとえば、それぞれの軸に月名を入れたい場合、以下のように記述します。

```
$titles = $gDateLocale->GetShortMonth();
$graph->SetTitles($titles);
```

レーダー プロットでは、タイトルやプロット ポイントは反時計回りで描画されます。逆順で描画したい場合は、データ配列とタイトル配列を逆順に変換すれば可能です。

凡例を指定するには、それぞれのレーダー プロットに対して [SetLegend\(\)](#) メソッドを使用します。

8.1.3 レーダー プロットにグリッド ラインを描画する

それぞれの目盛りを使用してグリッドを作成できます。グリッドはグラフの `grid` プロパティを使用してアクセスできます。グリッドを有効にして、そのスタイルを点線にしたい場合は、以下のように記述します。

```
$graph->grid->Show();
$graph->grid->SetLineStyle("dotted");
```

これにより、以下のグラフが表示されます。

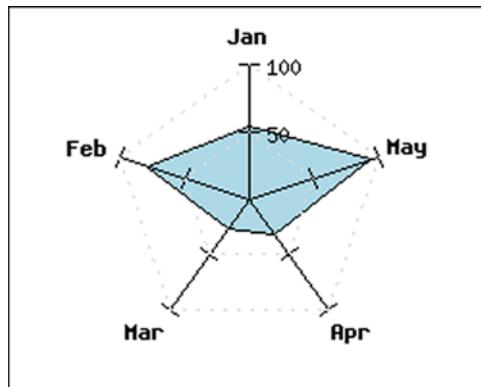


図 117: 点線のグリッド線をグラフに追加する

```
<?php
include ("../jraph.php");
include ("../jraph_radar.php");

// Some data to plot
$data = array(55,80,26,31,95);

// Create the graph and the plot
$graph = new RadarGraph(250,200,"auto");

// Create the titles for the axis
$titles = $gDateLocale->GetShortMonth();
$graph->SetTitles($titles);
```

```
// Add grid lines
$graph->grid->Show();
$graph->grid->SetLineStyle('dotted');

$pplot = new RadarPlot($data);
$pplot->SetFillColor('lightblue');

// Add the plot and display the graph
$graph->Add($pplot);
$graph->Stroke();
?>
```

プロットはグリッド線の上部、そしてそれぞれの軸の下部に描画されます。そのため、グリッド線には非表示になる部分があります。

グリッド線をより見やすくするには、SetColor() メソッドなどを使用してカラーを変更します。

また、レーダー グラフの背景色を変更することもできます。これも、グラフ オブジェクトのSetColor() メソッドを使用します。

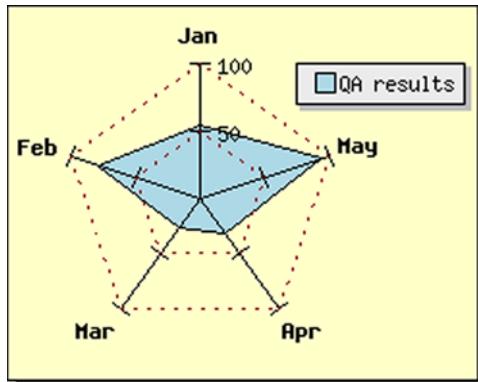


図 118: 背景色を変更する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_radar.php");

// Some data to plot
$data = array(55,80,26,31,95);

// Create the graph and the plot
$graph = new RadarGraph(250,200,"auto");

// Add a drop shadow to the graph
$graph->SetShadow();

// Create the titles for the axis
$titles = $gDateLocale->GetShortMonth();
$graph->SetTitles($titles);
$graph->SetColor('lightyellow');

// ADjust the position to make more room
// for the legend
$graph->SetCenter(0.4,0.5);

// Add grid lines
$graph->grid->Show();
$graph->grid->SetColor('darkred');
$graph->grid->SetLineStyle('dotted');

$pplot = new RadarPlot($data);
$pplot->SetFillColor('lightblue');
$pplot->SetLegend("QA results");

// Add the plot and display the graph
$graph->Add($pplot);
$graph->Stroke();
?>
```

8.1.4 1つのグラフ内に複数のレーダー プロットを追加する

複数のレーダー プロットを 1 つのグラフにまとめて表示することができます。また、色の塗りつぶしを行うレーダー プロットを使用する場合は、追加した順番にグラフが描画されるので、指定する順番に注意する必要があります。

簡単な例を以下に表示します。

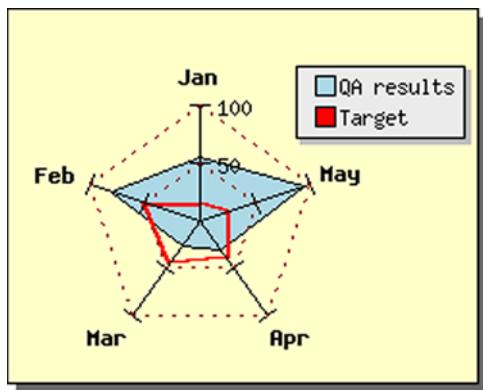


図 119: 1 つのグラフ内に複数のレーダー プロットを表示する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_radar.php");

// Some data to plot
$data = array(55,80,26,31,95);
$data2 = array(15,50,46,39,25);

// Create the graph and the plot
$graph = new RadarGraph(250,200,"auto");

// Add a drop shadow to the graph
$graph->SetShadow();

// Create the titles for the axis
$titles = $gDateLocale->GetShortMonth();
$graph->SetTitles($titles);
$graph->SetColor('lightyellow');

// ADjust the position to make more room
// for the legend
$graph->SetCenter(0.4,0.55);
$graph->SetSize(0.6);

// Add grid lines
$graph->grid->Show();
$graph->grid->SetColor('darkred');
$graph->grid->SetLineStyle('dotted');

$plot = new RadarPlot($data);
$plot->SetFillColor('lightblue');
$plot->SetLegend("QA results");

$plot2 = new RadarPlot($data2);
$plot2->SetLegend("Target");
$plot2->SetColor('red');
$plot2->SetFill(false);
$plot2->SetLineWidth(2);

// Add the plot and display the graph
$graph->Add($plot);
$graph->Add($plot2);
$graph->Stroke();
?>

```

8.1.5 レーダー プロットにプロットマークを表示する

ライン プロットと同様に、レーダー プロットにプロット マークを表示させることができます。プロットマークのプロパティには `RadarPlot::mark` を通してアクセスします。以下は赤いボールをマーカーとして表示させた例です。

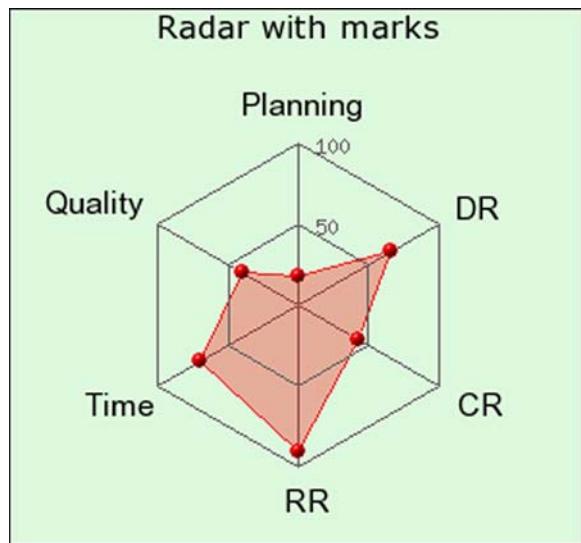


図 120: レーダー プロットにプロットマークを表示させます

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_radar.php");

$titles=array('Planning','Quality','Time','RR','CR','DR');
$data=array(18, 40, 70, 90, 42,66);

$graph = new RadarGraph (300,280,"auto");

$graph->title->Set("Radar with marks");
$graph->title->SetFont(FF_VARDANA,FS_NORMAL,12);

$graph->SetTitles($titles);
$graph->SetCenter(0.5,0.55);
$graph->HideTickMarks();
$graph->SetColor('lightgreen@0.7');
$graph->axis->SetColor('darkgray');
$graph->grid->SetColor('darkgray');

$graph->grid->Show();

$graph->axis->title->SetFont(FF_ARIAL,FS_NORMAL,12);
$graph->axis->title->SetMargin(5);
$graph->SetGridDepth(DEPTH_BACK);
$graph->SetSize(0.6);

$pplot = new RadarPlot($data);
$pplot->SetColor('red@0.2');
$pplot->SetLineWeight(1);
$pplot->SetFillColor('red@0.7');

$pplot->mark->SetType(MARK_IMG_SBALL,'red');

$graph->Add($pplot);
$graph->Stroke();
?>

```

8.2 円プロット

これまで XY 座標に基づいたプロットだけを扱ってきました。JpGraph で作成できるグラフはこれだけではありません。その一つの形式が円プロットです。JpGraph では 2D と 3D の円プロットをサポートしています。また、2D 円プロットの場合は後述するとおり 2 種類のものがあります。

XY プロットとの最大の違いは、円プロットは Graph() オブジェクトではなく [PieGraph\(\)](#) オブジェクトを基に作成することです。これを使用するには、jpgraph_pie.php ファイル、あるいは jpgraph_pie3d.php ファイルをインクルードする必要があります。

まずはデフォルトの最もシンプルな円プロットを作成する例を掲載します。

```

include ("../jpgraph.php");
include ("../jpgraph_pie.php");

$data = array(40,60,21,33);

$graph = new PieGraph(300,200);
$graph->SetShadow();

$graph->title->Set("A simple Pie plot");

$p1 = new PiePlot($data);
$graph->Add($p1);
$graph->Stroke();

```

このコードを使うと、下記のような外見のグラフが出力されます。

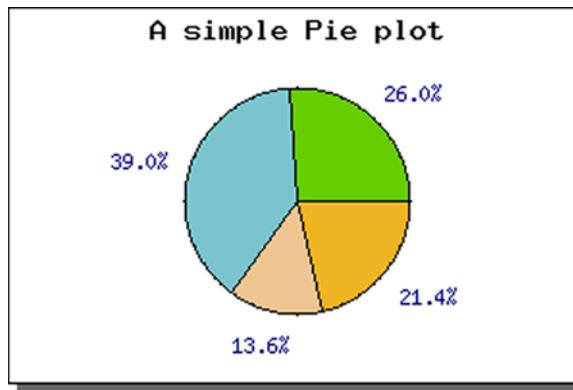


図 121: もっとも簡単な円プロットの例

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_pie.php");

$data = array(40,60,21,33);

$graph = new PieGraph(300,200,"auto");
$graph->SetShadow();
```

```
$graph->title->Set("A simple Pie plot");

$p1 = new PiePlot($data);
$graph->Add($p1);
$graph->Stroke();

?>
```

円プロットでは以下ののような機能があります。

- デフォルトですべてのスライスの外側に値の割合が表示されます。
- すべてのスライスにカラーが自動的に設定されます。
- デフォルトで各スライスに黒い枠線が表示されます。
- 1番目のスライスは0度(3時の方向)から始まります。

また、円プロットの外見も様々に変更できます。たとえば、以下の通りです。

- 1番目のスライスの開始角度を変更する ([PiePlot::SetStartAngle\(\)](#))
- 円プロットの周りにある枠線を取り除く ([PiePlot::ShowBorder\(\)](#))
- 枠線の色を変更する ([PiePlot::SetColor\(\)](#))、
- スライスのカラーを変更する ([PiePlot::SetSliceColors\(\)](#))
- 円プロットのサイズや位置を変更する ([PiePlot::SetSize\(\)](#), [PiePlot::SetCenter\(\)](#))
- スライスのラベル(色、フォント、フォーマット、位置)を円プロットの [value プロパティ](#) にアクセスして変更する、たとえば ([PiePlot::value::SetFont\(\)](#))と指定します。ラベルのフォーマットや表示される値の変更の方法についてはこの章のいちばん下で読むことができます。

次に、円プロットに凡例を追加してみましょう。凡例は [SetLegends\(\)](#) メソッドを使用します。凡例を追加すると以下のような画像が出力されます。

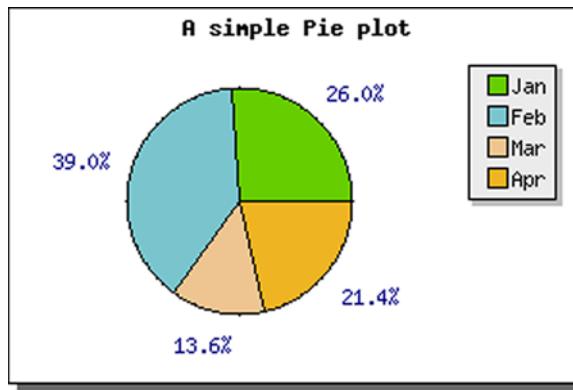


図 122: 円プロットに凡例を追加する

```

<?php
include ("../jgraph.php");
include ("../jgraph_pie.php");

$data = array(40,60,21,33);

$graph = new PieGraph(300,200,"auto");
$graph->SetShadow();

$graph->title->Set("A simple Pie plot");
    
```

```

$graph->title->SetFont(FF_FONT1,FS_BOLD);

$p1 = new PiePlot($data);
$p1->SetLegends($gDateLocale->GetShortMonth());
$p1->SetCenter(0.4);

$graph->Add($p1);
$graph->Stroke();

?>
    
```

上記の例では、円プロットの中心を少し左にずらし、凡例を表示するスペースを空けています。

凡例のテキストでは、数値の整形に `printf()` 形式の書式を指定することができます。この場合、文字列に渡される数値は絶対値かパーセント値のいずれかです。詳細はこの章の後半で説明します。

次に、円プロットのサイズと位置を調節しましょう。円プロットのサイズは [setSize\(\)](#) メソッドを使用します。また、円プロットの位置を中心に配置するには、[SetCenter\(\)](#) メソッドを使用します。サイズはピクセル単位の絶対値か、横幅もしくは縦幅と比較した割合のいずれかで指定できます。円プロットの位置は横幅と縦幅で指定できます。

1 つのグラフ内に、複数の円プロットを設置する方法を紹介します。下の例では、スライスの凡例を小さなフォントに設定しました。

サイズや配置の説明をするために、今回は 4 個の円プロットを作成し、それらをグラフの四隅に配置します。それにより、以下の画像が出力されます。

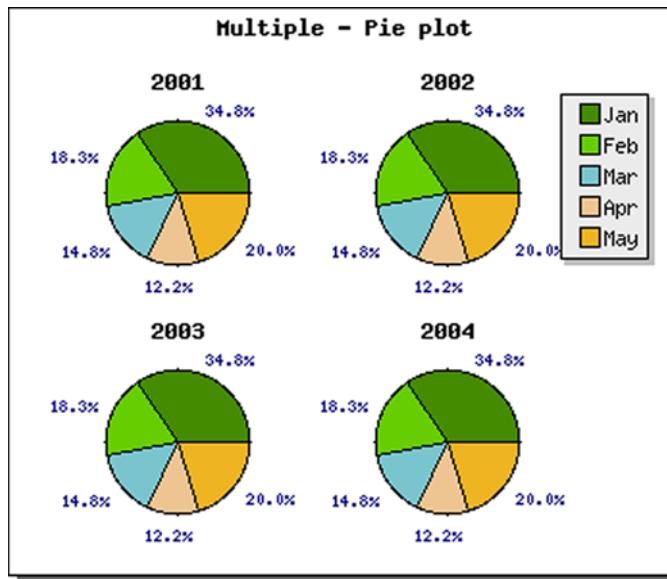


図 123: 複数の円プロットを 1 つのグラフ内に配置する

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_pie.php");

// Some data
$data = array(40,21,17,14,23);

// Create the Pie Graph.
$graph = new PieGraph(350,300,"auto");
$graph->SetShadow();

// Set A title for the plot
$graph->title->Set("Multiple - Pie plot");
$graph->title->SetFont(FF_FONTO,FS_BOLD);

// Create plots
$size=0.13;
$p1 = new PiePlot($data);
$p1-
>SetLegends(array("Jan","Feb","Mar","Apr","May"));
$p1->SetSize($size);
$p1->SetCenter(0.25,0.32);
$p1->value->SetFont(FF_FONTO);
$p1->title->Set("2001");

$p2 = new PiePlot($data);
$p2->SetSize($size);
$p2->SetCenter(0.65,0.32);
$p2->value->SetFont(FF_FONTO);
$p2->title->Set("2002");

$p3 = new PiePlot($data);
$p3->SetSize($size);
$p3->SetCenter(0.25,0.75);
$p3->value->SetFont(FF_FONTO);
$p3->title->Set("2003");

$p4 = new PiePlot($data);
$p4->SetSize($size);
$p4->SetCenter(0.65,0.75);
$p4->value->SetFont(FF_FONTO);
$p4->title->Set("2004");

$graph->Add($p1);
$graph->Add($p2);
$graph->Add($p3);
$graph->Add($p4);

$graph->Stroke();

?>

```

8.2.1 円プロットにガイド ラインを追加する

とても細かい円プロットではスペースが足りないので、各スライスの横にラベルを表示することができません。そのような場合、ラベルにガイド ラインを使用することができます。この機能を使用すると、円プロットから離れて配置されたラベルにスライスの中心からガイドラインが描画されます。

これを使用するには、PiePlot::SetGuideLines() メソッドを使用します。

```
$pieplot->SetGuideLines();
```

以下はこの機能を使用した例です。

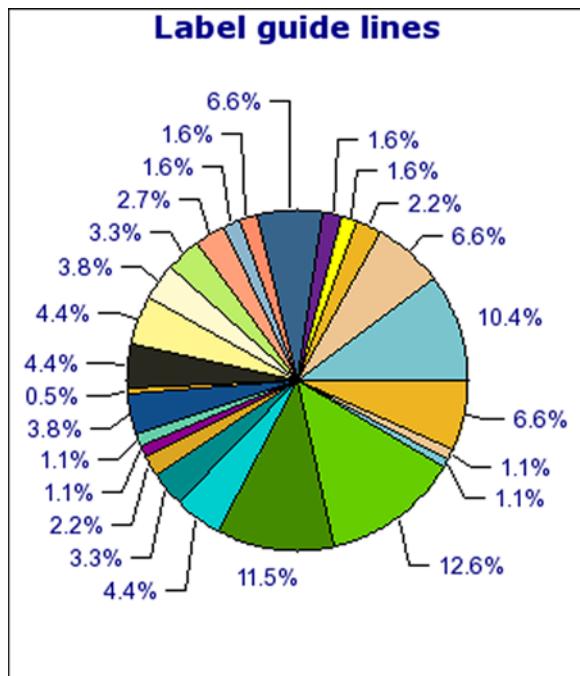


図 124: 円プロットにガイド ラインを使用する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_pie.php");

$data = array(19,12,4,3,3,12,3,3,5,6,7,8,8,1,7,2,2,4,6,8,2
1,23,2,2,12);

// Create the Pie Graph.
$graph = new PieGraph(300,350);

// Set A title for the plot
$graph->title->Set("Label guide lines");
$graph->title->SetFont(FF_VERDANA,FS_BOLD,12);
$graph->title->SetColor("darkblue");
$graph->legend->Pos(0.1,0.2);

// Create pie plot
$p1 = new PiePlot($data);

$p1->SetCenter(0.5,0.55);
$p1->SetSize(0.3);

// Enable and set policy for guide-lines
$p1->SetGuideLines();
$p1->SetGuideLinesAdjust(1.4);

// Setup the labels
$p1->SetLabelType(PIE_VALUE_PER);
$p1->value->Show();
$p1->value->SetFont(FF_ARIAL,FS_NORMAL,9);
$p1->value->SetFormat("%2.1f%");

// Add and stroke
$graph->Add($p1);
$graph->Stroke();

?>

```

上の例に表示されているように、Excel のような他のプログラムによって生成されたガイド ラインとよく似たものを作成できます。上記の改良に加えて、読みやすくするためにラベルを縦に並べるように指定することもできます。これは、SetGuideLines() の 2 つ目のパラメーターを 'false' に指定することで可能です。

```
$pieplot->SetGuideLines(true,false);
```

一つ目の引数はガイド ラインの表示・非表示を指定します。上の例と同じグラフを用いてこの例を表示します。

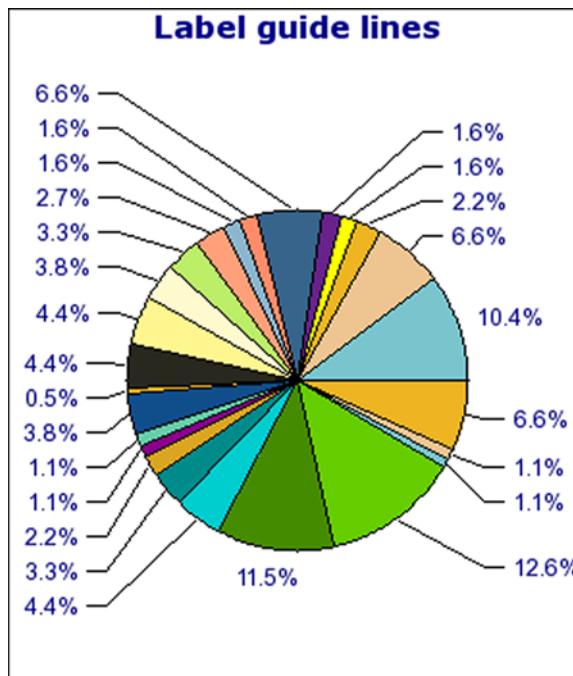


図 125: 円プロットにガイド ラインを使用し、ラベルを縦に並べる

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_pie.php");

$data = array(19,12,4,3,3,12,3,3,5,6,7,8,8,1,7,2,2,4,6,8,2
1,23,2,2,12);

// Create the Pie Graph.
$graph = new PieGraph(300,350);

// Set A title for the plot
$graph->title->Set("Label guide lines");
$graph->title->SetFont(FF_VERDANA,FS_BOLD,12);
$graph->title->SetColor("darkblue");
$graph->legend->Pos(0.1,0.2);

// Create pie plot
$p1 = new PiePlot($data);

$p1->SetCenter(0.5,0.55);
$p1->SetSize(0.3);

// Enable and set policy for guide-
lines. Make labels line up vertically
$p1->SetGuideLines(true,false);
$p1->SetGuideLinesAdjust(1.5);

// Setup the labels
$p1->SetLabelType(PIE_VALUE_PER);
$p1->value->Show();
$p1->value->SetFont(FF_ARIAL,FS_NORMAL,9);
$p1->value->SetFormat("%2.1f%");

// Add and stroke
$graph->Add($p1);
$graph->Stroke();

?>

```

ラベル間の縦の幅を設定することもできます。デフォルトでは、ラベルの間の幅はおおよそラベルフォントの高さの 40 %です。PiePlot::SetGuideLinesAdjust() を使用することによって、ラベルの下部と次のラベルの下部との間の幅の値を指定することができます。'1.0' を指定することは、ラベルの間にスペースを空けないことを意味しています。デフォルト値は 1.4 です。

この値を増減させることで、よりコンパクトにラベルを配置することができます。下記の図は上の例と用いて'1.1' に幅を減少したもので、さらにコンパクトになったラベルを表示しています。

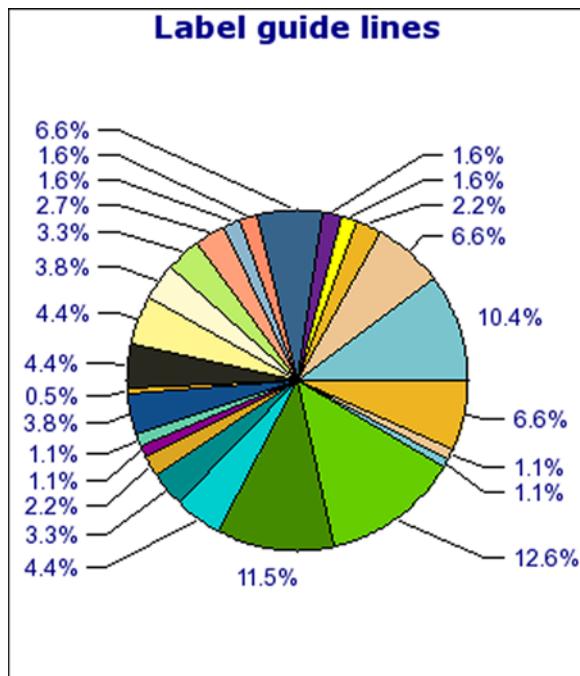


図 126: ラベル間の縦幅を調整する

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_pie.php");

$data = array(19,12,4,3,3,12,3,3,5,6,7,8,8,1,7,2,2,4,6,8,2
1,23,2,2,12);

// Create the Pie Graph.
$graph = new PieGraph(300,350);

// Set A title for the plot
$graph->title->Set("Label guide lines");
$graph->title->SetFont(FF_VERDANA,FS_BOLD,12);
$graph->title->SetColor("darkblue");
$graph->legend->Pos(0.1,0.2);

// Create pie plot
$p1 = new PiePlot($data);
$p1->SetCenter(0.5,0.55);
$p1->SetSize(0.3);

// Enable and set policy for guide-
lines. Make labels line up vertically
$p1->SetGuideLines(true,false);
$p1->SetGuideLinesAdjust(1.1);

// Setup the labels
$p1->SetLabelType(PIE_VALUE_PER);
$p1->value->Show();
$p1->value->SetFont(FF_ARIAL,FS_NORMAL,9);
$p1->value->SetFormat("%2.1f%");

// Add and stroke
$graph->Add($p1);
$graph->Stroke();

?>

```

注意:ガイド ラインは 2D の円プロットでのみ使用できます。

8.2.2 3D 円プロットを作成する

これまでには 2D 円プロットを作成してきましたが、3D プロットを作成するのも難しくありません。3D 円プロットを作成する場合は、[PiePlot3D\(\)](#) オブジェクトを使用します。先ほどの円プロットを使用して、PiePlot() オブジェクトの代わりに PiePlot3D() オブジェクトを作成すると、以下の出力が得られます。

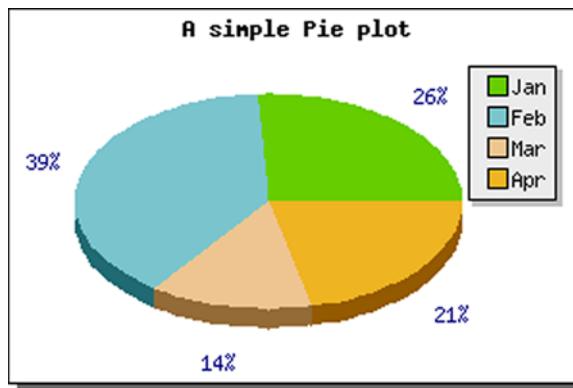


図 127: 3D 円プロットの例

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_pie.php");
include ("./jpgraph_pie3d.php");

$data = array(40,60,21,33);

$graph = new PieGraph(300,200,"auto");
$graph->SetShadow();

$graph->title->Set("A simple Pie plot");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

$p1 = new PiePlot3D($data);
$p1->SetSize(0.5);
$p1->SetCenter(0.45);
$p1->SetLegends($gDateLocale->GetShortMonth());

$graph->Add($p1);
$graph->Stroke();

?>
```

3D 円プロットは、通常の円プロットと同様の機能を有しています。また、[SetAngle\(\)](#) メソッドを使って角度を調節できます。以下に、角度を 20 度に設定した例を掲載します。

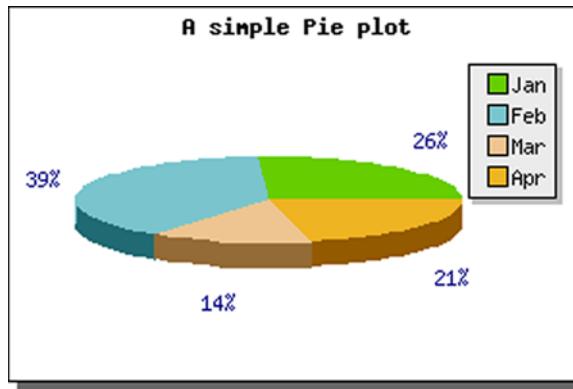


図 128: 視野角度を調節する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_pie.php");
include ("./jpgraph_pie3d.php");

$data = array(40,60,21,33);

$graph = new PieGraph(300,200,"auto");
$graph->SetShadow();

$graph->title->Set("A simple Pie plot");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

$p1 = new PiePlot3D($data);
$p1->SetAngle(20);
$p1->SetSize(0.5);
$p1->SetCenter(0.45);
$p1->SetLegends($gDateLocale->GetShortMonth());

$graph->Add($p1);
$graph->Stroke();

?>
```

8.2.3 円スライスを抜き出す

円プロットの一部のスライスを抜き出すことで、その部分を目立たせることができます。JpGraph では、2D および 3D 円プロットの両方で、任意の数のスライスを抜き出すことができます。

スライスを抜き出すには、[Explode\(\)](#) メソッドと[ExplodeSlice\(\)](#) メソッドを使用します。前者は複数個のスライスを抜き出すときに使用し、後者は 1 個のスライスだけを抜き出すときに便利な方法です。

例として、1 個のスライスをデフォルトの形式で抜き出すには、以下の通り記述します。

```
$pieplot->ExplodeSlice(1)
```

上記の指定では、2 番目のスライス（スライスは 0 番から順次番号付けされます）をデフォルトの距離だけ抜き出します。これにより、以下の結果が得られます。

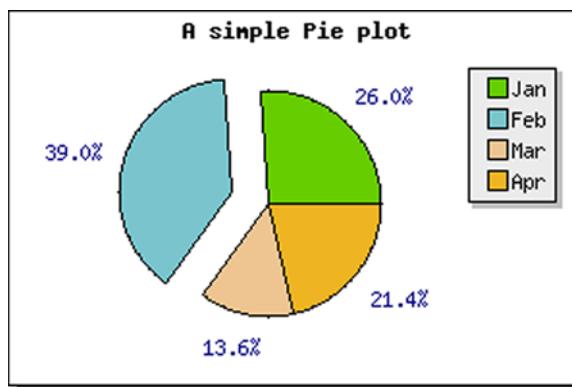


図 129: スライスを抜き出す

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_pie.php");
include ("../jpgraph_pie3d.php");

$data = array(40,60,21,33);

$graph = new PieGraph(300,200,"auto");
$graph->SetShadow();

$graph->title->Set("A simple Pie plot");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

$p1 = new PiePlot($data);
$p1->ExplodeSlice(1);
$p1->SetCenter(0.45);
$p1->SetLegends($gDateLocale->GetShortMonth());

$graph->Add($p1);
$graph->Stroke();

?>
```

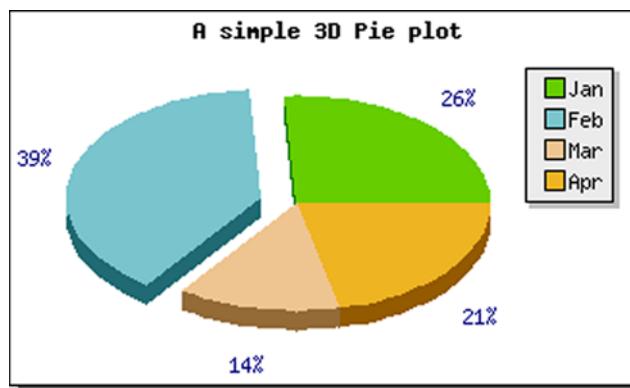


図 130: 3D スライスを抜き出す

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_pie.php");
include ("../jpgraph_pie3d.php");
```

```
$data = array(40,60,21,33);

$graph = new PieGraph(330,200,"auto");
$graph->SetShadow();

$graph->title->Set("A simple 3D Pie plot");
$graph->title->SetFont(FF_FONT1,FS_BOLD);
```

```
$p1 = new PiePlot3D($data);
$p1->ExplodeSlice(1);
$p1->SetCenter(0.45);
$p1->SetLegends($gDateLocale->GetShortMonth());

$graph->Add($p1);
$graph->Stroke();

?>
```

すべてのスライスを一度に抜き出すには、[PiePlot::ExplodeAll\(\)](#) メソッドを使用します。複数のスライスを抜き出すには、[PiePlot::Explode\(\)](#) メソッドを使用して、適切な配列を引数で手渡します。

8.2.4 円プロットのラベルを指定して調整する

デフォルトでは、各スライスの値は円プロットの外側に、パーセント値で表示されます。絶対値で表示したい場合は、[SetLabelType\(\)](#) メソッドを使用します。以下に例を掲載します。

```
$pieplot->SetLabelType("PIE_VALUE_ABS");
```

さらに、printf() 形式で書式を設定する ([SetFormat\(\)](#) メソッド)、あるいはコールバック関数を使用する ([SetFormatCallback\(\)](#) メソッド) ことで、より高度な書式設定が可能です。

また、ラベルの位置は[PiePlot::SetLabelPos\(\)](#) メソッドで調節できます。このメソッドの引数には、角度の割合か auto のいずれかを指定します。後者の場合は、JpGraph は自動的に最前の場所を判定し、以下のような出力を行います。

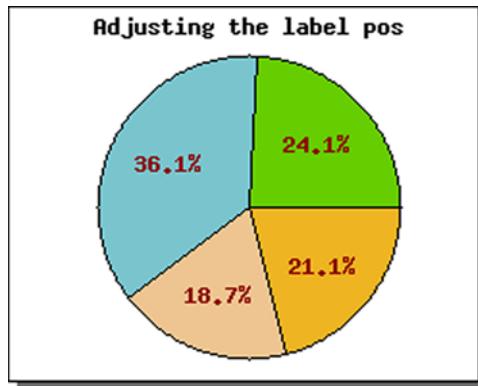


図 131: スライスのラベルの位置を調節する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_pie.php");

$data = array(40,60,31,35);

// A new pie graph
$graph = new PieGraph(250,200,"auto");
$graph->SetShadow();

// Title setup
$graph->title->Set("Adjusting the label pos");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Setup the pie plot
$p1 = new PiePlot($data);
```

```
// Adjust size and position of plot
$p1->SetSize(0.4);
$p1->SetCenter(0.5,0.52);

// Setup slice labels and move them into the plot
$p1->value->SetFont(FF_FONT1,FS_BOLD);
$p1->value->SetColor("darkred");
$p1->SetLabelPos(0.6);

// Finally add the plot
$graph->Add($p1);

// ... and stroke it
$graph->Stroke();

?>
```

さらには、手動で各ラベルを別々に指定することもできます。これを行うには、[PiePlot::SetLabels\(\)](#) メソッドを使用します。このメソッドを使用することで、各ラベルのテキスト文字列を指定できます。また、数値に対しては printf() 形式の書式も設定できます。手渡す数値は[SetLabelType\(\)](#) メソッドの指定により、絶対値、あるいはパーセント値になります。

また、SetLabels() メソッドでは第 2 引数としてラベルの位置を指定できます。これは、SetLabelPos() メソッドの代わりとして使用できます。引数が省略された場合、デフォルトで auto が使用されます。

注意:ラベルの配置は円プロットの内側か外側かで動作が異なります。円プロットの内側の場合は、文字列は指定した角度でスライスの中央に配置されます。円プロットの外側の場合は、角度に依存し、円プロットの上にラベルの文字列がかからないようにします。

8.2.5 円プロットのスライスに色とテーマを指定する

他に可能な変更としては、スライスのカラーを変更することが挙げられます。これを行うには 2 通りの方法があります。1 つはそれぞれのスライスのカラーを[SetSliceColors\(\)](#) メソッドで手動で変更することです。もしスライス数よりも少ない数のカラーを指定した場合は、カラーのローテーションが行われます。

もう 1 つの方法は、定義済みのテーマを使用する方法です。テーマとは、あらかじめ定義されたカラーの集合で、スライスのカラーが全て決められています。テーマを使用するには、[SetTheme\(\)](#) メソッドを使用します。現在、以下のテーマが使用できます。

- earth テーマ
- pastel テーマ
- sand テーマ
- water テーマ

テーマの例を下に紹介します。



図 132



図 133

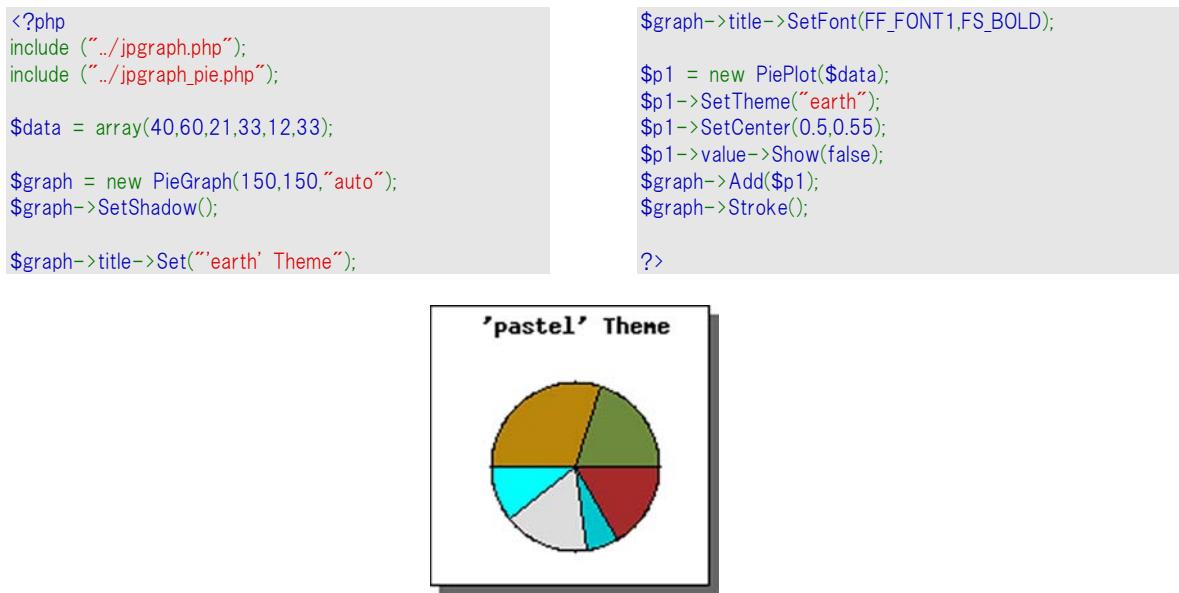


図 134

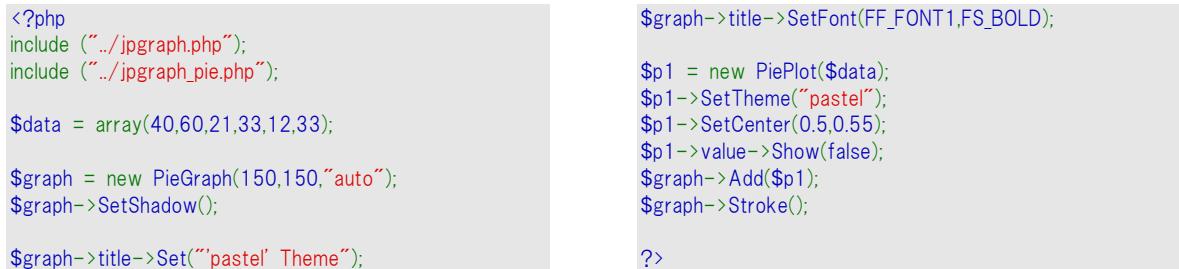




図 135

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_pie.php");

$data = array(40,60,21,33,12,33);

$graph = new PieGraph(150,150,"auto");
$graph->SetShadow();

$graph->title->Set("water' Theme");
```

```
$graph->title->SetFont(FF_FONT1,FS_BOLD);

$p1 = new PiePlot($data);
$p1->SetTheme("water");
$p1->SetCenter(0.5,0.55);
$p1->value->Show(false);
$graph->Add($p1);
$graph->Stroke();

?>
```

また、各スライスの枠線を取り外したり、カラーを変更したりすることもできます。これには、[ShowBorder\(\)](#) メソッドを使用します。

8.2.6 スライスに影を表示させる

各スライスに影を加えることで、グラフをよりきれいに表示することができます。影を加えるには、[PiePlot::SetShadow\(\)](#) メソッドを使用します。それぞれのスライスを抜き出して、影を追加すると、下記のように表示されます。

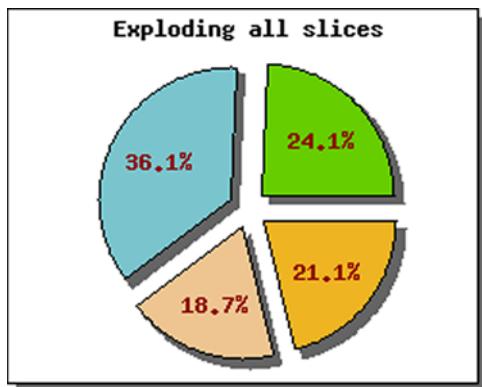


図 136: 影をスライスに追加する

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_pie.php");

$data = array(40,60,31,35);

// A new pie graph
$graph = new PieGraph(250,200,"auto");
$graph->SetShadow();

// Title setup
```

```
$graph->title->Set("Exploding all slices");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// Setup the pie plot
$p1 = new PiePlot($data);

// Adjust size and position of plot
$p1->SetSize(0.35);
$p1->SetCenter(0.5,0.52);

// Setup slice labels and move them into the plot
```

```
$p1->value->SetFont(FF_FONT1,FS_BOLD);
$p1->value->SetColor("darkred");
$p1->SetLabelPos(0.65);

// Explode all slices
$p1->ExplodeAll(10);

// Add drop shadow
$p1->SetShadow();
```

```
// Finally add the plot
$graph->Add($p1);

// ... and stroke it
$graph->Stroke();

?>
```

8.2.7 2D 円プロットに中心円を表示させる

2D 円プロットには、2 種類の形状があります。一般的な円プロットは [PiePlot](#) クラスで作成します。もうひとつは、[PiePlotC](#) クラスで作成します。

PiePlotC クラスにより円プロットを作成した場合、中心に同心円が描画されます。以下に、その例を掲載します。

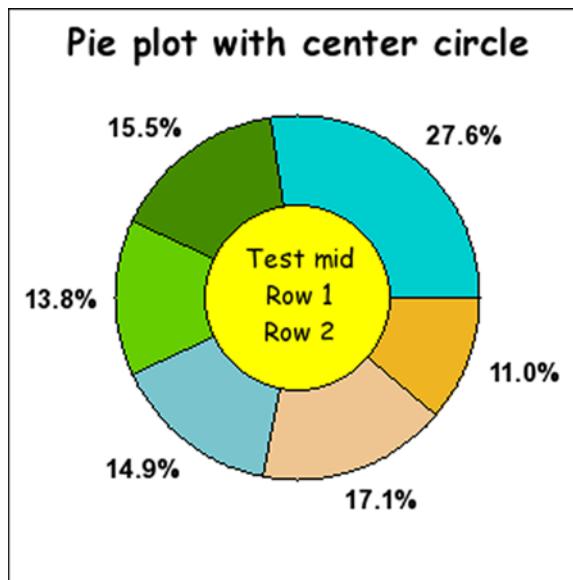


図 137: 中心円が描かれた円プロットの例

```
<?php
// $id
// Example of pie with center circle
include ("../jpgraph.php");
include ("../jpgraph_pie.php");

// Some data
$data = array(50,28,25,27,31,20);

// A new pie graph
$graph = new PieGraph(300,300,'auto');

// Setup title
$graph->title->Set("Pie plot with center circle");
$graph->title->SetFont(FF_COMIC,FS_BOLD,14);
$graph->title-
>SetMargin(8); // Add a little bit more margin from th
e top

// Create the pie plot
$p1 = new PiePlotC($data);
```

```
// Set size of pie
$p1->SetSize(0.32);

// Label font and color setup
$p1->value->SetFont(FF_ARIAL,FS_BOLD,10);
$p1->value->SetColor('black');

// Setup the title on the center circle
$p1->midtitle->Set("Test mid\nRow 1\nRow 2");
$p1->midtitle->SetFont(FF_COMIC,FS_NORMAL,10);

// Set color for mid circle
$p1->SetMidColor('yellow');

// Use percentage values in the legends values (This i
s also the default)
$p1->SetLabelType(PIE_VALUE_PER);

// Add plot to pie graph
$graph->Add($p1);

// ... and send the image on its merry way to the bro
```

```
wser
$graph->Stroke();
```

?>

PiePlotC クラスは通常の円プロットのクラスを拡張した形で実装されているため、PiePlot クラスのすべての機能を使用できます。

また、中心円の形状を変更することもできます。この機能では、中心円のサイズや色、フォントを調整することができます。これらの機能は以下の関数で使用することができます。

PiePlotC::SetMidColor()	円の塗りつぶしカラーを設定する。
PiePlotC::SetMidSize()	円のサイズ(半径比)を設定する。
PiePlotC::SetMidTitle()	タイトル文字列を設定する。複数行を指定可能。
PiePlotC::SetMid()	すべてのパラメータをまとめて変更できる。

通常の円プロットのイメージ マップ機能に加え、中心円にもクリック可能領域をつけることができます。

[PiePlotC::SetMidCSIM\(\)](#) メソッドで、対象を指定できます。

次のサンプルでは以下の特徴を備えたグラフを表示しています。

- 円グラフの周りにあるフレームが非表示
- すべてのスライスを抜き出し
- 影を各スライスと中心円に追加
- 複数行のラベルの指定
- タイトルのフォントを変更

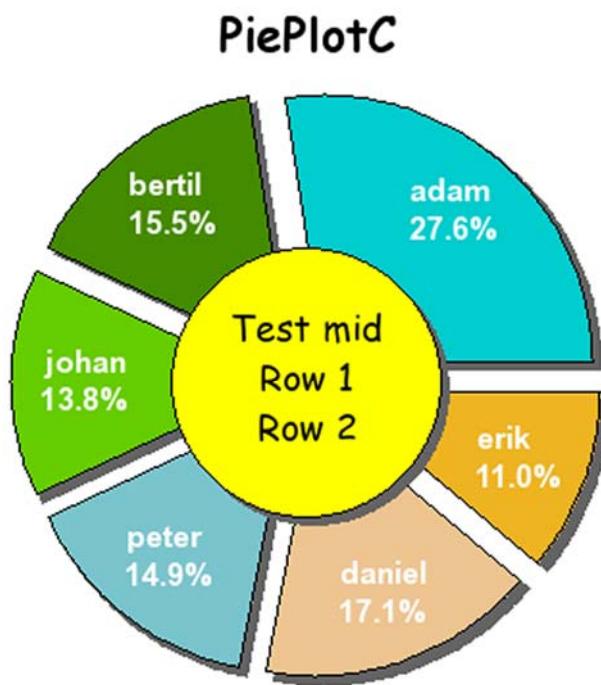


図 138: PiePlotC クラスを使った円プロット

```

<?php
// $Id: piecex2.php,v 1.3.2.1 2003/08/19 20:40:12 aditus Exp $
// Example of pie with center circle
include ("../jpgraph.php");
include ("../jpgraph_pie.php");

// Some data
$data = array(50,28,25,27,31,20);

// A new pie graph
$graph = new PieGraph(400,400,'auto');

// Don't display the border
$graph->SetFrame(false);

// Uncomment this line to add a drop shadow to the border
// $graph->SetShadow();

// Setup title
$graph->title->Set("PiePlotC");
$graph->title->SetFont(FF_COMIC,FS_BOLD,18);
$graph->title-
>SetMargin(8); // Add a little bit more margin from the top

// Create the pie plot
$p1 = new PiePlotC($data);

// Set size of pie
$p1->SetSize(0.35);

// Label font and color setup
$p1->value->SetFont(FF_ARIAL,FS_BOLD,12);
$p1->value->SetColor('white');

$p1->value->Show();

// Setup the title on the center circle
$p1->midtitle->Set("Test mid\nRow 1\nRow 2");
$p1->midtitle->SetFont(FF_COMIC,FS_NORMAL,14);

// Set color for mid circle
$p1->SetMidColor('yellow');

// Use percentage values in the legends values (This is also the default)
$p1->SetLabelType(PIE_VALUE_PER);

// The label array values may have printf() formatting in them. The argument to the
// format string will be the value of the slice (either the percentage or absolute
// depending on what was specified in the SetLabelType() above.
$lbl = array("adam\n%.1f%%","bertil\n%.1f%%","johan\n%.1f%%",
            "peter\n%.1f%%","daniel\n%.1f%%","erik\n%.1f%%");
$p1->SetLabels($lbl);

// Uncomment this line to remove the borders around the slices
// $p1->ShowBorder(false);

// Add drop shadow to slices
$p1->SetShadow();

// Explode all slices 15 pixels
$p1->ExplodeAll(15);

// Add plot to pie graph
$graph->Add($p1);

// .. and send the image on its way to the browser
$graph->Stroke();

?>

```

8.3 極プロット

8.3.1 概要

極プロットの各データ ポイントは、角度と半径から表現されます。極プロットは、指定した色で塗りつぶすこともできます。さらに、各ポイントにはマーカー（ライン プロットや散布プロットと同じもの）を表示させることもできます。

半径のスケールには、線形あるいは対数形式のものが使用できます。

極プロットを作成するには、[PolarGraph::PolarGraph\(\)](#) オブジェクトのインスタンスを作成します。極プロットは、通常の XY グラフと同じ性質を持っています。たとえば、背景画像やグラデーション カラー、タブ形式のタイトルなどが使用できます。

極プロットには、2 種類の基本的な形式があります。360 度のグラフ、あるいは 180 度のグラフです。それらの例を以下に掲載します。

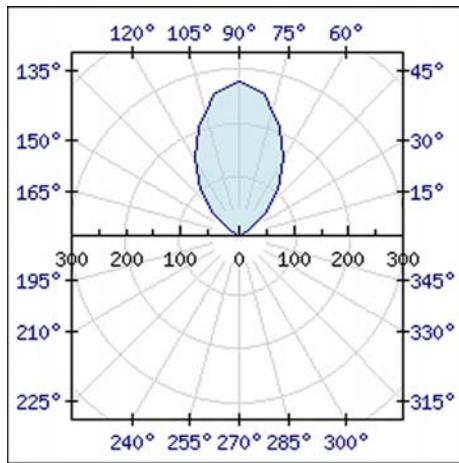


図 139: 360 度の極プロット

```
<?php
// A simple Polar graph, example 0

include "../jpgraph.php";
include "../jpgraph_polar.php";

$data = array(0,1,10,2,30,25,40,60,
      50,110,60,160,70,210,75,230,80,260,
      85,270,90,280,
      95,270,100,260,105,230,
      110,210,120,160,130,110,140,60,
      150,25,170,2,180,1);
```

```
$graph = new PolarGraph(250,250);
$graph->SetScale('lin');
$graph->SetMargin(35,35,25,25);

$p = new PolarPlot($data);
$p->SetFillColor('lightblue@0.5');
$graph->Add($p);

$graph->Stroke();

?>
```

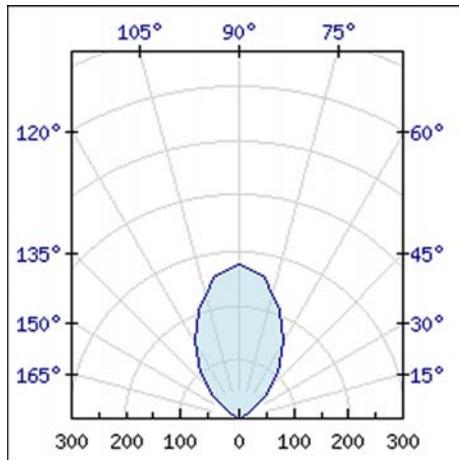


図 140: 180 度の極プロット

```
<?php
// A simple Polar graph, example 0

include "../jpgraph.php";
include "../jpgraph_polar.php";

$data = array(0,1,10,2,30,25,40,60,
      50,110,60,160,70,210,75,230,80,260,
      85,270,90,280,
      95,270,100,260,105,230,
      110,210,120,160,130,110,140,60,
      150,25,170,2,180,1);
```

```
$graph = new PolarGraph(250,250);
$graph->SetScale('lin');
$graph->SetMargin(35,35,25,25);
$graph->SetType(POLAR_180);

$p = new PolarPlot($data);
$p->SetFillColor('lightblue@0.5');
$graph->Add($p);

$graph->Stroke();

?>
```

8.3.2 半径スケールを調整する

半径軸は、リニアあるいは対数スケールで表示されます。スケールの指定は、[PolarGraph::SetScale\(\)](#) メソッドを使用します。以下のサンプルでは、リニア スケールと対数 スケールでプロットを行っています。

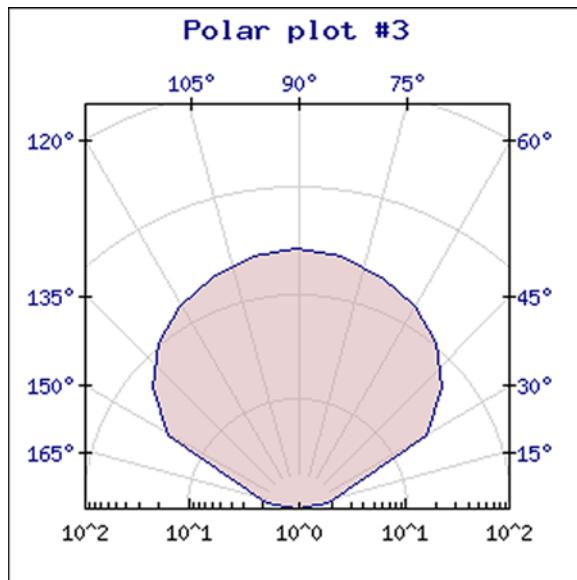


図 141: 対数スケールを使用した場合

```
<?php
// A simple Polar graph, example 2

include "../jpgraph.php";
include "../jpgraph_polar.php";

$data = array(0,1,10,2,30,25,40,60,
              50,110,60,160,70,210,75,230,80,260,
              85,270,90,280,
              95,270,100,260,105,230,
              110,210,120,160,130,110,140,60,
              150,25,170,2,180,1);

$graph = new PolarGraph(300,300);
$graph->SetScale('log',100);

$graph->SetType(POLAR_180);
$graph->title->Set('Polar plot #3');
$graph->title->SetFont(FF_FONT2,FS_BOLD);
$graph->title->SetColor('navy');

$graph->axis->ShowGrid(true,false);
$p = new PolarPlot($data);
$p->SetFillColor('lightred@0.5');

$graph->Add($p);
$graph->Stroke();
?>
```

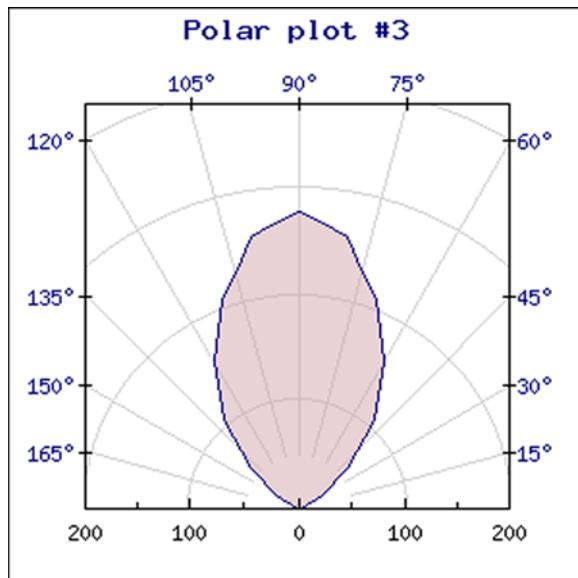


図 142: リニア スケールを使用した場合

```

<?php
// A simple Polar graph, example 2

include "../jpgraph.php";
include "../jpgraph_polar.php";

$data = array(0.1,10.2,30.25,40.60,
              50.110.60,160.70,210.75,230.80,260,
              85.270.90,280,
              95.270.100,260,105,230,
              110,210,120,160,130,110,140,60,
              150,25,170,2,180,1);

$graph = new PolarGraph(300,300);
$graph->SetScale('lin',200);
$graph->SetType(POLAR_180);

$graph->title->Set('Polar plot #3');
$graph->title->SetFont(FF_FONT2,FS_BOLD);
$graph->title->SetColor('navy');

$graph->axis->ShowGrid(true,false);

$p = new PolarPlot($data);
$p->SetFillColor('lightred@0.5');

$graph->Add($p);

$graph->Stroke();

?>

```

スケールの最大値が異なっている事に注意してください。

デフォルトでは、データに応じてスケールが自動で調整されます。また、`SetScale()` メソッドに引数を渡すことで、手動でスケールを指定することもできます。スケールの指定で XY 形式のグラフと異なる点は、極プロットでは最大値だけを指定する事です。極プロットの場合は、最小値は必ず 0 になります。そして、対数スケールの場合は、10 の対数値 (1, 0.1, 0.001 など) になります。

プロットはプロット エリアに表示される際にクリッピングされるため、最大値よりも小さなスケールを指定した場合はプロット エリアの外部に表示されます。

8.3.3 グリッド ラインを調整する

極プロットではグリッド ラインを指定することもできます。調節できるグリッド ラインには 3 種類あります。それぞれ、半径の小さなグリッド ライン、そして半径の大きなグリッド ラインと、角度のグリッド ラインです。

[PolarAxis::ShowGrid\(\)](#) メソッドを呼ぶことで、グリッド ラインを選択できます。以下の 2 種類の例では、グリッド ラインを調節した対数プロットを表示しています。

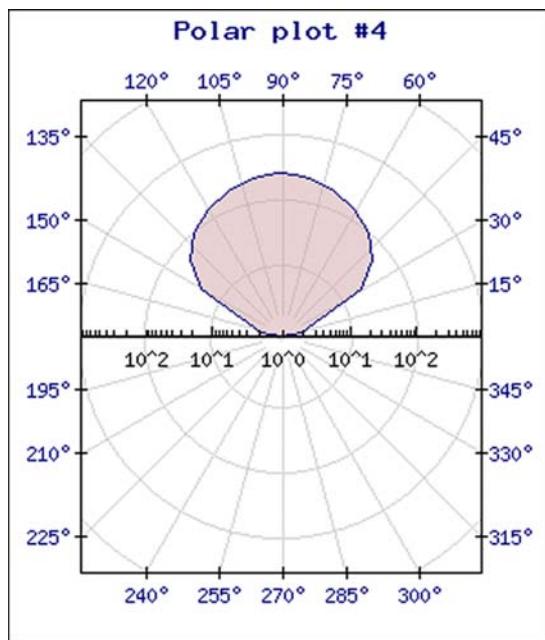


図 143: グリッド ラインを調節した極プロット

```
<?php
// A simple Polar graph, example 2

include "../jpgraph.php";
include "../jpgraph_polar.php";

$data = array(0,1,10,2,30,25,40,60,
50,110,60,160,70,210,75,230,80,260,
85,270,90,280,
95,270,100,260,105,230,
110,210,120,160,130,110,140,60,
150,25,170,2,180,1);

$graph = new PolarGraph(300,350);
$graph->SetScale('log');

$graph->title->Set('Polar plot #4');
$graph->title->SetFont(FF_FONT2,FS_BOLD);
$graph->title->SetColor('navy');

// Hide last labels on the Radius axis
// They intersect with the box otherwise
$graph->axis->HideLastTickLabel();

$p = new PolarPlot($data);
$p->SetFillColor('lightred@0.5');

$graph->Add($p);

$graph->Stroke();

?>
```

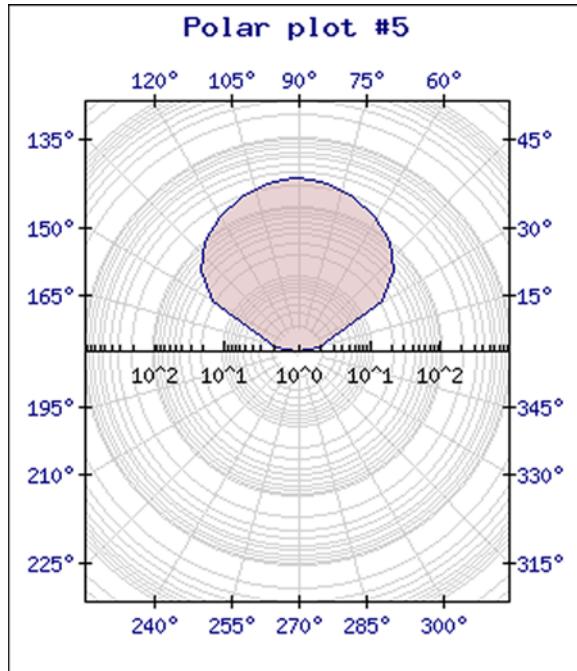


図 144: グリッド ラインを調節した極プロット

```
<?php
// A simple Polar graph, example 2

include "../jpgraph.php";
include "../jpgraph_polar.php";

$data = array(0,1,10,2,30,25,40,60,
              50,110,60,160,70,210,75,230,80,260,
              85,270,90,280,
              95,270,100,260,105,230,
              110,210,120,160,130,110,140,60,
              150,25,170,2,180,1);

$graph = new PolarGraph(300,350);
$graph->SetScale('log');

// Show both major and minor grid lines
$graph->axis->ShowGrid(true,true);

$graph->title->Set('Polar plot #5');
$graph->title->SetFont(FF_FONT2,FS_BOLD);
$graph->title->SetColor('navy');

// Hide last labels on the Radius axis
// They intersect with the box otherwise
$graph->axis->HideLastTickLabel();

$p = new PolarPlot($data);
$p->SetFillColor('lightred@0.5');

$graph->Add($p);

$graph->Stroke();

?>
```

グリッド ラインのカラーを指定するには、[PolarAxis::SetGridColor\(\)](#) メソッドを呼び出します。

角度グリッド ラインの場合は、各グリッド ラインの角度を [PolarAxis::SetAngleStep\(\)](#) メソッドで調節します。この角度はデフォルトでは 15 度に設定されています。

8.3.4 ラベルのフォントを設定する

角度や半径のラベルには、それぞれ異なったフォントやカラーを指定することができます。半径のフォントは、[PolarAxis::SetFont\(\)](#) メソッドで指定できます。また、角度のフォントは [PolarAxis::SetAngleFont\(\)](#) メソッドで指定できます。

カラーの調整は、[PolarAxis::SetColor\(\)](#) メソッドを使用します。

以下の例では、ラベルにカラーを指定しています。また、タブ形式のタイトルや軸のタイトルにも指定してあります。また、この例ではプロットの端のフレームを非表示にしています。

Polar plot #9

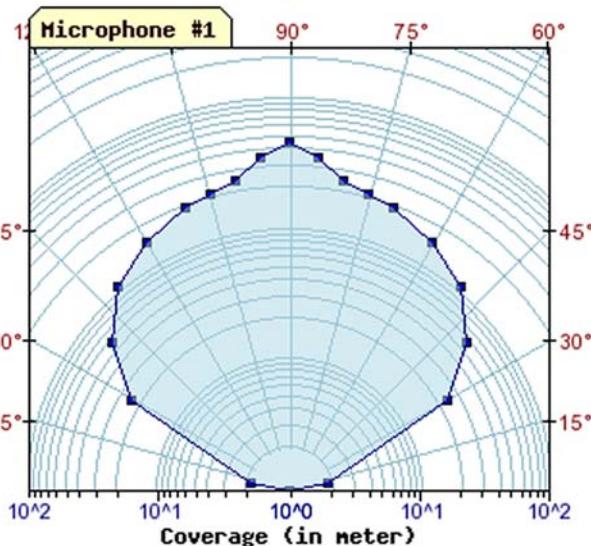


図 145: ラベルにカラーを指定する例

```
<?php
// A simple Polar graph.

include "../jpgraph.php";
include "../jpgraph_polar.php";

$data = array(0,1,10,2,30,25,40,60,
              50,110,60,160,70,210,75,230,80,260,85,370,
              90,480,
              95,370,100,260,105,230,
              110,210,120,160,130,110,140,60,
              150,25,170,2,180,1);

$graph = new PolarGraph(350,320);
$graph->SetScale('log',100);
$graph->SetType(POLAR_180);

// Hide frame around graph (by setting width=0)
$graph->SetFrame(true,'white',1);

// Show both major and minor grid lines
$graph->axis->ShowGrid(true,true);

// Set color for gradient lines
$graph->axis-
>SetGridColor('lightblue:0.9','lightblue:0.9','lightblue:0.9');

// Set label and axis colors
$graph->axis->SetColor('black','navy','darkred');

// Draw the ticks on the bottom side of the radius axis
$graph->axis->SetTickSide(SIDE_DOWN);

// Increase the margin for the labels since we changed
// the
// side of the ticks.
$graph->axis->SetLabelMargin(6);

// Change fonts
$graph->axis->SetFont(FF_ARIAL,FS_NORMAL,8);
$graph->axis->SetAngleFont(FF_ARIAL,FS_NORMAL,8);

// Setup axis title
$graph->axis->SetTitle('Coverage (in meter)', 'middle');
$graph->axis->title->SetFont(FF_FONT1,FS_BOLD);

// Setup graph title
$graph->title->Set('Polar plot #9');
$graph->title->SetFont(FF_ARIAL,FS_BOLD,16);
$graph->title->SetColor('navy');

// Setup tab title
$graph->tabtitle->Set('Microphone #1');
$graph->tabtitle->SetColor('brown:0.5','lightyellow');

$p = new PolarPlot($data);
$p->SetFillColor('lightblue@0.5');
$p->mark->SetType(MARK_SQUARE);

$graph->Add($p);

$graph->Stroke();

?>
```

8.3.5 ラベルを調整する

先ほどの例では、角度ラベルの後ろに角度を表すマーク「°」が表示されます。角度マークの表示・非表示を切り替えたい場合は、[PolarAxis::SetAngleDegreeMark\(\)](#) メソッドを使用します。

角度ラベルの場合は、書式文字列やコールバック関数なども使用できます。

また、360 度のプロットを使用している場合、最後のラベルがプロットエリアの周りのプロットボックスと重なってしまうため、最後のラベルを非表示にすることもできます。最後のラベルを非表示にするには、[Axis::HideLastTickLabel\(\)](#) メソッドを呼び出します。

8.3.6 イメージ マップを使用する

極プロットの場合は、各マーカーがイメージ マップのホット スポット(クリック可能な箇所)になります。対象の URL は、[SetCSIMTargets\(\)](#) メソッドを使って指定できます。

```
// イメージマップに使用される URL を指定する
$targets = array("#1", "#2", .....);
$polarplot= new PolarPlot($data);
$polarplot->mark->SetType(MARK_SQUARE);
$polarplot->SetCSIMTargets(targets);
$graph->Add($polarplot);
$graph->StrokeCSIM();
```

8.3.7 詳細に設定した極プロットの例

最終的な例として、360 度の極プロットに、正方形のマーカーとグラデーションの背景、プロットの凡例を追加したものを掲載します。

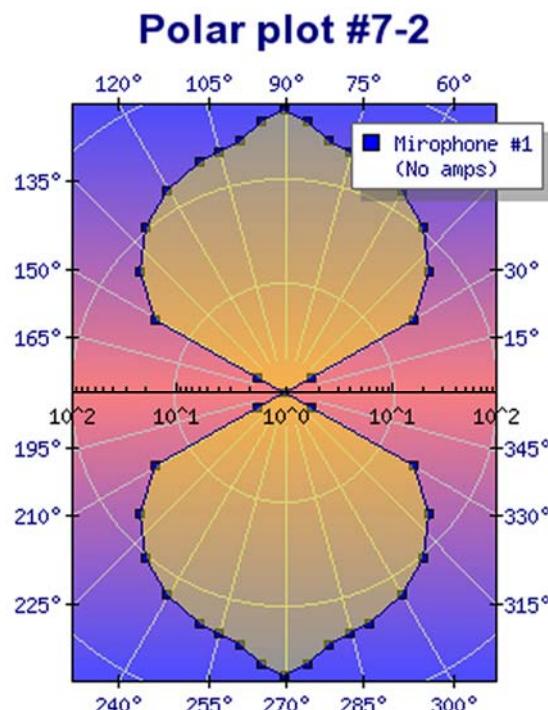


図 146: グラデーションの背景と凡例を持つ極プロットの例

```

<?php
// A simple Polar graph,
include "../jpgraph.php";
include "../jpgraph_polar.php";

$data = array(0,1,30,2,30,25,40,60,
              50,110,60,160,70,210,75,230,80,260,85,370,
              90,480,
              95,370,100,260,105,230,
              110,210,120,160,130,110,140,60,
              150,25,150,2,180,1);

$n = count($data);
for($i=0; $i < $n; $i+=2 ) {
    $data[$n+$i] = 360-$data[$i];
    $data[$n+$i+1] = $data[$i+1];
}

graph = new PolarGraph(300,400);
$graph->SetScale('log',100);
$graph->SetType(POLAR_360);
$graph->SetPlotSize(220,300);

// Hide frame around graph (by setting width=0)
$graph->SetFrame(true,'white',1);

$graph->SetBackgroundGradient('blue:1.3','brown:1.4',GRAD_MIDHOR_GRAD_PLOT);

// Set color for gradient lines
$graph->axis->SetGridColor('gray','gray','gray');

$graph->title->Set('Polar plot #7-2');
$graph->title->SetFont(FF_ARIAL,FS_BOLD,16);
$graph->title->SetColor('navy');

// Adjust legend box position and color
$graph->legend->SetColor('navy','darkgray');
$graph->legend->SetFillColor('white');
$graph->legend->SetShadow('darkgray@0.5',5);

$p = new PolarPlot($data);
$p->SetFillColor('yellow@0.6');
$p->mark->SetType(MARK_SQUARE);
$p->SetLegend("Mirophone #1\nNo amps");

$graph->Add($p);

$graph->Stroke();

?>

```

9 ガントチャート

9.1 ガントチャートの使用目的

皮肉的な理由:あなたのプロジェクトがなぜ期限を過ぎていて、予算額を超過しているかを説明するため。

実用的理由:私たちの管理維持や、何をしなければならないかを知るため。

一般的理由:プロジェクト問題を確認したり、問題範囲を明らかにするため。

基本的に、ガントチャートはアクティビティ数の状態(グループ化が可能)を時間軸と比較して表すのに使用されます。

9.2 JpGraph ガントチャートモジュールの機能

- 日付の自動・手動スケーリング
- ラベルおよびアクティビティのCSIMフルサポート
- 柔軟なスケール調整で、月、週、日、時間、分のどれもがスケールとして使用可能。さらに、(MSプロジェクトとは違って)好きなだけ同時にスケールを表示できます。
- グループ化されたアクティビティの視覚化をサポート

- ガント チャートは、バーの数や使用されるスケールによって自動的にサイズを変更します。すなわち、グラフを作成するときはサイズを指定しなくてもいいということです。
- アクティビティ数は無制限に追加可能
- ISO:8601 に準拠して、プラットフォームの独立した週数計算をサポートします
- 様々なスケールや、日、週、月、年をサポートするスケールの組み合わせの表示が可能。それぞれのスケール ヘッダは、フォント、サイズ、カラー、背景などが全て設定可能です。
- スケールのヘッドラインの日付フォーマットが指定可能
- アクティビティ間、スタートからスタート、スタートからエンド、エンドからスタート、エンドからエンド のコンストレインのビジュアル化
- スケールには“インテリジェント”グリッドがあります
- ユーザが指定したフォント、サイズ、カラーでタイトルやサブタイトルをサポートします
- アクティビティ バーには複数のパターンやカラーがあります
- アクティビティ バーには影付きにも設定できます。
- アクティビティ バーには、与えられたタスクがどのくらい完了したかを表すためのプログレス バーの表示ができます
- アクティビティ タイトルには個々のフォント、カラー、背景があります
- アクティビティ バーには説明文をつけることができます
- アクティビティ バーには、指定された右端付けと左端付けマーカーがあります
- バーの高さは、ピクセルの絶対値またはアクティビティのライン幅の割合(%)で指定します
- 様々なマークでマイルストンをサポートします
- テキストで縦マーカーのラインをサポートします
- 容易にローカライズ可能です
- CSIM (またはドリル ダウン グラフ) のフルサポート
- 非常に幅広い色が指定可能
- ... などなど

9.3 シンプルなガント チャート

ここでは、ガント チャートの作成例と、作成がいかに簡単かということを説明したいと思います。それでは、最もシンプルなガント チャートを作成してみましょう。このガントチャートには、“2001-11-01”から“2002-02-20”までの“Project”という名前のアクティビティを 1 つだけ登録します。

これを表示するコードは以下のとおりです。(すべてデフォルト値を使用しています)

```
(File: radarex1.php)
<?php
include ("../jpgraph.php");
include ("../jpgraph_radar.php");

// 自動的にサイズを設定する新しいグラフ
$graph = new GanttGraph(0,0,"auto");

// 0 行目の新しいアクティビティ
$activity = new GanttBar(0,"Project","2001-12-21","2002-01-20");
$graph->Add($activity);

// グラフを表示
$graph->Stroke();
?>
```

下の図 147 はこの出力結果です。

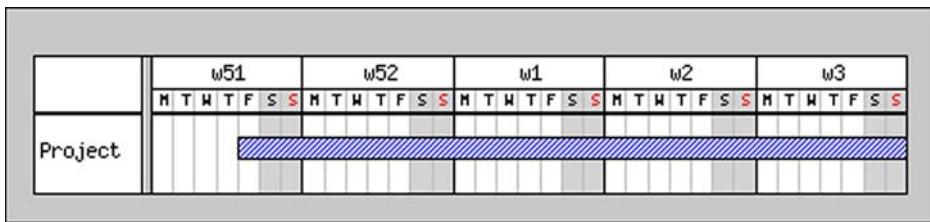


図 147: シンプルなガントチャート

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

// A new graph with automatic size
$graph = new GanttGraph(0,0,"auto");

// A new activity on row '0'
$activity = new GanttBar(0,"Project","2001-12-21","2002-01-20");
$graph->Add($activity);

// Display the Gantt chart
$graph->Stroke();
?>
```

上図の画像とコードに関して、いくつかの点に注意しましょう:

- まず、jpgraph.php と jpgraph_gantt.php の両方を常にインクルードする必要があります。
- 鉛直方向の位置(これについてはすぐに詳細を説明)、タイトル、スタートの日付、終了の日付、の最低4つのパラメータを用いてバーを指定します。
- 画像の大きさを指定しない場合、バーの最小と最大の日付を含む週全体を表示するように自動的に決定されます。
- デフォルトでは、週および日のスケールが表示されます。
- 週末の背景は、デフォルトでは薄いグレー色で表示されます。
- 日曜日は赤色で描かれます。
- 週は ISO8601 に準拠して番号付けされます。
- アクティビティバーは、デフォルトでは白地に青色のストライプとして表示されます。

それでは、このグラフを少しだけ面白いものにしてみましょう。最初にタイトルを加え、次に月スケールを加え、最後にバーの色を変更してみましょう。

それらすべては以下のコードで実行できます。

```
(File:ganttex01.php)
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetShadow();

// タイトルとサブタイトルを加える
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(Draft version)");

// 日、週、月のスケールを表示する
$graph->ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// 週番号のかわりに、
// 週スケールで週の最初の日付けを表示する
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);

// 週スケールで週の最初の日付けを表示する
```

```
$graph->scale->week->SetFont(FF_FONTO);

// 週スケールフォントをデフォルトより小さくする
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR2);

// 最初のアクティビティのバーを初期化する
// ($row,$title,$startdate,$enddate)
$activity?= new GanttBar(0,"Project","2001-12-
21","2002-01-20");

// アクティビティを赤色の背景、黄色の斜線にする
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// グラフにバーを加える
$graph->Add($activity);

// グラフを表示する
$graph->Stroke();
?>
```

下の図 148 はこのスクリプトの出力結果です。



図 148: タイトルや多くの色を使用してガントチャートを作成する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(Draft version)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_-
HMONTH);

// Instead of week number show the date for the first
// day in the week
// on the week scale
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
```

```
$graph->scale->week->SetFont(FF_FONTO);

// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR2);

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-
21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Finally add the bar to the graph
$graph->Add($activity);

// ... and display it
$graph->Stroke();
?>
```

上の例で、いくつかの注意点があります。

- 余白の大きさは、追加されたタイトルやサブタイトルに応じて自動的に調整されます。
- スケールヘッダの高さは、フォントを変更すると自動的に調整されます。
- スケールラベルのフォーマットの選択は高い柔軟性を持っています。例えば、月のヘッダーに西暦を 4 行で表示したい場合、上のコード内の定数 MONTHSTYLE_SHORTNAMEYEAR2 を MONTHSTYLE_SHORTNAMEYEAR4 へ変更するだけで可能です。
- ヘッダーのフォントや色や背景、サイズ等は全て自由に変更することができます。

これらは全て簡単に設定することができます。以下の文を加え、月スケール内で西暦を 4 行で表示し、ヘッダースタイルが紺色の背景に白字となるように設定してみましょう。

```
// 4 行の西暦と月の名前を
// 月スケール上に並べて表示させる
$graph->scale->month->SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetTextColor("white");
$graph->scale->month->SetBackgroundColor("blue");
```

この出力結果は下図 149 のようになります。

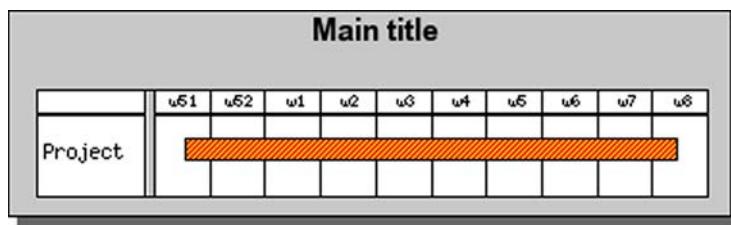


図 149:スケールヘッダーを調整する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("Main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);

// Show day, week and month scale
// $graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);
$graph->ShowHeaders(GANTT_HWEEK );

// Instead of week number show the date for the first
// day in the week
// on the week scale
$graph->scale->week->SetStyle(WEEKSTYLE_WNBR);

// Make the week scale font smaller than the default
$graph->scale->week->SetFont(FF_FONTO);
```

```
// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-
21","2002-02-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Finally add the bar to the graph
$graph->Add($activity);

// ... and display it
$graph->Stroke();
?>
```

9.4 ガント チャートの構造

ガント チャートは 4 つの領域から構成されています。

1. 左部にあるアクティビティ タイトルの列
2. 上部にあるスケール ヘッダー(最大4つのヘッダーが表示可能)
3. 全てのガント バーやマーカーが表示されるプロット エリア
4. タイトルなどが表示される余白部

ガント チャートは Jpgraph Graph()のプロパティを全て受け継ぐので、これまでと同様に、イメージをフォーマットするメソッドを利用することができます。例えば、Graph::SetShadow()を呼び出してイメージの周辺に影をつけたり、Graph::SetMarginColor()を使用して余白の色を設定したりすることができます。サポートされている機能の完全なリストについては、リファレンスドキュメントをご覧下さい。

ガント チャートを生成するためには、オブジェクトをガント チャートに `add` する必要があります。GanttChart::Add()メソッドを使用して、以下のオブジェクトをガント チャートに追加することができます。

- ガント バー(アクティビティの期間を示す)
- マイルストン(指定した日付につけるマーク)
- バーティカルライン(プロジェクトの進捗状況の評価を示すために使用する縦線)

これらのオブジェクトはすべて広範囲にわたって拡張することができます。色(塗りつぶし色やフレームの色の両方)、サイズ、タイトル、スタイルやパターンなどを指定することができます。これらのオブジェクトはすべて(私の考えでは)適切なデフォルト値が設定されるので、全てのパラメータを指定する必要はありません。しかし、細かな調整の必要があつたり自分の好みに合わないような場合は変更することができます。

9.5 ガント チャートを作成する

GanttChart()を呼び出して、新しいガント チャートを作成します。ガント チャートの書き方は、通常の JpGraph の場合と同様です。

```
function GanttGraph($aWidth,$aHeight,$aCacheName,$aTimeOut,$aInline)
```

ただ一つ違うことは、大きさのパラメータ(奥行きや高さ)に-1 を指定することができるということです。その場合、大きさは選択したスケールやフォントに応じて自動的に決められます。次の例で、新しいグラフを生成するいくつかの方法を見てみましょう。

- `$graph=new GanttGraph()`
グラフのサイズが自動的に決定されます。キャッシュは使用されず、グラフはインラインで作成されます。
- `$graph=new GanttGraph(-1,-1,"auto")`
グラフのサイズは自動的に決定され、キャッシュが使用されます(キャッシュファイル名はスクリプト名に基づいて付けられます)。また、キャッシュのタイムアウトは使用されず、グラフはインラインで作成されます。

- `$graph=newGanttGraph(450,-1,"auto",5)`
ほぼ前の例と同じですが、横幅が 450 ピクセルに固定され、キャッシュされたイメージは 5 分でタイムアウトされます。
- `$graph=newGanttGraph(-1,-1,"auto",5,false)`
イメージはインラインで作成されず、キャッシュされたイメージがタイムアウトした場合はそのキャッシュのみが更新され、それ以外の場合は何も起りません。

`GanttGraph()`は `Graph()`から全ての(ガントグラフに対して意味のある)メソッドを受け継いでいるので、通常のフレームの影や色などを指定することができます。

9.6 ガントチャート内のオブジェクトの位置を調整する

バー や マイルストーンには鉛直位置と水平位置の両方が必要となります。水平方向のスタート位置は “2001-06-23” のように日付で指定し、鉛直方向の位置は [0,1,2,3,...] のような数値で指定します。この縦方向の数値は、オブジェクトが位置すべき場所の上端からの位置を示しています。このことを理解するために、ある高さを持った多くの「目に見えない」水平のバンドを想像してください。鉛直方向の位置を 0 と指定すると、バーは 1 つ目のバンド内に配置され、また 3 と指定すると 4 番目のバンド内に配置されるでしょう。

関連するアクティビティをグループ化するためにバンドを配置する際に「ギャップ」を残すことはある程度仕方なく、おそらくは推奨される方法でもあります。例えば、1、2、3 の位置に 3 つのアクティビティまたはバーを持ち、さらにバンド 0,4,5 を空けたままで 6,7 の位置に別の 2 つのバーを置くことができます。

これらの「見えないバンド」のすべては、同じ高さを(等間隔でも)もっています。各バンドの高さは自動的に決定され、レイアウトのメソッド(`GanttChart::SetLayout()`で指定されます)と、バーおよびタイトルの個々の高さとの両方に依存します。その仕組みはとてもシンプルです。

- `layout=GANTT_FROMTOP` を使用する場合(これがデフォルトで最も一般的です)、その高さは最も高いガントバーの高さ(+余白)で設定されるでしょう。それぞれのバーの高さの計算は、存在している実際のバー や タイトル、および左右のマークのすべてを考慮します(これについては後述します)。”FROMTOP” の意味するところは、高さを明示的に指定した場合、バーはバンド 0 から順に加えられる、つまりは上端から加えられる、ということです。(グラフの高さが明示的に指定された場合、プロットエリアの下端が空白のまま残る可能性があります。)
- `layout=GANTT_EVEN` を使用する場合、バーはガントチャート内で可能な高さにわたくて均等に広がり、個々のバーの高さは考慮されません。自動サイズ調整を使用した場合は EVEN layout も使用できないということに注意してください。それは何の意味もなしません。EVEN レイアウトは、故意に非常に大きな画像を指定する場合や、バーを最大の高さで均等に配置したい場合などに使われます。

9.7 ガント バー

ガントチャートのオブジェクトのうち最もよく使用するのは、もちろんアクティビティバー(GanttBar())です。このオブジェクトは表示フォーマットにおいて非常に高い柔軟性を持っています。ガントバーの完全なコンストラクタは以下のようになります。

```
function GanttBar($aVPos,$aTitle,$aStart,$aEnd,$aCaption,$aHeight)
```

aVPos バーの鉛直位置 [0..n]

\$aTitle アクティビティのタイトル

\$aStart “2001-09-22”といった文字列として与えられるアクティビティのスタートの日付

\$aEnd 日付(文字列)あるいは継続時間(日数)として与えらるアクティビティの最後の日付。
例えば、”2001-10-15”も 20.5 のどちらも有効な入力値です。

\$aCaption バーの終端(右側)に表示される文字列(キャプション)

\$aHeight 0 から 1 の範囲の値で与えられるバーの高さ。この場合、バーが鉛直方向の位置の
どのくらいの割合を占めているかということを意味します。また、高さは 1 から 200 の
範囲のピクセルの絶対値で与えることも可能です。

9.7.1 鉛直位置を指定する

上述のように、鉛直方向の位置は「n」が任意の定数である[0..n]の範囲の数値として指定されます。
(実用的な目的では、ほとんどの場合 n < 100 です)

先の例を使用してこのパラメータを説明しましょう。[project]アクティビティの位置をポジション7へ
変更します。そこで、GanttBar()の呼び出しを以下のように変更します。

```
$activity = new GanttBar(7,"Project","2001-12-21","2002-02-20");
```

すると下図 150 のようなチャートが output されます。

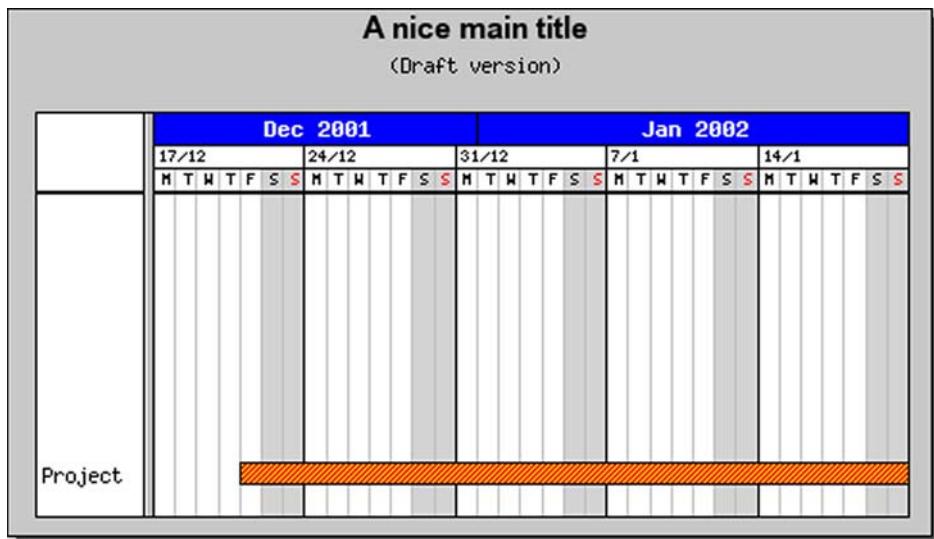


図 150: 垂直の位置を 7 へ変更する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(Draft version)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// Instead of week number show the date for the first
// day in the week
// on the week scale
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
$graph->scale->week->SetFont(FF_FONTO);

```

```

// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(7,"Project","2001-12-
21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Finally add the bar to the graph
$graph->Add($activity);

// ... and display it
$graph->Stroke();
?>

```

それぞれの位置(鉛直方向の位置)の高さは、バーの実際の高さに依存するということに注意してください。

9.7.2 バーの始点と終点を指定する

バーの始点は、日付文字列として指定します。そのフォーマットは、現在のロケールに依存します。有効な日付文字列の例は、次のようなものです。

- “2001-10-22”
- “22 October 2001”

- “22 Oct 2001”

これらの複数のフォーマットがサポートされてますが、数値だけを使用する日付を使うことをお勧めします。つまり、“2001-10-22”といった形式です。

終点を指定するには2つの方法があり、1つの方法は、スタートの日付を指定した場合と同じ方法で終了の日付を指定する方法です。もう一つの方法は、日数(小数も含む)でアクティビティの長さを指定するというものです。有効な終了日の例は、次のようなものです。

- “2001-11-15”
- “15 Nov 2001”
- 22,(22日の継続期間を指定)
- 22.7,(22.7日の継続期間を指定)

継続時間は、文字列ではなく、数値で指定します。

9.7.3 マイルストーン

マイルストーンはバーに似ていますが、終了日をもちません。というのは、マイルストーンはある一日のみに適用されるためです。マイルストーンは多くがアクティビティと同じ方法で作成されますが、代わりに MileStone()メソッドを使用します。

マイルストーンのコンストラクタは以下のとおりです。

```
function MileStone($aVPos,$aTitle,$aDate,$aCaption)
```

\$aVPos バーの鉛直位置 [0..n]

\$aTitle アクティビティのタイトル

\$aDate マイルストーンの日付

\$aCaption マイルストーンの右側に表示されるテキスト

有効なマイルストーンの例は次のようなものです。

```
$milestone = new MileStone(3,"Code complete","2001-12-01");
$milestone = new MileStone(3,"Code complete","2001-12-01","(2001-12-01)");
```

デフォルトでは、マイルストーンは塗りつぶしの「ダイヤモンド」形状として描かれます。これはオプションで修正することができます。実際の形状は、PlotMark()クラスのインスタンスの一つであるマイルストーンの「mark」プロパティを用いて指定します。(バープロットにおけるマークと同じクラス)

例えばマイルストーンの形状を三角形に変更するためには、以下のように SetType()メソッドを使用します。

```
$milestone->mark->SetType(MARK_DTRIANGLE)
```

これを実行し、これまでの例にマイルストーンを加えてみましょう。これにより下図 151 のような結果が得られます。

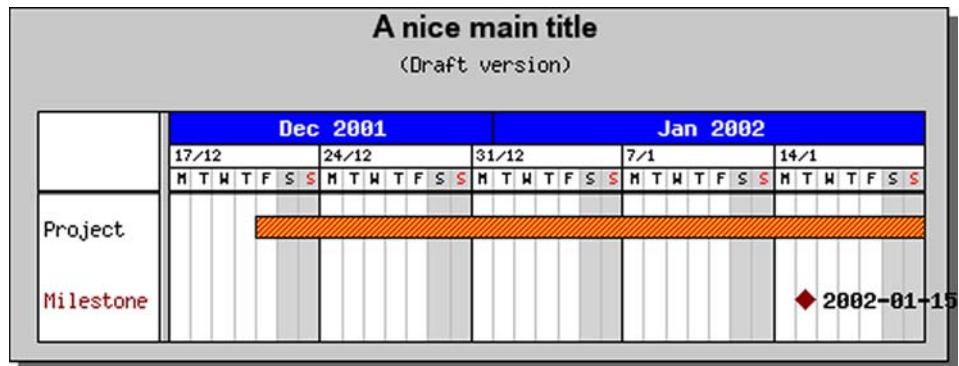


図 151: ガントチャートにマイルストーンを加える

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(Draft version)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// Instead of week number show the date for the first
// day in the week
// on the week scale
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
$graph->scale->week->SetFont(FF_FONTO);

// Use the short name of the month together with a
```

```
2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetTextColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-
21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Finally add the bar to the graph
$graph->Add($activity);

// Create a milestone
$milestone = new MileStone(2,"Milestone","2002-01-
15","2002-01-15");
$graph->Add($milestone);

// ... and display it
$graph->Stroke();
?>
```

デフォルトでは、マイルストーンのタイトル色は赤色であることに気づくでしょう。たとえばこれを太字の黒色に変更したいなら、以下のようにマイルストーンのタイトルプロパティで、SetColor()やSetFont()を実行します。

```
$milestone->title->SetFont(FF_FONT1,FF_BOLD);
$milestone->title->SetColor("black");
```

この結果は次のようになります。



図 152: マイルストーンタイトルの色とフォントを変更する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0, 0, "auto");
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL, FS_BOLD, 12);
$graph->subtitle->Set("(Draft version)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// Instead of week number show the date for the first day in the week
// on the week scale
$graph->scale->week->SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
$graph->scale->week->SetFont(FF_FONTO);

// Use the short name of the month together with a 2 digit year
// on the month scale
$graph->scale->month->SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0, "Project", "2001-12-21", "2002-01-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG, "yellow");
$activity->SetFillColor("red");

// Finally add the bar to the graph
$graph->Add($activity);

// Create a milestone
$milestone = new MileStone(2, "Milestone", "2002-01-15");
$milestone->title->SetColor("black");
$milestone->title->SetFont(FF_FONT1, FS_BOLD);
$graph->Add($milestone);

// ... and display it
$graph->Stroke();
?>

```

見出しを修正するためには、title プロパティのかわりに caption プロパティに使用すること以外は、全く同じことを行います。

```
$milestone->caption->SetFont(FF_FONT1,FS_BOLD);
$milestone->caption->SetColor("black");
```

バーのタイトルやキャプションプロパティに同じ方法で使用することによって、バータイトルや見出しを修正することもできます。

9.7.4 バーティカル ライン

ガントチャートに加える最後のオブジェクトは、簡単ですがとても有用なもので、プロットの縦全体に伸びる鉛直の直線です。これは、例えば、あるプロジェクトの進捗段階の遅れや誤差を示すためなどに使用されます。GanttVLine() を呼び出すことで、ラインオブジェクトを作成できます。

GanttVLine()の完全な構造は以下通りです。

```
function GanttVLine($aDate,$aTitle,$aColor,$aWeight,$aStyle)
```

\$aDate マイルストーンの日付

\$aTitle アクティビティのタイトルタイトルはラインの下に表示されます。

\$aColor ラインの色

\$aWeight ラインの太さ

\$aStyle 破線(dashed)や点線(dotted)などといったラインのスタイル

ラインの生成法は以下の通りです。

```
$vline = new GanttVLine("2001-12-24");
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline = new GanttVLine("2001-12-24","Phase 1","darkred");
$vline = new GanttVLine("2001-12-24","Phase 1","darkred",5);
$vline = new GanttVLine("2001-12-24","Phase 1","darkred",5,"dotted");
```

グラフにラインを付け加えるために、マイルストーンやバーの時と同様に GanttGraph::Add()を呼び出す必要があります。先の例にラインを加えることによってバーティカルラインの使用法を説明しましょう。

```
$vline = new GanttVLine("2001-12-24","Phase 1");
$graph->Add($vline);
```

この例は図 153 のようになります。

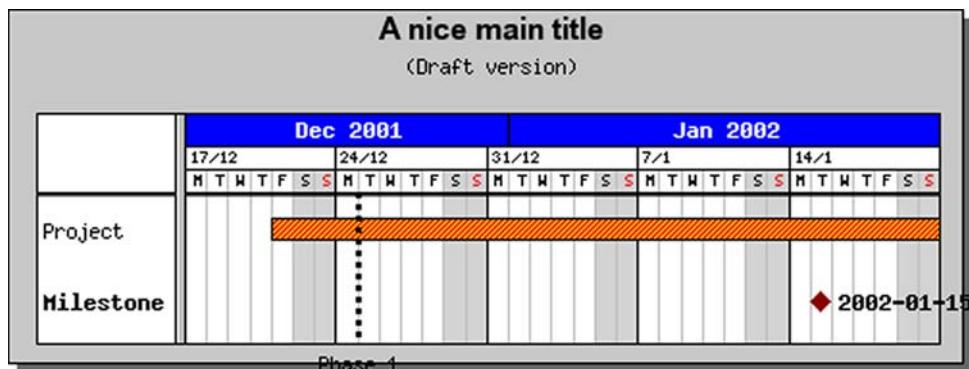


図 153: ガントチャートにタイトルつきバーティカルラインを加える

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(Draft version)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);
```

```
// Instead of week number show the date for the first
day in the week
// on the week scale
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
$graph->scale->week->SetFont(FF_FONTO);

// Use the short name of the month together with a
2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");
```

```
// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Finally add the bar to the graph
$graph->Add($activity);

// Create a milestone
```

```
$milestone = new MileStone(2,"Milestone","2002-01-15","2002-01-15");
$milestone->title->SetColor("black");
$milestone->title->SetFont(FF_FONT1,FS_BOLD);
$graph->Add($milestone);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$graph->Add($vline);

// ... and display it
$graph->Stroke();
?>
```

上の図から、デフォルトでは指定された日の始まりに、破線で描かれることができます。これは、もちろん変更可能で、線は指定した日のどこにでも描いたり並べたりできます。修正を行うには、線を引きたい場所に対応する一日の割合を指定する引数を用いて、SetDayOffset() メソッドを使用します。

例えば一日の真ん中に線を加えたい場合、次の文をこれまでのコードへ付け加えるだけです。

```
$vline->SetDayOffset(0.5);
```

その結果は以下のようになります。



図 154: 線の位置を変更する

```
<?php
include ("../jgraph.php");
include ("../jgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(Draft version)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// Instead of week number show the date for the first
// day in the week
// on the week scale
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
```

```
$graph->scale->week->SetFont(FF_FONTO);

// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetTextColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Finally add the bar to the graph
$graph->Add($activity);

// Create a milestone
$milestone = new MileStone(2,"Milestone","2002-01-
```

```

15","2002-01-15");
$milestone->title->SetColor("black");
$milestone->title->SetFont(FF_FONT1,FS_BOLD);
$graph->Add($milestone);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");

```

```

$vline->SetDayOffset(0.5);
$graph->Add($vline);

// ... and display it
$graph->Stroke();
?>

```

通常通り、適切なメソッド(SetFont(), SetColor())を実行して、フォントやサイズや色を修正できます。

9.7.5 ガント バーにマーカーを追加する

ガントバーの始点と終点の両方に様々なマーカーを簡単に追加できます。例えば、重要なマイルストーンなどを図示するための代わりの方法として、マーカーが使用されます。

左付き、右付きのマーカーは、'leftMark' プロパティと'rightMark' プロパティから指定します。それらは両方とも、マイルストーン(そしてラインプロットでも)に対しても使用される PlotMark クラスのインスタンスです。PlotMark クラスは、例えばダイヤモンド(マイルストーンではデフォルト)、円、もしくは塗りつぶしの円、四角、星印などのようないくつかの異なるスタイルをサポートします。完全なリストはリファレンスセクションを参照してください。

マーカーを説明するために、右付きのマーカーをこれまでの例に付け加えてみましょう。”M5”という白地のタイトルを持つ、赤色の塗りつぶしの円のスタイルを使用しましょう。これを実現させるために、次の行をこれまでの例に付け加えます。

```

$activity->rightMark->Show();
$activity->rightMark->title->Set("M5");
$activity->rightMark->SetType(MARK_FILLED CIRCLE);
$activity->rightMark->SetWidth(10);
$activity->rightMark->SetColor("red");
$activity->rightMark->SetFillColor("red");
$activity->rightMark->title->SetFont(FF_ARIAL,FS_BOLD,12);
$activity->rightMark->title->SetColor("white");

```

これはかなり長いコードに思えるかもしれません、これ以上複雑になることはありません。以下のイラストの例では、変更可能なものはデフォルトのフォント、フォント カラー、塗りつぶしカラー、フレーム カラー、マーカーの幅に変わります。実際に必要なのはマークを表示しタイトルを設定する最初の 2 行のみです。残りのプロパティはデフォルト値を使用すれば正常に表示されます。

結果の画像は、下図 155 に示されたとおりです。

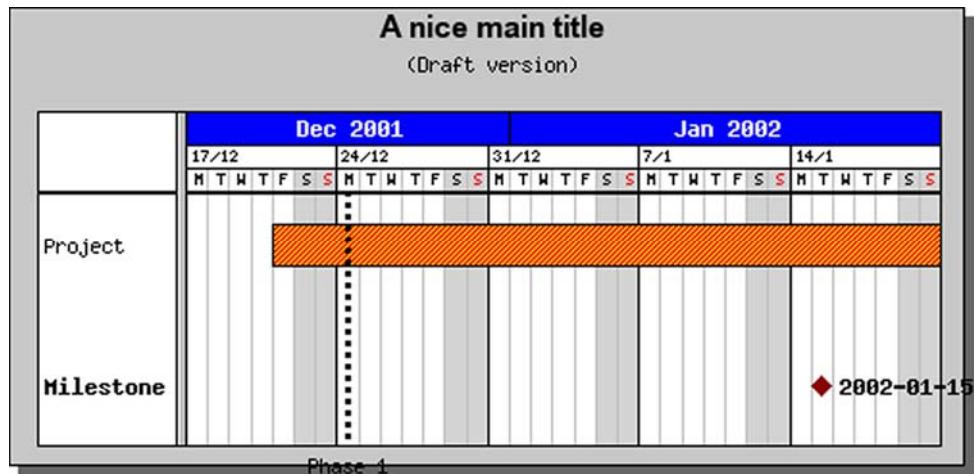


図 155:バーへ右つきマーカーを加える

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(Draft version)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// Instead of week number show the date for the first
// day in the week
// on the week scale
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
$graph->scale->week->SetFont(FF_FONTO);

// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetTextColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
// 
```

```

$activity = new GanttBar(0,"Project","2001-12-
21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Add a right marker
$activity->rightMark->Show();
$activity->rightMark->SetType(MARK_FILLED CIRCLE);
$activity->rightMark->SetWidth(13);
$activity->rightMark->SetColor("red");
$activity->rightMark->SetFillColor("red");
$activity->rightMark->title->Set("M5");
$activity->rightMark->title-
>SetFont(FF_ARIAL,FS_BOLD,12);
$activity->rightMark->title->SetColor("white");

// Finally add the bar to the graph
$graph->Add($activity);

// Create a milestone
$milestone = new MileStone(2,"Milestone","2002-01-
15","2002-01-15");
$milestone->title->SetColor("black");
$milestone->title->SetFont(FF_FONT1,FS_BOLD);
$graph->Add($milestone);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
$graph->Add($vline);

// ... and display it
$graph->Stroke();
?>

```

ここで、ある不思議な現象を解説したいと思います。これまでの2つの例を比較した場合、最後の例はその前の例よりも大きいことに気づかれるでしょう。それは、なぜでしょうか？

バーの高さが水平方向の間隔に比例して決められることを覚えているならば、その説明は簡単です。水平方向の間隔は、タイトルサイズや、ここで説明するマーカーサイズを含めた最も高いバーを基準としています。最小の高さがいま 10 ポイント(=マークの高さ)に基づいているので、水平方向の間隔が延長されました。バーの高さは同じ割合で設定されるので、最後の例が大きくなったのです。

この動作が好ましくない場合、次の関数を用いて、バーの高さを絶対サイズ(ここでは 8 ピクセル)に指定することができます。

```
$activity->SetHeight(8);
```

これにより下図 156 のような結果が得られます。

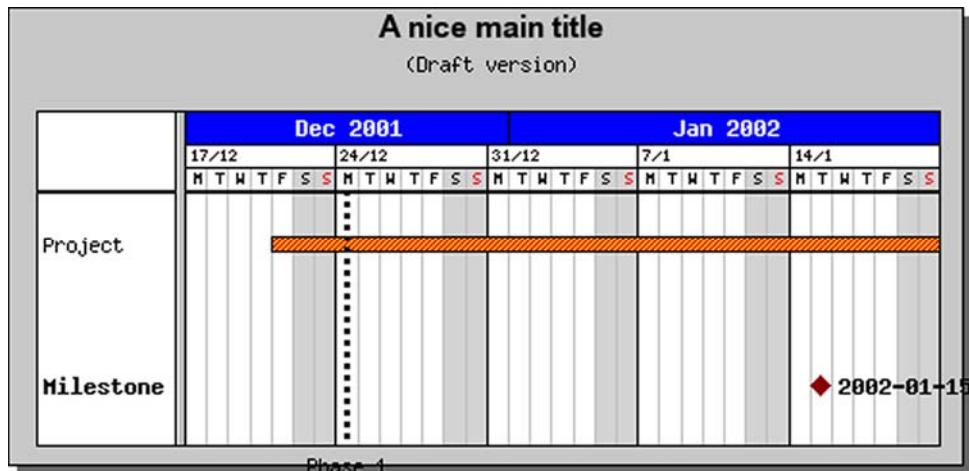


図 156:バーの高さの絶対サイズを指定する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(Draft version)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// Instead of week number show the date for the first
// day in the week
// on the week scale
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
$graph->scale->week->SetFont(FF_FONTO);

// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-
21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Add a right marker
$activity->rightMark->Show();
$activity->rightMark->SetType(MARK_FILLEDCIRCLE);
$activity->rightMark->SetWidth(13);
$activity->rightMark->SetColor("red");
$activity->rightMark->SetFillColor("red");
$activity->rightMark->title->Set("M5");
$activity->rightMark->title-
>SetFont(FF_ARIAL,FS_BOLD,12);
$activity->rightMark->title->SetColor("white");

// Set absolute height
$activity->SetHeight(8);

// Finally add the bar to the graph
$graph->Add($activity);

// Create a milestone
$milestone = new MileStone(2,"Milestone","2002-01-
15","2002-01-15");
$milestone->title->SetColor("black");
$milestone->title->SetFont(FF_FONT1,FS_BOLD);
$graph->Add($milestone);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
$graph->Add($vline);

// ... and display it
$graph->Stroke();
?>
```

9.7.6 バーの間の最小距離を調整する

各バーの高さを確保された高さの 100%として設定すると、一つ目のバーの下にもうひとつのアクティビティバーを加えることによってどんなことが起こるのか見てみましょう。そして、次の2行を前の例に付け加えることによって、各バーの高さを設定された高さの 100%として設定するとどうなるのか見てみましょう(ここでは追加するもうひとつのバーについての追加行は省略してあります)。

```
$activity->SetHeight(1.0);
$activity2->SetHeight(1.0);
```

([0..1] の範囲の値は設定された高さの割合として解釈され、一方、1 より大きな値はピクセルの絶対値として解釈されることに注意してください。)

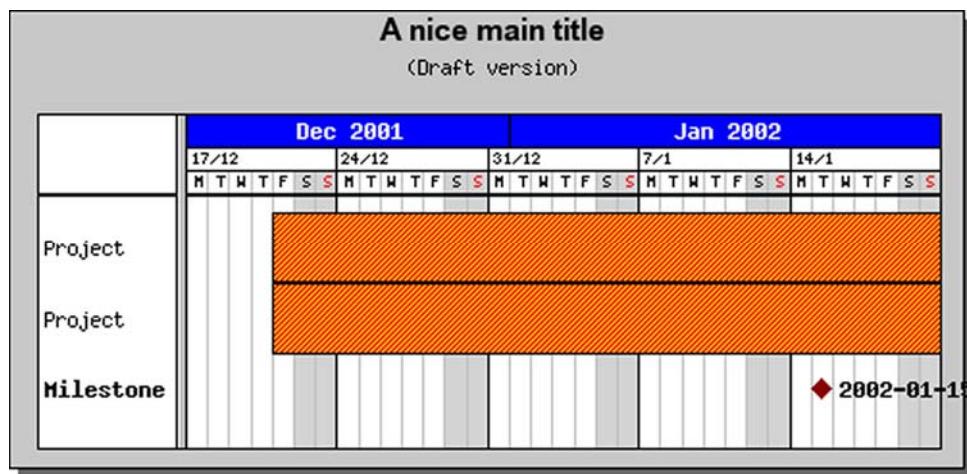


図 157:それぞれのバーの高さを 100%に設定する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0.0,"auto");
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(Draft version)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// Instead of week number show the date for the first
// day in the week
// on the week scale
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
$graph->scale->week->SetFont(FF_FONTO);

// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
```

```
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-
21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Add a right marker
$activity->rightMark->Show();
$activity->rightMark->SetType(MARK_FILLEDCIRCLE);
$activity->rightMark->SetWidth(13);
$activity->rightMark->SetColor("red");
$activity->rightMark->SetFillColor("red");
$activity->rightMark->title->Set("M5");
$activity->rightMark->title-
>SetFont(FF_ARIAL,FS_BOLD,12);
$activity->rightMark->title->SetColor("white");

// Set absolute height
$activity->SetHeight(1);

// Format the bar for the second activity
// ($row,$title,$startdate,$enddate)
$activity2 = new GanttBar(1,"Project","2001-12-
21","2002-01-20");
```

```
// Yellow diagonal line pattern on a red background
$activity2->SetPattern(BAND_RDIAG,"yellow");
$activity2->SetFillColor("red");

// Add a right marker
$activity2->rightMark->Show();
$activity2->rightMark->SetType(MARK_FILLEDIRCLE);
$activity2->rightMark->SetWidth(13);
$activity2->rightMark->SetColor("red");
$activity2->rightMark->SetFillColor("red");
$activity2->rightMark->title->Set("M5");
$activity2->rightMark->title-
>SetFont(FF_ARIAL,FS_BOLD,12);
$activity2->rightMark->title->SetColor("white");

// Set absolute height
$activity2->SetHeight(1);

// Finally add the bar to the graph
$graph->Add($activity);
$graph->Add($activity2);

// Create a milestone
$milestone = new MileStone(2,"Milestone","2002-01-
15","2002-01-15");
$milestone->title->SetColor("black");
$milestone->title->SetFont(FF_FONT1,FS_BOLD);
$graph->Add($milestone);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
// $graph->Add($vline);

// ... and display it
$graph->Stroke();
?>
```

バーが確保された利用可能な高さの 100%を占めるように指定したので、それぞれのバーの間には余白がありません。では、バーを絶対値で 10 ピクセルと指定したらどうなるのでしょうか。

```
$activity->SetHeight(10);
$activity2->SetHeight(10);
```

代わりに、次のような結果を得るでしょう。

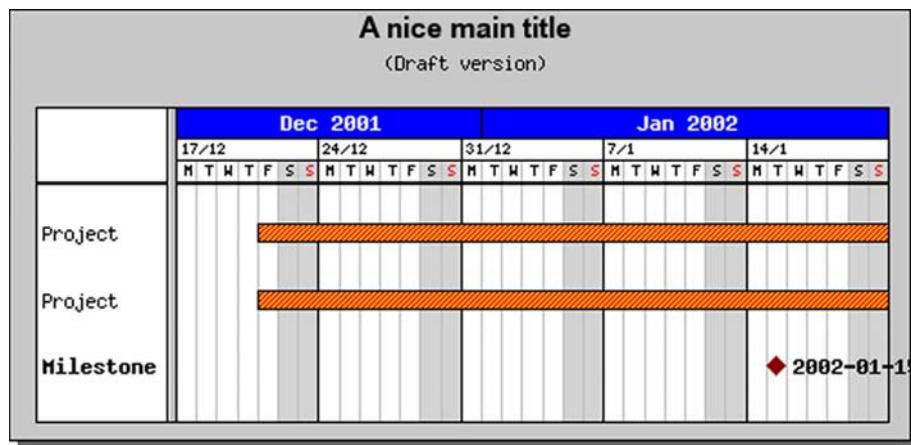


図 158: 両方のバーの高さを、10 ピクセルに設定する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(Draft version)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_
HMONTH);

// Instead of week number show the date for the first
// day in the week
// on the week scale
$graph->scale->week->SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
$graph->scale->week->SetFont(FF_FONTO);

// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month->SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-
21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Add a right marker
$activity->rightMark->Show();
$activity->rightMark->SetType(MARK_FILLEDIRCLE);
$activity->rightMark->SetWidth(13);
$activity->rightMark->SetColor("red");
$activity->rightMark->SetFillColor("red");
$activity->rightMark->title->Set("M5");
$activity->rightMark->title-
>SetFont(FF_ARIAL,FS_BOLD,12);
$activity->rightMark->title->SetColor("white");

// Set absolute height
$activity->SetHeight(10);

// Finally add the bar to the graph
$graph->Add($activity);
$graph->Add($activity2);

// Create a milestone
$milestone = new MileStone(2,"Milestone","2002-01-
15","2002-01-15");
$milestone->title->SetColor("black");
$milestone->title->SetFont(FF_FONT1,FS_BOLD);
$graph->Add($milestone);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
// $graph->Add($vline);

// ... and display it
$graph->Stroke();
?>
```

```

$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Add a right marker
$activity->rightMark->Show();
$activity->rightMark->SetType(MARK_FILLED CIRCLE);
$activity->rightMark->SetWidth(13);
$activity->rightMark->SetColor("red");
$activity->rightMark->SetFillColor("red");
$activity->rightMark->title->Set("M5");
$activity->rightMark->title-
>SetFont(FF_ARIAL,FS_BOLD,12);
$activity->rightMark->title->SetColor("white");

// Set absolute height
$activity->SetHeight(10);

// Format the bar for the second activity
// ($row,$title,$startdate,$enddate)
$activity2 = new GanttBar(1,"Project","2001-12-
21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity2->SetPattern(BAND_RDIAG,"yellow");
$activity2->SetFillColor("red");

// Add a right marker
$activity2->rightMark->Show();
$activity2->rightMark->SetType(MARK_FILLED CIRCLE);

$activity2->rightMark->SetWidth(13);
$activity2->rightMark->SetColor("red");
$activity2->rightMark->SetFillColor("red");
$activity2->rightMark->title->Set("M5");
$activity2->rightMark->title-
>SetFont(FF_ARIAL,FS_BOLD,12);
$activity2->rightMark->title->SetColor("white");

// Set absolute height
$activity2->SetHeight(10);

// Finally add the bar to the graph
$graph->Add($activity);
$graph->Add($activity2);

// Create a milestone
$milestone = new MileStone(2,"Milestone","2002-01-
15","2002-01-15");
$milestone->title->SetColor("black");
$milestone->title->SetFont(FF_FONT1,FS_BOLD);
$graph->Add($milestone);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
// $graph->Add($vline);

// ... and display it
$graph->Stroke();
?>

```

それでは、私たちは実際にどのようなことが出来るのでしょうか。各バーに対して確保された高さが、タイトルを含むバーの高さの最大値であることを覚えているでしょうか。これは、2つのバーが決して重ならないということを保証します。タイトルがお互いあまり近くにならないようにするために、タイトル間に余白を指定する **Vertical Label Margin** があります。余白の量は、タイトルの高さのパーセントで指定します。これを設定するためには、以下の文を使用します。

```
GanttGraph::SetLabelVMarginFactor($aMargin)
```

前の例のマージンを 0 に設定すると、どのように変化するのか見てみましょう。

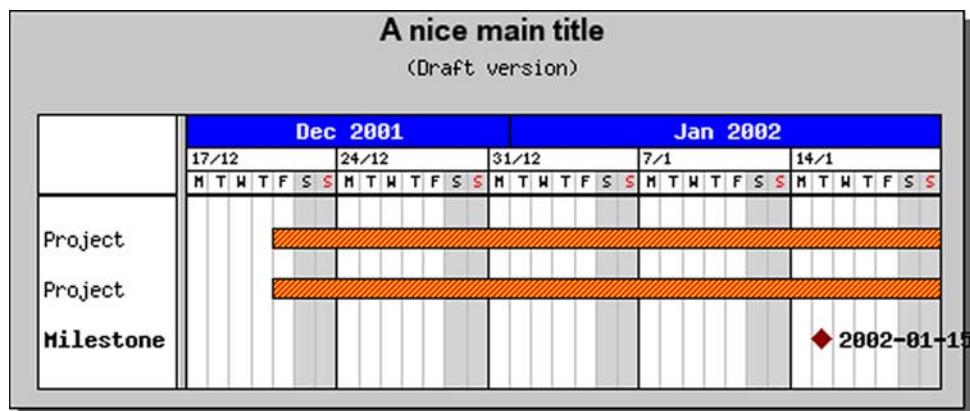


図 159: ラベルの縦方向のマージンを 0%に設定する

```

<?php
include ("./jpgraph.php");
include ("./jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetBox();
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("A nice main title");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(Draft version)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_

```

```

HMMONTH);

// Instead of week number show the date for the first
// day in the week
// on the week scale
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
$graph->scale->week->SetFont(FF_FONTO);

// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// 0 % vertical label margin
$graph->SetLabelVMarginFactor(0);

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-
21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Add a right marker
$activity->rightMark->Show();
$activity->rightMark->SetType(MARK_FILLEDCIRCLE);
$activity->rightMark->SetWidth(13);
$activity->rightMark->SetColor("red");
$activity->rightMark->SetFillColor("red");
$activity->rightMark->title->Set("M5");
$activity->rightMark->title-
>SetFont(FF_ARIAL,FS_BOLD,12);
$activity->rightMark->title->SetColor("white");

// Set absolute height
$activity->SetHeight(10);

```

```

// Format the bar for the second activity
// ($row,$title,$startdate,$enddate)
$activity2 = new GanttBar(1,"Project","2001-12-
21","2002-01-20");

// Yellow diagonal line pattern on a red background
$activity2->SetPattern(BAND_RDIAG,"yellow");
$activity2->SetFillColor("red");

// Add a right marker
$activity2->rightMark->Show();
$activity2->rightMark->SetType(MARK_FILLEDCIRCLE);
$activity2->rightMark->SetWidth(13);
$activity2->rightMark->SetColor("red");
$activity2->rightMark->SetFillColor("red");
$activity2->rightMark->title->Set("M5");
$activity2->rightMark->title-
>SetFont(FF_ARIAL,FS_BOLD,12);
$activity2->rightMark->title->SetColor("white");

// Set absolute height
$activity2->SetHeight(10);

// Finally add the bar to the graph
$graph->Add($activity);
$graph->Add($activity2);

// Create a milestone
$milestone = new MileStone(2,"Milestone","2002-01-
15","2002-01-15");
$milestone->title->SetColor("black");
$milestone->title->SetFont(FF_FONT1,FS_BOLD);
$graph->Add($milestone);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
// $graph->Add($vline);

// ... and display it
$graph->Stroke();
?>

```

予想されたように、余白が全く加えられていないので、二つのバーはほとんど接触している状態です。2つのバーに余分な右マーカーがない場合、とても圧縮されたように見えます。

デフォルトでは、縦方向の余白は 40% に設定されています。

9.8 スケール ヘッダーをフォーマットする

スケールヘッダーは、同時に4つまでの異なるスケールを表示させることができます。4つの基本的なスケールは、以下のとおりです。

- 日スケール
- 週スケール
- 月スケール
- 年スケール

SetScale() メソッドを使用することで、どのスケールを使うかを選択することができます。例えば、詳細なガント チャートに対しては、次のように指定して日や週の表示を選択できます。

```
$graph->ShowHeaders( GANTT_HWEEK | GANTT_DAY );
```

大きな画像を生成したい場合、次のように指定すれば年や月を表示させるのに十分なスペースを確保することができます。

```
$graph->ShowHeaders( GANTT_HWEEK | GANTT_DAY );
```

好みのスケールの組み合わせを自由に選択できますが、当然ながら1つのチャートは少なくとも1つのスケールを持っている必要があります。

どの程度詳細なスケールを必要としているかを一旦決めたならば、次に、使用できる各スケールの詳細なレイアウト/フォーマットを以下のようにして調整します。

これらのスケールヘッダーは、すべて次のように graph インスタンスの 'scale' を使って指定します。

```
$graph->scale->week
```

あるいは

```
$graph->scale->day
```

これらすべてのヘッダーは、次のプロパティを共有しています。

```
Show()
```

スケールを表示するかどうかを設定する

```
SetFont()
```

ヘッダー内のテキストのフォントを設定する

```
SetFontColor()
```

ヘッダーのテキストの色を指定する

```
SetStyle()
```

使用する日付の形式を指定する。例えば、週スケールにおいては、週の数、週の開始の日付などを指定することができます。

```
SetBackgroundColor()
```

ヘッダーの背景色を指定する

```
SetFrameWeight()
```

スケール周りのボックスの線の太さを指定する

```
SetFrameColor()
```

フレームの線の色を指定する

```
SetTitleVertMargin()
```

タイトル文の上下の余白をパーセントで設定する

これらの関数に加えて、どのスケールもグリッドラインの表示方法を決定する「grid」プロパティを備えています。すなわち SetColor()、SetWeight()、SetStyle()、Show()といった通常のラインメソッドを用いてグリッドラインの表示を変更することができます。例えば、週のグリッドラインを赤色に設定するためには、次のように設定します。

```
$graph->scale->week->grid->SetColor("red");
```

これから記述するように、どのスケールにも特別なフォーマットが備えられています。

9.8.1 分スケール

分スケールは、使用することのできるスケールの中で最も小さなスケールです。デフォルトでは、分スケールは 1 分きぎみなので、分スケールを GanttScale::SetInterval() と一緒に使用すると便利なことがしばしばあります。分スケールのスタイルは、次のうちの 1 つの style パラメータを使用して、さらに調節することができます。

1. "MINUTESTYLE_MM" これは 10 の位にもゼロを使用し、必ず 2 衔の数字で分を表示します。
2. "MINUTESTYLE_CUSTOM"。これを使用すると、HeaderProperty::SetFormatString() を呼び出すことで、独自の分スタイルを指定できます。

分スケールは、GanttGraph::ShowHeaders() を呼び出す際に、GANTT_HMIN を与えることで使用できます。例えば次のようになります。

```
$graph->ShowHeaders(GANTT_HDAY | GANTT_HHOUR | GANTT_HMIN);
```

以下のコードは、分スケールを 30 分間隔とし、いくつかのカスタムカラーを設定する方法を示したものです。

```
$graph->scale->minute->SetInterval(30);
$graph->scale->minute-
>SetBackgroundColor('lightyellow:1.5');
```

```
$graph->scale->minute->SetFont(FF_FONT0);
$graph->scale->minute->SetStyle(MINUTESTYLE_MM);
$graph->scale->minute->grid->SetColor('lightgray');
```

9.8.2 時スケール

時スケールは、より多くの組み込みフォーマットを備えています。以下のフォーマットオプションを使用することが可能です。

1. "HOURSTYLE_HM24" 13:00 のように、24 時間制の時刻表示で、時と分の両方を表示します。
2. "HOURSTYLE_H24" 13 のように、24 時間制の時刻表示で、時のみを表示します。
3. "HOURSTYLE_HMAMPM" 1:30pm のように、適切な am/pm とともに時と分を表示します。
4. "HOURSTYLE_HAMPM" 1pm のように、適切な am/pm とともに時のみを表示します。
5. "HOURSTYLE_CUSTOM" ヘッダープロパティの SetFormatString()を呼び出して指定されるカスタム定義のフォーマットで表示します。

時間の間隔は 2 つの方法のどちらかで指定することができます。6 のように整数を使用する方法、もしくは "1:30" のように時間を指定することができます。これは時間間隔を 1 時間 30 分にする例です。唯一の制限は、その間隔が 24 時間にに対して割り切れる数値でなければならないということです。というのは、一日は、表示可能な最も小さな間隔であるためです。つまり、例えば、2、4、6、"1:30"、"0:45" は使うことができますが、7 や "2:45" は使えないということです。

以下のコードは、分スケールを 45 分間隔とし、いくつかのカスタムカラーを設定する方法を示したものです。

```
$graph->scale->hour->SetBackgroundColor('lightyellow:1.5');
$graph->scale->hour->SetFont(FF_FONT1);
```

```
$graph->scale->hour->SetStyle(HOURSTYLE_HMAMPM);
$graph->scale->hour->SetInterval("0:45");
```

以下は、日スケールと時スケールを使えるようにした例です。

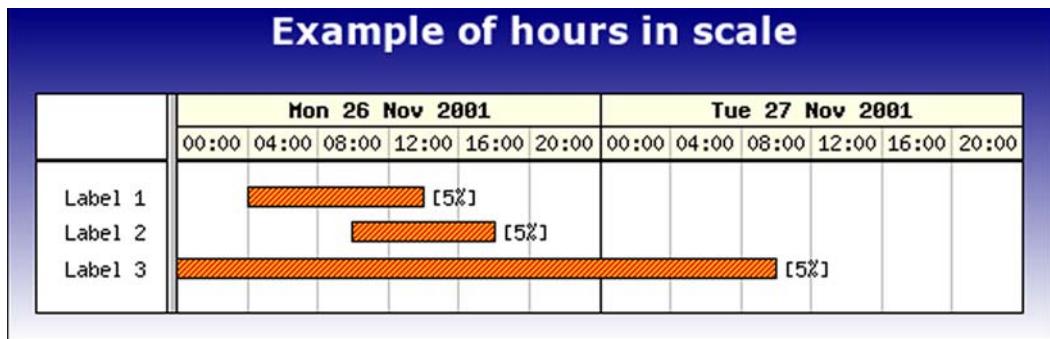


図 160: ガントチャート内で時スケールを使用する

```
<?php
// Gantt hour example
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph();
$graph->SetMarginColor('blue:1.7');
$graph->SetColor('white');

$graph-
>SetBackgroundGradient('navy','white',GRAD_HOR,BGRAD_MARGIN);
$graph->scale->hour-
>SetBackgroundColor('lightyellow:1.5');
$graph->scale->hour->SetFont(FF_FONT1);
$graph->scale->day-
>SetBackgroundColor('lightyellow:1.5');
$graph->scale->day->SetFont(FF_FONT1,FS_BOLD);

$graph->title->Set("Example of hours in scale");
$graph->title->SetColor('white');
$graph->title->SetFont(FF_VERDANA,FS_BOLD,14);

$graph->ShowHeaders(GANTT_HDAY | GANTT_HHOUR);

$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);
$graph->scale->week->SetFont(FF_FONT1);
$graph->scale->hour->SetInterval(4);
```

```
$graph->scale->hour->SetStyle(HOURSTYLE_HM24);
$graph->scale->day-
>SetStyle(DAYSTYLE_SHORTDAYDATE3);

$data = array(
    array(0,"Label 1", "011126 04:00","011126 14:00"),
    array(1,"Label 2", "011126 10:00","011126 18:00"),
    array(2,"Label 3", "011126","011127 10:00")
);

for($i=0; $i<count($data); ++$i) {
    $bar = new GanttBar($data[$i][0],$data[$i][1],$data[$i][2],$data[$i][3],"[5%]",10);
    if( count($data[$i])>4 )
        $bar->title-
    >SetFont($data[$i][4],$data[$i][5],$data[$i][6]);
    $bar->SetPattern(BAND_RDIAG,"yellow");
    $bar->SetFillColor("red");
    $graph->Add($bar);
}

$graph->Stroke();

?>
```

9.8.3 日スケール

デフォルトでは、日スケールは曜日のアルファベット表記の最初の1文字を表示します。しかし、日スケールを違う方法でフォーマットすることも可能です。 日にちは以下のスケール フォーマットのうちの 1 つを使用することができます。

- DAYSTYLE_ONELETTER 曜日のアルファベット表記の最初の 1 文字 例 ,“M”
- DAYSTYLE_LONG 曜日のアルファベット表記 例,“Monday”
- DAYSTYLE_LONGDAYDATE1 日付と曜日の同時表記 例,“Monday 23 Jun”

- DAYSTYLE_LONGDAYDATE2 西暦と日付と曜日を同時表記 例,“Monday 23 Jun 2003”
- DAYSTYLE_SHORT 曜日を短く表記 例,“Mon”
- DAYSTYLE_SHORTDAYDATE1 日付と曜日を短く表記 例,“Mon 23/6”
- DAYSTYLE_SHORTDAYDATE2 月は英語で表記し、日付と曜日を短く表示 例,“Mon 23 Jun”
- DAYSTYLE_SHORTDAYDATE3 曜日を短く表記し、日にちのみを加えて表示 例,“Mon 23”
- DAYSTYLE_SHORTDATE1 日付を短く表示 例,“23/6”
- DAYSTYLE_SHORTDATE2 月を英語で表記し、日付を短く表示 例,“23 Jun”
- DAYSTYLE_SHORTDATE3 日付を短く表示 例,“Mon 23”
- DAYSTYLE_SHORTDATE4 日にちのみを表示 例,“23”
- DAYSTYLE_CUSTOM 指定されたフォーマットで表示 例,“M”

SetStyle()関数を使用することで以下のようにフォーマットを指定できます。

```
$graph->scale->day->SetStyle(DAYSTYLE_LONG);
```

日スケールに対して用意されている特別なフォーマット方法を使用すると、週末や日曜日の背景色を異なった色に指定することもできます。

```
SetWeekendColor()
```

週末の背景色を設定します。(デフォルトでは薄いグレーに設定)

```
SetSundayFontColor()
```

日曜のフォントの色を設定します。(デフォルトでは赤色に設定)

これに加え、週末の背景をプロットエリアの下方向に延ばすか延ばさないかを選択することも可能です。(デフォルト) これはプロット全体の大部分に対するプロパティでなので、この動作を変更するには、スケールの UseWeekendBackground()を呼び出します。

例)

```
$graph->scale->UseWeekendBackground(false);
```

9.8.4 週スケール

週スケールは、デフォルトでは 1~53 の範囲で週数を表示します。(ISO8601 によって定義されています。詳細はリファレンスセクションを参照ください)

ここで、週数の計算は JpGraph 内で実行され、OS の日付ライブラリによらないということを記述しておく必要があるかもしれません。このことは、多くの OS 上で一貫した挙動を示すことを意味します(少なくとも、MS Windows は ISO-8601 に従わないか、あるいは、strftime()などの通常のライブラリを用いて週数を計算する方法を提供しています。)

3つの方法で、週の動作を変更することができます。以下の定数を `SetStyle()` 関数で使用することで異なる日付フォーマットを指定することができます。

`WEEKSTYLE_WNBR`

週数を表示します。週数の表示フォーマットをさらに変更するために、次の関数を呼び出してオプションでフォーマットの文字列を与えます。

`SetLabelFormatString()`

ここで使用する文字列のフォーマットは、整数(週数)を使う `sprintf()` 関数の書式でなくてはなりません。デフォルトでは、'W'が数字の前に付け加えられます。

`WEEKSTYLE_FIRSTDAY`

週の最初の日付を表示します。

`WEEKSTYLE_FIRSTDAY2`

週の最初の日付と月の略称を表示します。

`WEEKSTYLE_FIRSTDAYWNBR`

週の最初の日にちの週数を表示します。

`WEEKSTYLE_FIRSTDAY2WNBR`

週と月の最初の日の週数を表示します。

9.8.5 月スケール

月スケールには、様々なフォーマットの選択に、`SetStyle()`メソッドを使用することができます。

`MONTHSTYLE_SHORTNAME`

Jan, Feb などのように、ロケール特有の省略形で月の名称を表示します。

`MONTHSTYLE_SHORTNAMEYEAR2`

Jan '01, Feb '01 などのように、2桁表示の年と共に、ロケール特有の省略形で月の名前を表示します。

`MONTHSTYLE_SHORTNAMEYEAR4`

Jan 2001, Feb 2001 などのように、4桁表示の年と共に、ロケール特有の省略形を用いて月の名前を表示します。

`MONTHSTYLE_LONGNAME`

January, February というように、ロケール特有の正式名称で月の名称を表示します。

`MONTHSTYLE_LONGNAMEYEAR2`

January '01, February '01 などのように、2桁表示の年と共に、ロケール特有の正式名称を用いて月の名称を表示します。

`MONTHSTYLE_LONGNAMEYEAR4`

January 2001, February 2001 などのように、4桁表示の年と共に、ロケール特有の正式名前で月の名称を表示します。

`MONTHSTYLE_FIRSTLETTER`

月名の最初の文字を表示します。

9.8.6 年スケール

年スケールには、特殊なフォーマットはありません。

9.9 バーをより詳細にフォーマットする

この章では、アクティビティバーに対してさらに細かい変更を加える方法を説明します。

9.9.6 バーに見出しを追加する

バーの見出しが、バーの一番右側につけられます。それらは、例えば、タスクそのもやタスクの継続時間、アクティビティの経過に対して割り当てられるリソースを表示するために使用されます。

バーの見出し文は、バーの作成時に指定するか、後でバーのキャプションプロパティから指定することができます。

```
$activity = new GanttBar(0,"Activity 1","2001-11-21","2001-12-20","[BS,ER]")
$activity->caption->Set("[BS,ER]");
```

上記の 2 行はともにアクティビティの見出し “[BS,ER]” を指定する方法です。アクティビティは標準の JpGraph テキストオブジェクトなので、SetFont() や SetColor() を用いて、簡単にフォントや色、サイズを修正できます。

例)

```
$activity->caption->SetFont(FF_ARIAL,FF_BOLD,9);
```

以下の図は、見出しの使用例を示したものです。

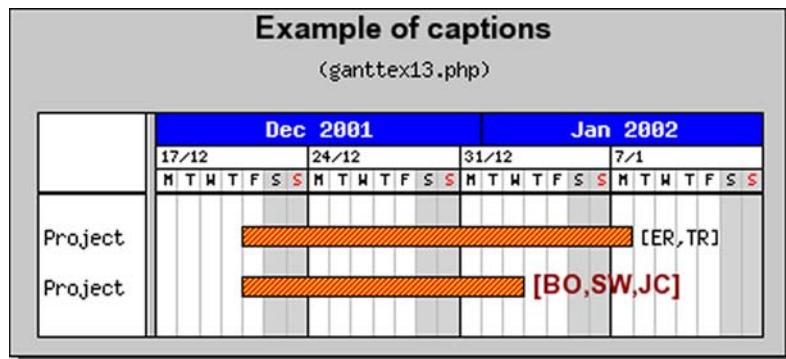


図 161: 見出しを使用した場合の実例

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetBox();
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("Example of captions");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(ganttex13.php)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_
HMONTH);

// Instead of week number show the date for the first
day in the week
// on the week scale
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);

// Make the week scale font smaller than the default
$graph->scale->week->SetFont(FF_FONTO);

// Use the short name of the month together with a
2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR4);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// 0 % vertical label margin
$graph->SetLabelVMarginFactor(1);
```

```
// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-21","2002-01-07","[ER,TR]");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Set absolute height
$activity->SetHeight(10);

// Format the bar for the second activity
// ($row,$title,$startdate,$enddate)
$activity2 = new GanttBar(1,"Project","2001-12-21","2002-01-02","[BO,SW,JC]");

// ADjust font for caption
$activity2->caption->SetFont(FF_ARIAL,FS_BOLD);
$activity2->caption->SetColor("darkred");
```

```
// Yellow diagonal line pattern on a red background
$activity2->SetPattern(BAND_RDIAG,"yellow");
$activity2->SetFillColor("red");

// Set absolute height
$activity2->SetHeight(10);

// Finally add the bar to the graph
$graph->Add($activity);
$graph->Add($activity2);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
// $graph->Add($vline);

// ... and display it
$graph->Stroke();
?>
```

9.9.7 バーに進行指標(progress indicators)を追加する

特定なアクティビティの進捗状況を示すために、進行指標をそれぞれのバーに加えることも可能で、この経過指標は、バーの中にある、より小さいバーから構成されます。デフォルトでは、この経過バーは黒色で、その高さはバーの 70%の高さをもっています。これらのパラメータはすべて変更することができます。

進行指標のプロパティは 'progress' プロパティとその関数を通じてアクセスすることができます。

特定のアクティビティに対してその進捗状況を設定するためには、以下のように割合を指定するだけです。

```
$activity->progress->Set(0.4)
```

図 162 では、デフォルトの進行指標を用いてそれぞれのアクティビティの進捗状況を示すように先ほどの例を修正しています。ソリッドバーを分かりやすくするために、表示された進捗状況を反映させるキャプションの修正も行いました。（同時に、フォーマットのオプションの例をあげるためだけにスケール ヘッダーを変更しました）。

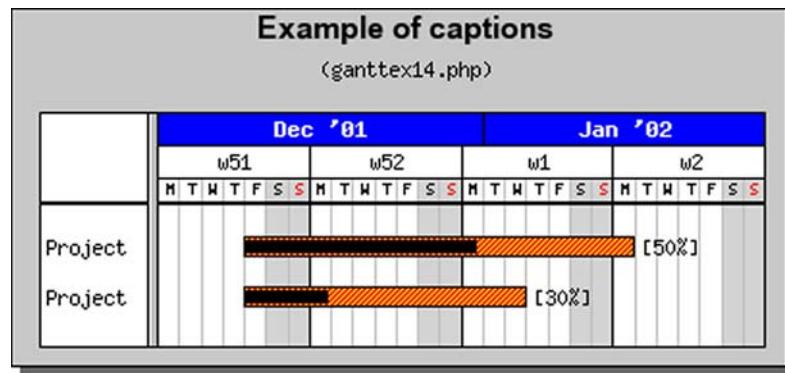


図 162:進行指標を加える

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");
```

```
$graph = new GanttGraph(0,0,"auto");
$graph->SetBox();
```

```

$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("Example of captions");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(ganttex14.php)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_
HMONT);

// Use the short name of the month together with a
2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR2);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// 0 % vertical label margin
$graph->SetLabelVMarginFactor(1);

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-
21","2002-01-07","[50%");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Set absolute height
$activity->SetHeight(10);

// Specify progress to 60%
$activity->progress->Set(0.6);

// Format the bar for the second activity
// ($row,$title,$startdate,$enddate)
$activity2 = new GanttBar(1,"Project","2001-12-
21","2002-01-02","[30%");

// Yellow diagonal line pattern on a red background
$activity2->SetPattern(BAND_RDIAG,"yellow");
$activity2->SetFillColor("red");

// Set absolute height
$activity2->SetHeight(10);

// Specify progress to 30%
$activity2->progress->Set(0.3);

// Finally add the bar to the graph
$graph->Add($activity);
$graph->Add($activity2);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
// $graph->Add($vline);

// ... and display it
$graph->Stroke();
?>

```

進行指標に違ったフォーマットを指定するには、次のように SetPattern() メソッドを使用します。

```
$activity->progress-> SetPattern(BAND_RDIAG,"blue");
```

適切なパラメータや使用できるすべてのメソッドについては、リファレンスの章を参照下さい。

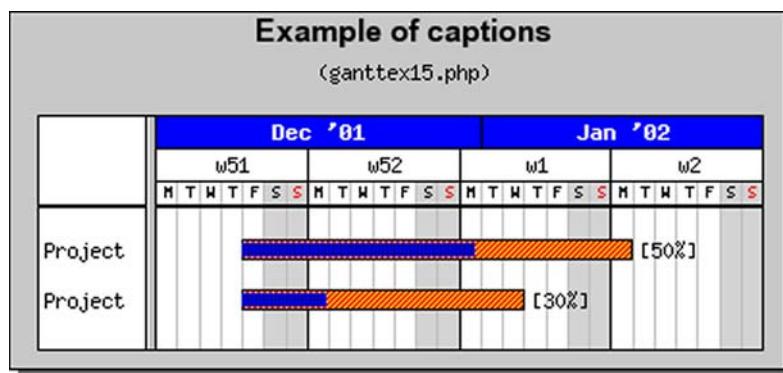


図 163: 進行指標のスタイルを変更する

```

<?php
include ("../jgraph.php");
include ("../jgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetBox();
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("Example of captions");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(ganttex15.php)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_
HMONT);

// Use the short name of the month together with a
2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR2);

```

```

$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// 0 % vertical label margin
$graph->SetLabelVMarginFactor(1);

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-21","2002-01-07","[50%]");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Set absolute height
$activity->SetHeight(10);

// Specify progress to 60%
$activity->progress->Set(0.6);
$activity->progress->SetPattern(BAND_HVCROSS,"blue");

// Format the bar for the second activity
// ($row,$title,$startdate,$enddate)
$activity2 = new GanttBar(1,"Project","2001-12-21","2002-01-02","[30%]");

```



```

// Yellow diagonal line pattern on a red background
$activity2->SetPattern(BAND_RDIAG,"yellow");
$activity2->SetFillColor("red");

// Set absolute height
$activity2->SetHeight(10);

// Specify progress to 30%
$activity2->progress->Set(0.3);
$activity2->progress-
>SetPattern(BAND_HVCROSS,"blue");

// Finally add the bar to the graph
$graph->Add($activity);
$graph->Add($activity2);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
// $graph->Add($vline);

// ... and display it
$graph->Stroke();
?>

```

9.10 アクティビティのグループ化

アクティビティをグループ化することは一般的なことです。私たちは、制限の例のところでこの機能を使用しました。グループ化の例をあげるために表示するアクティビティバーには、特別なタイプはありません。(先に使用したように)グループ化していることを示す一般的な方法は、バーの両端に“half”の三角マークを加えるというものです。JpGraph がグループ化の表示に関して行う特別な処理は、あなたが MARK_LEFTTRIANGLE や MARK_RIGHTTRIANGLE といったタイプのマークを加える場合、上の例で示されたような効果をもたらすために、バーの下にそれらの三角形が描かれるということです。また、上の例の中で、端の三角形が視覚的にバーを長くしてしまうので、私たちはグループ化したバーをやや低くしました。

そのため、グループバーの効果を得るためにには、以下の 2 行を使用する必要があります。

```

$groupbar->leftMark->SetType("MARK_LEFTTRIANGLE");
$groupbar->rightMark->SetType("MARK_RIGHTTRIANGLE");

```

タイトルに字下げを施す機能はついていないため、これを行うためにはタイトルの文字列の先頭にスペースを加えてください。

9.11 タイトルに複数行を設定する

ガントバーに 1 つのタイトルを表示するだけでなく、タイトルやスタート日、終了日、継続時間といったものを表示したいということがしばしば起こります。これまで、作業を簡単化するために、それぞれのアクティビティに対して 1 つのタイトルのみを表示してきました。そこで、ここではグラフ列へアイコンをえたときと同様に、ガントチャートに任意の数の文字列をタイトルとして指定する方法を説明しましょう。

まず最初に、私たちがこれから行うことをイメージしやすくするため、1つの例を示します。

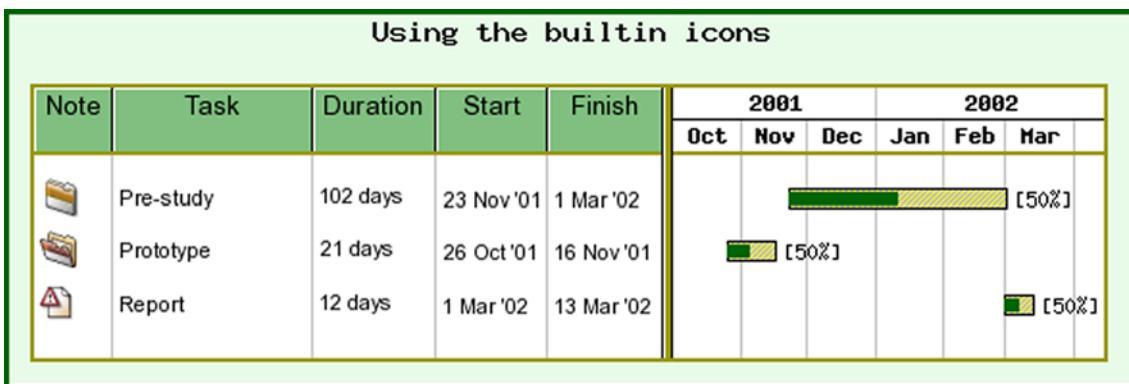


図 164: 複数の列を使用したガントチャート

```

<?php
// Gantt example
include ("./jpgraph.php");
include ("./jpgraph_gantt.php");

// Basic Gantt graph
$graph = new GanttGraph();
$graph->title->Set("Using the builtin icons");

// Explicitly set the date range
// (Autoscaling will of course also work)
$graph->SetDateRange('2001-10-06','2002-4-10');

// 1.5 line spacing to make more room
$graph->SetVMarginFactor(1.5);

// Setup some nonstandard colors
$graph->SetMarginColor('lightgreen@0.8');
$graph->SetBox(true,'yellow:0.6',2);
$graph->SetFrame(true,'darkgreen',4);
$graph->scale->divider->SetColor('yellow:0.6');
$graph->scale->dividerh->SetColor('yellow:0.6');

// Display month and year scale with the gridlines
$graph-
>ShowHeaders(GANTT_HMONTH | GANTT_HYEAR);
$graph->scale->month->grid->SetColor('gray');
$graph->scale->month->grid->Show(true);
$graph->scale->year->grid->SetColor('gray');
$graph->scale->year->grid->Show(true);

// For the titles we also add a minimum width of 100
// pixels for the Task name column
$graph->scale->actinfo->SetColTitles(
    array('Note','Task','Duration','Start','Finish'),array(30,1
    00));
$graph->scale->actinfo-
>SetBackgroundColor('green:0.5@0.5');
$graph->scale->actinfo-
>SetFont(FF_ARIAL,FS_NORMAL,10);
$graph->scale->actinfo->vgrid->SetStyle('solid');
$graph->scale->actinfo->vgrid->SetColor('gray');

// Uncomment this to keep the columns but show no h
eaders
// $graph->scale->actinfo->Show(false);

// Setup the icons we want to use
$erricon = new IconImage(GICON_FOLDER,0.6);
$startconicon = new IconImage(GICON_FOLDEROPEN,0.6);
$endconicon = new IconImage(GICON_TEXTIMPORTANT,0.
5);

// Store the icons in the first column and use plain te
xt in the others
$data = array(
    array(0,array($erricon,"Pre-
study","102 days","23 Nov '01","1 Mar '02"
, "2001-11-23","2002-03-
1",FF_ARIAL,FS_NORMAL,8),
        array(1,array($startconicon,"Prototype","21 days",
26 Oct '01,"16 Nov '01",
"2001-10-26","2001-11-
16",FF_ARIAL,FS_NORMAL,8),
            array(2,array($endconicon,"Report","12 days","1 Ma
r '02","13 Mar '02",
"2002-03-01","2002-03-
13",FF_ARIAL,FS_NORMAL,8)
        );
    );

// Create the bars and add them to the gantt chart
for($i=0; $i<count($data); ++$i) {
    $bar = new GanttBar($data[$i][0],$data[$i][1],$dat
a[$i][2],$data[$i][3],"[50%]",10);
    if( count($data[$i])>4 )
        $bar->title-
    >SetFont($data[$i][4],$data[$i][5],$data[$i][6]);
    $bar->SetPattern(BAND_RDIAG,"yellow");
    $bar->SetFillColor("gray");
    $bar->progress->Set(0.5);
    $bar->progress-
    >SetPattern(GANTT_SOLID,"darkgreen");
    $bar->title-
    >SetCSIMTarget(array('#1'.$i,'#2'.$i,'#3'.$i,'#4'.$i,'#5'.
$i),array('11'.$i,'22'.$i,'33'.$i));
    $graph->Add($bar);
}

// Output the chart
$graph->Stroke();

?>

```

複数行を使用するためには、2つのステップがあります。まず最初に、タイトル(名前、色、フォントなど)を設定し、その後、個々のバーに対してタイトル行を加える必要があります。

タイトル行を設定するためには、Activity information プロパティを使用して、作業する必要があります。以下のコードは、その設定方法を示したものです。

```
$graph->scale->actinfo->SetColTitles(  
array('Note','Task','Duration','Start','Finish'),array(30,100));
```

ここで2つのこと気にづくでしょう。はじめに、1つの配列を使用してタイトルを指定しました。また、30と100という数を用いて2番目の配列も指定しました。この配列は、この場合最初の2つのカラムの幅の最小値を指定するオプションの配列です。デフォルトでは、カラムは最大長のテキスト文字列を全て含むことができる長さに設定されます。しかし、見た目を整るために最小幅を大きくしたいと思うことがあるかもしれません。これが私たちがここで最初の2つのカラムに対して行ったことです。

さらに、背景の色やスタイル、縦の仕切りグリッドラインの色も調節することができます。これまでのイメージの中では、次の行を使用しました。

```
$graph->scale->actinfo->SetBackgroundColor('green:0.5@0.5');  
$graph->scale->actinfo->SetFont(FF_ARIAL_FS_NORMAL,10);  
$graph->scale->actinfo->vgrid->SetStyle('solid');  
$graph->scale->actinfo->vgrid->SetColor('gray');
```

また、ライブラリ内にある線に対する別のフォーマットの場合と同様に、グリッドラインのスタイルを”破線(dashed)”や”ドット(dotted)”、”長い破線(longdashed)”にすることも可能です。また、タイトル内に小さく3D効果をつけることも可能です。これはデフォルトで有効になっています。以下の文を使用すると、簡単に3D効果を付け加えることが出来ます。

```
$graph->scale->actinfo->SetStyle(ACTINFO_2D);
```

タイトル内の縦の仕切り線の色を調節するには、SetColor()メソッドを次のように使用します。
“\$graph->scale->actinfo->SetColor('navy');”.

次にすることは、実際にカラムを埋めることです。これはもちろん、グラフにアクティビティバーを加える際に実行されます。以前にカラムを使用していなかったとき、私たちはひとつの文字列をタイトルとして使用していました。この文字列を単に配列に取り替えることによって、カラムの内容を指定することができます。

例えば、上の例のようなガントバーを作成する2カラムのタイトルを指定する場合、数種類のバーを作成するので、よりよいコーディング習慣のためにこれらすべてを配列へ入れます。

通常のテキストに加え、画像や、先に定義した使用可能なアイコンを付け加えることもできます。カラムへそれらを加えるには、はじめにIconImage()のインスタンスを作成し、次にテキストのかわりにそれらのインスタンスを指定します。そこで、以前のコードの中で、最初のカラムに”オープンフォルダー”の画像を挿入したい場合、以下のように文を変更します。

```
$iconopen = new IconImage(GICON_FOLDEROPEN,0.6);  
$title2="";  
$bar = new GanttBar(0,array($iconopen,$title2),"2003-11-23","2003-12-05");
```

使用可能な内蔵アイコンは、以下の通りです。

- GICON_WARNINGRED
- GICON_TEXT
- GICON_ENDCONS
- GICON_MAIL
- GICON_STARTCONS
- GICON_CALC
- GICON_MAGNIFIER
- GICON_LOCK
- GICON_STOP
- GICON_WARNINGYELLOW
- GICON_FOLDEROPEN
- GICON_FOLDER
- GICON_TEXTIMPORTANT

さらに、引数を文字列として指定するならば、あなたが用意した画像を使用することもできます。

```
$myicon = IconImage('myicon.png');
```

奇妙に思えるかもしれません、IconImage() の呼び出しの際の 2 番目の引数は、画像の大きさを調節するために使用するオプションの値です。

9.11.6 列にタイトルを加える

これまでのよう正確に、すべての列にまたがるタイトルをつけることも可能です。このタイトルは、スケールの tableTitle プロパティを用いて指定します。テーブル タイトルを指定すると、テーブル タイトルに合うように、自動的にタイトル列の高さを調整します。以下の小さなコードは、タイトルを加える方法を示したものです。

```
$graph->scale->tableTitle->Set('Phase?1');
$graph->scale->tableTitle->SetFont(FF_ARIAL,FS_NORMAL,12);
$graph->scale->SetTableTitleBackground('darkgreen@0.6');
$graph->scale->tableTitle->Show(true);
```

タイトルを加える方法の詳細な例は、以下の通りです。

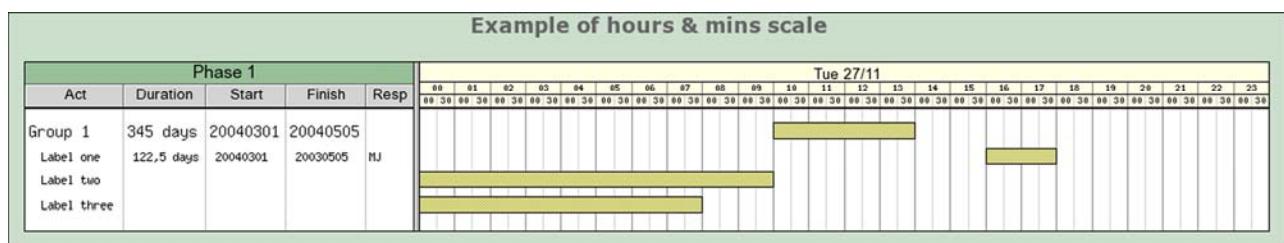


図 165:すべてのタイトル列を階層化する列タイトルを付け加える

```
<?php
// Gantt hour + minute example
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");
```

```
// Some sample Gantt data
$data = array(
    array(0,array("Group 1","345 days","20040301",
```

```

20040505"), "2001-11-27 10:00", "2001-11-
27 14:00", FF_FONT2, FS_NORMAL, 0),
    array(1, array(" Label one", '122,5 days', 200403
01), 20030505, 'MJ'), "2001-11-27 16:00", "2001-
11-27 18:00"),
    array(2, " Label two", "2001-11-27", "2001-11-
27 10:00"),
    array(3, " Label three", "2001-11-27", "2001-
11-27 08:00");
);

// Basic graph parameters
$graph = new GanttGraph();
$graph->SetMarginColor('darkgreen@0.8');
$graph->SetColor('white');

// We want to display day, hour and minute scales
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HHOUR | GANTT_
HMIN);

// We want to have the following titles in our columns
// describing each activity
$graph->scale->actinfo->SetColTitles(
    array('Act','Duration','Start','Finish','Resp'));//,array(10
0,70,70,70));

// Uncomment the following line if you don't want the
3D look
// in the columns headers
// $graph->scale->actinfo->SetStyle(ACTINFO_2D);

$graph->scale->actinfo-
>SetFont(FF_ARIAL,FS_NORMAL,10);

//These are the default values for use in the columns
// $graph->scale->actinfo->SetFontColor('black');
// $graph->scale->actinfo-
>SetBackgroundColor('lightgray');
// $graph->scale->actinfo->vgrid->SetStyle('solid');

$graph->scale->actinfo->vgrid->SetColor('gray');
$graph->scale->actinfo->SetColor('darkgray');

// Setup day format
$graph->scale->day-
>SetBackgroundColor('lightyellow:1.5');
$graph->scale->day->SetFont(FF_ARIAL);
$graph->scale->day-

```

```

>SetTitle(DAYSTYLE_SHORTDAYDATE1);

// Setup hour format
$graph->scale->hour->SetInterval(1);
$graph->scale->hour-
>SetBackgroundColor('lightyellow:1.5');
$graph->scale->hour->SetFont(FF_FONT0);
$graph->scale->hour->SetStyle(HOURSTYLE_H24);
$graph->scale->hour->grid->SetColor('gray:0.8');

// Setup minute format
$graph->scale->minute->SetInterval(30);
$graph->scale->minute-
>SetBackgroundColor('lightyellow:1.5');
$graph->scale->minute->SetFont(FF_FONT0);
$graph->scale->minute->SetStyle(MINUTESTYLE_MM);
$graph->scale->minute->grid->SetColor('lightgray');

$graph->scale->tableTitle->Set('Phase 1');
$graph->scale->tableTitle-
>SetFont(FF_ARIAL,FS_NORMAL,12);
$graph->scale-
>SetTableTitleBackground('darkgreen@0.6');
$graph->scale->tableTitle->Show(true);

$graph->title->Set("Example of hours & mins scale");
$graph->title->SetColor('darkgray');
$graph->title->SetFont(FF_VARDANA,FS_BOLD,14);

for($i=0; $i<count($data); ++$i) {
    $bar = new GanttBar($data[$i][0],$data[$i][1],$dat
a[$i][2],$data[$i][3]);
    if( count($data[$i])>4 )
        $bar->title-
>SetFont($data[$i][4],$data[$i][5],$data[$i][6]);
    $bar->SetPattern(BAND_RDIAG,"yellow");
    $bar->SetFillColor("gray");
    $graph->Add($bar);
}

$line = new GanttVLine("2001-11-27 13:00");
$line->title->Set("27/11 13:00");
$line->title->SetFont(FF_FONT1,FS_BOLD,10);
$graph->Add($line);

$graph->Stroke();

?>

```

9.11.7 列タイトルに対してCSIMエントリーを指定する

シングルタイトルの場合と全く同じ方法で、それぞれのタイトル列に対して個々のCSIMターゲットを指定することができます。これは、SetCSIMTarget()に対する引数として 1 つのストリングを指定するかわりに、ターゲットとオルトテキストの両方に対して1つの配列を指定することで実現されます。

```
$bar->title->SetCSIMTarget(array('sometarget1.html','sometarget1.html'),
array('alttext1','alttext2'));
```

9.12 より全体的なガント チャートのフォーマット

この章では、ガントチャートそのものをカスタマイズするための方法を、さらにいくつか紹介します。ここでは以下の内容を扱います。

- テーブルタイトルを加える(グラフのタイトルと混同しないこと)
- バーチャートの中の、様々なラインの外観を調節する

9.12.6 テーブル タイトルを追加する

ガントテーブルの左上の白い(デフォルトでは)部分には、タイトルをつけることができます。これは、ガントスケールの 'tableTitle' プロパティで設定します。以下に示すように、とても簡単です。

```
$graph->scale->tableTitle->Set("Rev: 1.22");
$graph->scale->tableTitle->SetFont(FF_FONT1,FS_BOLD);
$graph->scale->SetTableTitleBackground("silver");
$graph->scale->tableTitle->Show();
```

また、このサンプルコードは、デフォルトの白い背景を銀色に変更しています。先の例にこれらのコードを加えると、次のような結果が得られます。

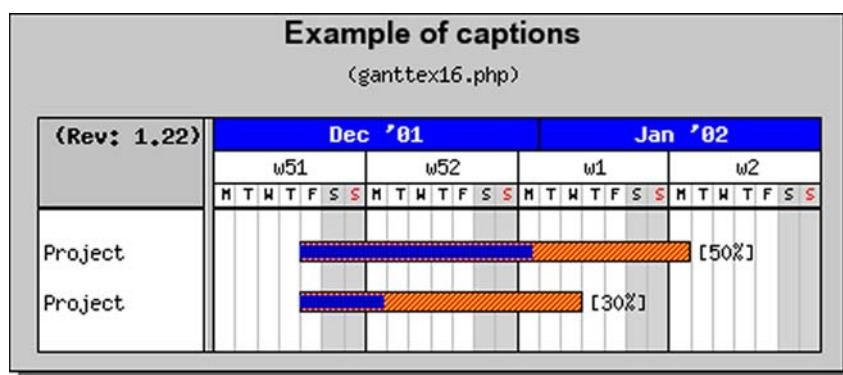


図 166:テーブル タイトルを追加する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetBox();
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("Example of captions");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(ganttex16.php)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// Set table title
$graph->scale->tableTitle->Set("Rev: 1.22");
$graph->scale->tableTitle->SetFont(FF_FONT1,FS_BOLD);
$graph->scale->SetTableTitleBackground("silver");
$graph->scale->tableTitle->Show();

// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR2);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// 0 % vertical label margin
$graph->SetLabelVMarginFactor(1);

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-
21","2002-01-07","[50%]");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Set absolute height
$activity->SetHeight(10);

// Specify progress to 60%
$activity->progress->Set(0.6);
$activity->progress->SetPattern(BAND_HVCROSS,"blue");

// Format the bar for the second activity
// ($row,$title,$startdate,$enddate)
$activity2 = new GanttBar(1,"Project","2001-12-
21","2002-01-02","[30%]");

// Yellow diagonal line pattern on a red background
$activity2->SetPattern(BAND_RDIAG,"yellow");
$activity2->SetFillColor("red");

// Set absolute height
$activity2->SetHeight(10);

// Specify progress to 30%
```

```
$activity2->progress->Set(0.3);
$activity2->progress-
>SetPattern(BAND_HVCROSS,"blue");

// Finally add the bar to the graph
$graph->Add($activity);
$graph->Add($activity2);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
// $graph->Add($vline);

// ... and display it
$graph->Stroke();
?>
```

以上の例から、左のコラムの幅(すべてのタイトルを含む)が、テーブルタイトルに合わせて自動的に調節されているということに気付かれるでしょう。

9.12.7 仕切り線を修正する

タイトルとバーの間の縦線や横線は、スケールの'divider' プロパティや'dividerh'プロパティで修正することができます。これもまた、次の例で示されるように簡単です。

```
$graph->scale->divider->SetWeight(3);
$graph->scale->divider->SetColor("navy");
$graph->scale->dividerh->SetWeight(3);
$graph->scale->dividerh->SetColor("navy");
```

この修正の効果は、下の図 167 に示されているとおりです。

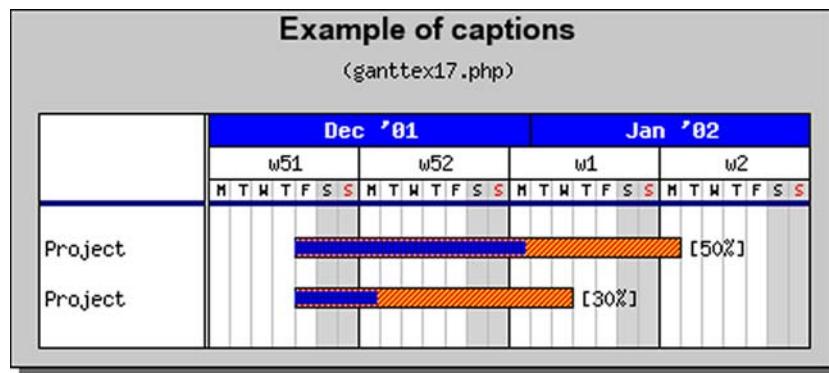


図 167:仕切り線を修正する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetBox();
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("Example of captions");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(gantttx17.php)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// Set table title
$graph->scale->tableTitle->Set("(Rev: 1.22)");
$graph->scale->tableTitle->SetFont(FF_FONT1,FS_BOLD);
$graph->scale->SetTableTitleBackground("silver");

// Modify the appearance of the dividing lines
$graph->scale->divider->SetWeight(3);
$graph->scale->divider->SetColor("navy");

$graph->scale->dividerh->SetWeight(3);
$graph->scale->dividerh->SetColor("navy");

// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR2);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// 0 % vertical label margin
$graph->SetLabelVMarginFactor(1);

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-
21","2002-01-07","[50%]");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Set absolute height
$activity->SetHeight(10);
```

```
// Specify progress to 60%
$activity->progress->Set(0.6);
$activity->progress->SetPattern(BAND_HVCROSS,"blue");

// Format the bar for the second activity
// ($row,$title,$startdate,$enddate)
$activity2 = new GanttBar(1,"Project","2001-12-21","2002-01-02","[30%");

// Yellow diagonal line pattern on a red background
$activity2->SetPattern(BAND_RDIAG,"yellow");
$activity2->SetFillColor("red");

// Set absolute height
$activity2->SetHeight(10);

// Specify progress to 30%
$activity2->progress->Set(0.3);
$activity2->progress-
>SetPattern(BAND_HVCROSS,"blue");

// Finally add the bar to the graph
$graph->Add($activity);
$graph->Add($activity2);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
// $graph->Add($vline);

// ... and display it
$graph->Stroke();
?>
```

9.12.8 プロット周辺のボックスを修正する

JpGraph の他のプロットと同様の方法で、次のように標準グラフメソッド 'SetBox()' を用いて、プロット周辺のボックスを修正することができます。

```
$graph->SetBox(true,"navy",3)
```

これは、以下に示されるように、エリア周辺により厚いボックスをプロットします。

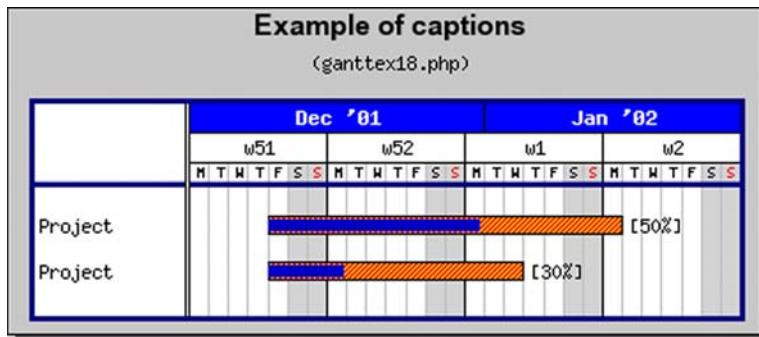


図 168: プロットエリア周辺のボックスを修正する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetBox();
$graph->SetShadow();

// Add title and subtitle
$graph->title->Set("Example of captions");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(ganttex18.php)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// Set table title
$graph->scale->tableTitle->Set("(Rev: 1.22)");
$graph->scale->tableTitle->SetFont(FF_FONT1,FS_BOLD);
$graph->scale->SetTableTitleBackground("silver");

// Modify the appearance of the dividing lines
$graph->scale->divider->SetWeight(3);
$graph->scale->divider->SetColor("navy");

$graph->scale->dividerh->SetWeight(3);
```

```
$graph->scale->dividerh->SetColor("navy");

$graph->SetBox(true,"navy",3);

// Use the short name of the month together with a
// 2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR2);
$graph->scale->month->SetFontColor("white");
$graph->scale->month->SetBackgroundColor("blue");

// 0 % vertical label margin
$graph->SetLabelVMarginFactor(1);

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-21","2002-01-07","[50%");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Set absolute height
$activity->SetHeight(10);
```

```
// Specify progress to 60%
$activity->progress->Set(0.6);
$activity->progress->SetPattern(BAND_HVCROSS,"blue");

// Format the bar for the second activity
// ($row,$title,$startdate,$enddate)
$activity2 = new GanttBar(1,"Project","2001-12-21","2002-01-02","[30%");

// Yellow diagonal line pattern on a red background
$activity2->SetPattern(BAND_RDIAG,"yellow");
$activity2->SetFillColor("red");

// Set absolute height
$activity2->SetHeight(10);

// Specify progress to 30%
$activity2->progress->Set(0.3);
$activity2->progress-
>SetPattern(BAND_HVCROSS,"blue");

// Finally add the bar to the graph
$graph->Add($activity);
$graph->Add($activity2);

// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
// $graph->Add($vline);

// ... and display it
$graph->Stroke();
?>
```

注意:ここでメソッドを使用する言語仕様と、修正したプロパティにアクセスした、以前の言語仕様との間に、ちょっとした矛盾があることに気付かれただかもしれません。これは、ライブラリ開発当初のオリジナルのデザインとそれ以降に拡張されたデザインの違いによる残念な結果です。

9.12.9 水平なグリッドとラインカラーの交互切り替え

水平なグリッドとガント チャートの背景のオプションのライン カラーを指定することで、より読みやすいチャートを作ることができます。水平なグリッドは Graph::hgrid プロパティを使ってアクセスし、ライン(グリッドで使用される)は Graph::hgrid::line のサブプロパティを使ってアクセスします。

ライン カラーを指定するには、SetRowFillColor() メソッドを使用します。例えば、青いグリッド ラインの青い背景が交互に表示されるようにするために、以下のコードをグラフ スクリプトに追加します。

```
// 水平グリッドの設定
$graph->hgrid->Show();
$graph->hgrid->line->SetColor('lightblue');
$graph->hgrid->SetRowFillColor('darkblue@0.9');
```

以下は、このフォーマットを設定したガントチャートの例です。

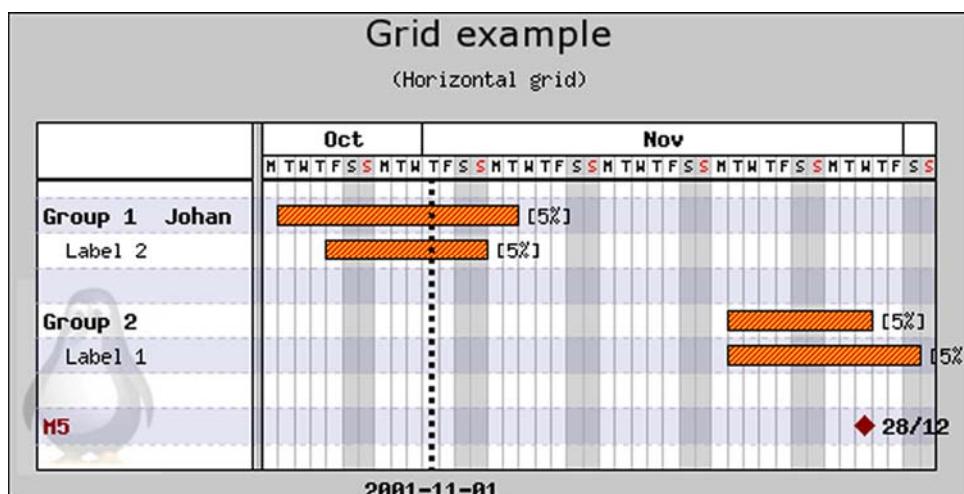


図 169:ガントチャートに水平グリッドを追加する

```

<?php
// Gantt horizontal grid example
include ("./jpgraph.php");
include ("./jpgraph_gantt.php");

// Some dummy data for some activities
$data = array(
    array(0,"Group 1 Johan", "2001-10-23","2001-11-06",FF_FONT1,FS_BOLD,8),
    array(1," Label 2", "2001-10-26","2001-11-04"),
    array(3,"Group 2", "2001-11-20","2001-11-28",FF_FONT1,FS_BOLD,8),
    array(4," Label 1", "2001-11-20","2001-12-1"));
}

// New Gantt Graph
$graph = new GanttGraph(500);

// Setup a title
$graph->title->Set("Grid example");
$graph->subtitle->Set("(Horizontal grid)");
$graph->title->SetFont(FF_VERDANA,FS_NORMAL,14);

// Specify what headers to show
$graph->ShowHeaders(GANTT_HMONTH|GANTT_HDAY );
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAY);
$graph->scale->week->SetFont(FF_FONTO);

// Setup a horizontal grid
$graph->hgrid->Show();
$graph->hgrid->SetRowFillColor('darkblue@0.9');

```

```

for($i=0; $i<count($data); ++$i) {
    $bar = new GanttBar($data[$i][0],$data[$i][1],$data[$i][2],$data[$i][3],"[5%]",10);
    if( count($data[$i]) > 4 )
        $bar->title-
>SetFont($data[$i][4],$data[$i][5],$data[$i][6]);
    $bar->SetPattern(BAND_RDIAG,"yellow");
    $bar->SetFillColor("red");
    $graph->Add($bar);
}

// Setup a vertical marker line
$vine = new GanttVLine("2001-11-01");
$vine->SetDayOffset(0.5);
$vine->title->Set("2001-11-01");
$vine->title->SetFont(FF_FONT1,FS_BOLD,10);
$graph->Add($vine);

// Setup a milestone
$ms = new MileStone(6,"M5","2001-11-28","28/12");
$ms->title->SetFont(FF_FONT1,FS_BOLD);
$graph->Add($ms);

// And to show that you can also add an icon we add
// "Tux"
$icon = new IconPlot('penguin.png',0.01,0.95,1,15);
$icon->SetAnchor('left','bottom');
$graph->Add($icon);

// ... and finally send it back to the browser
$graph->Stroke();

?>

```

9.12.10 ガント チャートにアイコンを追加する

これまでの画像のまま、IconPlot() インスタンスを作成し、それをグラフに追加することで、ガント チャートに小さいイメージ(または、アイコン)を追加することもできます。以下のコードは左下の端に “tux” の小さいイメージを追加します。

```

$icon = new IconPlot('penguin.png',0.01,0.95,1,15);
$icon->SetAnchor('left','bottom');
$graph->Add($icon);

```

9.12.11 自動サイズ調整で余白を調整する

バージョン 1.17 の時点においては、縦の高さが自動的に決定された場合であっても、Graph::SetMargin() でガント チャートの余白を指定することができます。たとえば、左、右、または下に余白のないグラフを生成するには、以下のコードが必要です。

```

$graph = new GanttGraph(500);
$graph->SetMargin(0,0,30,0);

```

9.13 ガント チャート作成の簡略化

これまでの例で示されるように、ガント チャートを構成するものの多くは同じような繰り返しのタスクから成り立っています。つまり、個々のアクティビティ バーを作成し、グラフへそれらを加えるという処理です。

この基本的な作動法が理解できたとき、あるヘルパー関数に感謝するはずです。[GanttGraph::CreateSimple\(\)](#)この関数は、ガントチャートとこのチャートを構成するデータを格納する配列を必要とします。この方法を使用すると、グラフ作成の簡略化のためにいくつかの調整は省かれます。この方法は、魔法のようなものでもなんでもなく、ただアクティビティに対するデータ(スタートや、終了日、タイトル、経過、制限など)を得て、アクティビティを組み立て、それらをグラフへと追加するだけです。

ここで作成するアクティビティは、次のようなフィールドをもったデータ配列で指定します。

- アクティビティの格納される配列のキーワード
- アクティビティタイプ 外見を調整する。これは以下からひとつ選択する。
 1. ACTYPE_NORMAL 標準的なアクティビティバー
 2. ACTYPE_GROUP グループ化されたバー
 3. ACTYPE_MILESTONE マイルストーン
 - タイトル文字列
 - スタート日
 - 終了日(マイルストーンに対するものではない)
 - 見出し

例えば、グループ化された 2 つのアクティビティとマイルストーンからなるガントチャートを作成するためにには、次のようなコードを使用する必要があります。

```
$data = array(  
    array(0,ACTYPE_GROUP, "Phase 1", "2001-10-26","2001-11-23","",""),  
    array(1,ACTYPE_NORMAL, " Label 2", "2001-10-26","2001-11-13","[KJ]"),  
    array(2,ACTYPE_NORMAL, " Label 3", "2001-11-20","2001-11-22","[EP]"),  
    array(3,ACTYPE_MILESTONE, " Phase 1 Done", "2001-11-23","M2" ));  
  
    // 基本的なグラフを作成する  
    $graph = new GanttGraph();  
    $graph->title->Set("Gantt Graph using CreateSimple()");  
  
    // スケールを設定する  
    $graph->ShowHeaders(GANTT_HYEAR | GANTT_HMONTH | GANTT_HDAY | GANTT_HWEEK);  
    $graph->scale->week->SetStyle(WEEKSTYLE_FIRSTDAY);  
  
    // 特別なアクティビティを加える  
    $graph->CreateSimple($data);  
  
    // グラフを出力する  
    $graph->Stroke();
```

この結果は、以下のようになります。

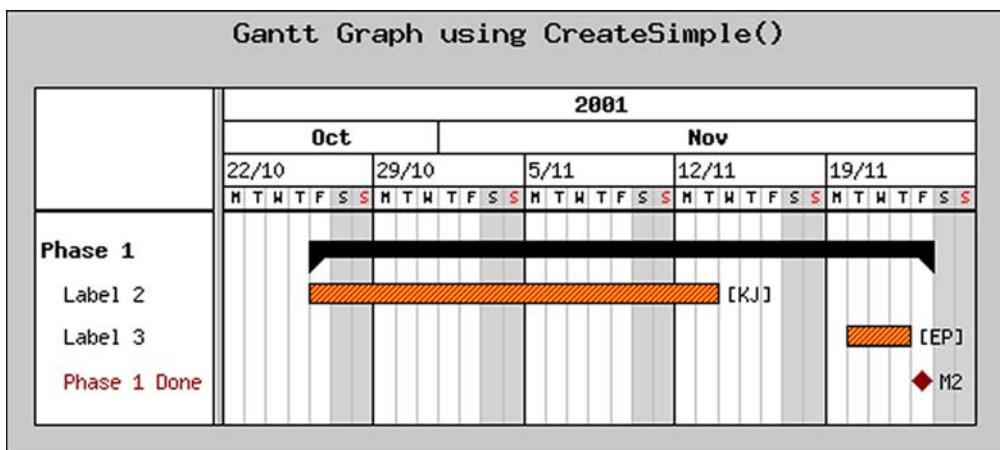


図 170: CreateSimple()メソッドによって簡略化した方法を使用する

```
<?php
// Gantt example
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

// 
// The data for the graphs
//
$data = array(
    array(0,ACTYPE_GROUP,      "Phase 1",        "2001
-10-26", "2001-11-23", ''),
    array(1,ACTYPE_NORMAL,     "Label 2",        "2001
-10-26", "2001-11-13", '[KJ]'),
    array(2,ACTYPE_NORMAL,     "Label 3",        "2001
-11-20", "2001-11-22", '[EP]'),
    array(3,ACTYPE_MILESTONE, "Phase 1 Done", "2001
-11-23", 'M2' ):
```

```
// Create the basic graph
$graph = new GanttGraph();
$graph->title-
>Set("Gantt Graph using CreateSimple()");

// Setup scale
$graph-
>ShowHeaders(GANTT_HYEAR | GANTT_HMONTH | GANTT_HD
AY | GANTT_HWEEK);
$graph->scale->week->SetStyle(WEEKSTYLE_FIRSTDAY);

// Add the specified activities
$graph->CreateSimple($data);

// ... and stroke the graph
$graph->Stroke();

?>
```

[GanttGraph::SetSimpleFont\(\)](#) メソッドや [GanttGraph::SetSimpleStyle\(\)](#) メソッドを用いて、簡単なガントチャートの外観を(すこしだけ)修正することもできます。しかし、簡単にグラフを構築するこの方法の目的は、ほかでもない作成の簡略化であるということを思い出して下さい。もしかなり高度な調整が必要であるようならば、通常の方法ですべてのアクティビティを構築する必要があります。

また、GanttGraph::CreateSimple()に付加的なデータ配列を与えることで、制限やそれぞれのバーに対する進捗状況を指定することもできます。

9.14 ガント チャートに CSIM (クライアントサイド イメージ マップ)を追加する

ガントチャートは、アクティビティ内や、それぞれのアクティビティに関連するラベル内に独立した(異なったターゲットの)ホット スポットを持つことが出来ます。

[GanttBar::SetCSIMTarget\(\)](#) や [GanttBar::SetCSIMAlt\(\)](#)を呼び出して、アクティビティバーにターゲットやそれに関連する“Alt”テキストを指定します。

同様の方法で、次のコードのように、アクティビティ タイトルに対してターゲットや Alt テキストを設定します。

```
$bar->SetCSIMTarget("http://localhost/abc/");
$bar->SetCSIMAlt("Alt Text for the bar");
$title->SetCSIMTarget("http://localhost/abc");
$title->SetCSIMAlt("Alt Text for the title");
```

上述のコードは、あなたのアクティビティが変数バーとして使用されることを仮定しています。example ディレクトリの中に、ガント チャートを CSIM を用いて使用する方法を示したサンプルファイル”ganttcsimex1.php” があります。(この例が、CreateSimple() メソッドを使用して簡単化されたガント チャートの例であることに注意してください。)

9.15 アクティビティ間に制限を追加する

ガント チャートを使用する際に、1つのアクティビティやいくつかのアクティビティ間の制限を示すことがしばしば必要となります。最もよく用いられる制限の1つは、アクティビティは別のアクティビティが終了していないと開始できないというものです。

JpGraph は、以下のような制限のタイプの視覚化をサポートしています。

- Start to End
- Start to Start
- End to Start
- End to End

2つのアクティビティ間の制限を指定する例を説明していきます。

ガント スキーマを以下のように開始することを前提としています。

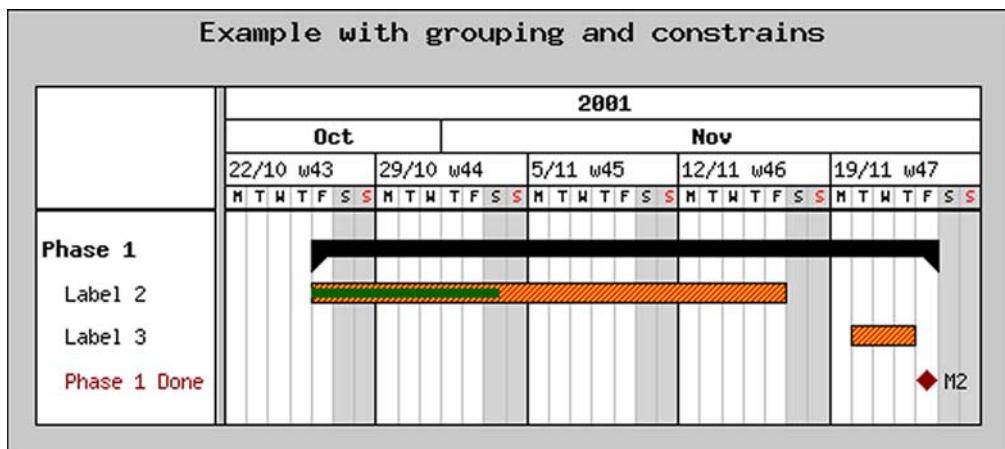


図 171: 制限を加えたいオリジナルのガントスキーマ

```
<?php
// Gantt example
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

// The data for the graphs
//
$data = array(
    array(0,ACTYPE_GROUP, "Phase 1", "2001-10-26","2001-11-23"),
    array(1,ACTYPE_NORMAL, "Label 2", "2001-10-26","2001-11-16"),
    array(2,ACTYPE_NORMAL, "Label 3", "2001-11-20","2001-11-22"),
    array(3,ACTYPE_MILESTONE, "Phase 1 Done", "2001-11-23",'M2');

// The constrains between the activities
// $constraints = array(array(1,2,CONSTRAIN_ENDSTART),
//                      array(2,3,CONSTRAIN_STARTSTART));
$constraints = array();

$progress = array(array(1,0.4));

// Create the basic graph
$graph = new GanttGraph();
$graph->title-
```

```
>Set("Example with grouping and constrains");

// Setup scale
$graph-
>ShowHeaders(GANTT_HYEAR | GANTT_HMONTH | GANT
T_HDAY | GANTT_HWEEK);
$graph->scale->week-
>SetStyle(WEEKSTYLE_FIRSTDAYWNBR);

// Add the specified activities
$graph->CreateSimple($data,$constraints,$progress);

// .. and stroke the graph
$graph->Stroke();

?>
```

ここで、アクティビティ”Label 3”はアクティビティ”Label 2”を終了させないと開始することができず、さらに”Phase 1 done”というマイルストーンはアクティビティ”Label 3”が行われた期日に依存するという制限を加えようとしています。

制限を加える際の決まりとしては、制限を与えたいたいアクティビティのそれぞれに対して、この制限が加えられるべき他のアクティビティが何かということを教えてあげなければならないということです。その他のアクティビティは、アクティビティがどの列で存在しているかを教えることで指定されます。たとえば、Start-to-End というような制限のタイプに応じて、矢印は正しい方法で 2 つのアクティビティを接続するでしょう。

制限を加えるには、アクティビティ上で [SetConstrain\(\)](#) メソッドを呼びだします。この方法では、制限のタイプを指定することに加え、この制限が加えられるべき他のアクティビティが何かということを指定します。クラスリファレンスを読むと、使用した矢印のタイプやサイズを指定することができるることも分かるでしょう。ここでは、デフォルトのサイズやタイプのみを使用しています。

例えば、”Label 2” と”Label 3”の間に End-To-Start 制限を加えるためには、以下のように記述します。

```
$bar2->SetConstrain(2,CONSTRAIN_ENDSTART)
```

上記の呼び出しの際の最初のパラメータ ”2” は、ターゲットアクティビティの行番号です。（”Label 3”アクティビティの行）以下の例は、制限を加えた結果を示したものです。

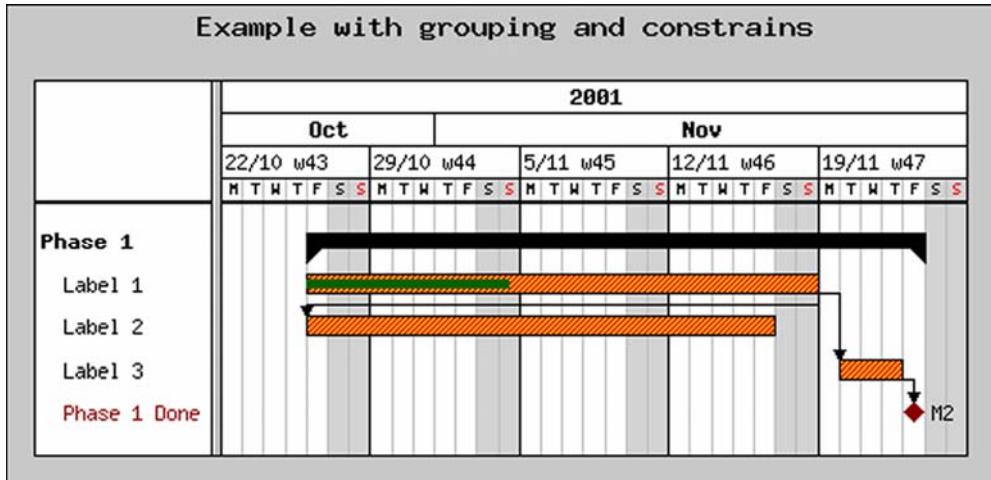


図 172:ガント チャートに制限を追加

```
<?php
// Gantt example
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

// // The data for the graphs
// 
$data = array(
array(0,ACTYPE_GROUP, "Phase 1", "200
```

```
1-10-26","2001-11-23"),  
    array(1,ACTYPE_NORMAL, " Label 1", "2001  
-10-26","2001-11-18"),  
    array(2,ACTYPE_NORMAL, " Label 2", "2001  
-10-26","2001-11-16"),  
    array(3,ACTYPE_NORMAL, " Label 3", "2001  
-11-20","2001-11-22"),  
    array(4,ACTYPE_MILESTONE," Phase 1 Done", "200  
1-11-23",'M2') );  
  
// The constrains between the activities  
$constraints = array(array(1,2,CONSTRAIN_ENDSTART),  
                     array(1,3,CONSTRAIN_STARTSTART),  
                     array(3,4,CONSTRAIN_STARTSTART));  
  
$progress = array(array(1,0.4));  
  
// Create the basic graph  
  
$graph = new GanttGraph();  
$graph->title->  
>Set("Example with grouping and constraints");  
  
// Setup scale  
$graph->  
>ShowHeaders(GANTT_HYEAR | GANTT_HMONTH | GANT  
T_HDAY | GANTT_HWEEK);  
$graph->scale->week-  
>SetStyle(WEEKSTYLE_FIRSTDAYWNBR);  
  
// Add the specified activities  
$graph->CreateSimple($data,$constraints,$progress);  
  
// .. and stroke the graph  
$graph->Stroke();  
  
?>
```

注意:矢印によって示される実際のパスは、どんな制限であるかを明確にするためにいくつかのヒューリスティックスにより調節されています。API をシンプルにするために、示された実際のパスをユーザーがさらに詳細に制御しないということが、当初の設計でした。しかし、今後のバージョンではそのヒューリスティックスが拡張され、ユーザー定義のパラメータが使用可能となるかもしれません。

9.16 より高度なフォーマット

9.16.6 ガント チャートの一部分のみを表示する

GanttGraph::SetDateRange() メソッドを用いて日付の範囲を明示的に指定することにより、指定した範囲のみを表示することが可能です。パラメータとして与えられた開始日から終了日の間でのみ表示されるように制限します。例えば、以下のように指定します。

```
$graph->SetDateRange("2001-12-20","2002-01-20");
```

この例では、2001 年 12 月 20 日 から 2002 年 1 月 20 日 の間のガントチャートの部分のみが表示されます。そのフォーマットは、ロケールの設定に依存することに注意してください。

9.16.7 週の開始曜日を指定する

GanttScale::SetWeekStart() を呼び出して、週の開始曜日を設定することができます。この方法は、週の開始曜日を表す引数として 0 から 6 の整数をとります。例えば、0 は日曜日を意味し、1 は月曜日、2 は火曜日などです。デフォルトでは、月曜日が週の開始曜日となります。

9.17 ローカライズ

PHP のインストール設定によっては、いくつかのロケールをサポートする場合があります。デフォルトでは、ロケールはサーバー上のデフォルトロケールを使用するように設定されています。

具体的に言うと、あるロケールに設定するためにはロケール文字列を用いて設定したいロケールを指定します。例えば、アメリカ英語であれば 'EN_US' と指定し、イギリス英語であれば 'EN_UK'、オランダ語であれば 'nl_NL' などです。現在のインストール設定が指定したロケールをサポートしていないような場合、エラー メッセージが表示されます。

```
$graph->scale->SetDateLocale("se_SE");
```

出力結果は以下のようになります。

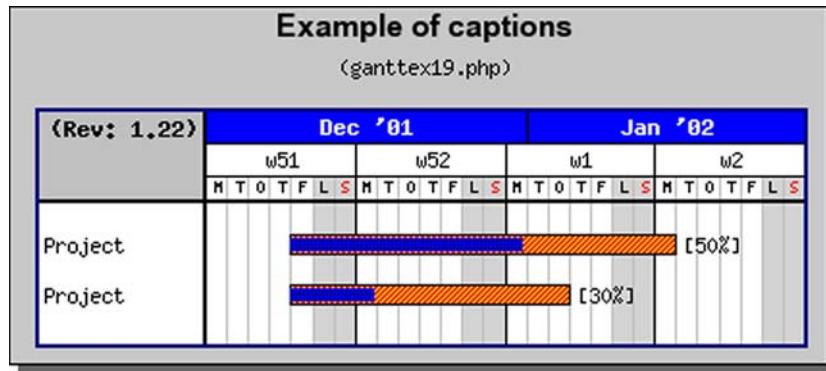


図 173:スウェーデンロケールを使用する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_gantt.php");

$graph = new GanttGraph(0,0,"auto");
$graph->SetBox();
$graph->SetShadow();

// Use swedish locale
$graph->scale->SetDateLocale("sv_SE");

// Add title and subtitle
$graph->title->Set("Example of captions");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,12);
$graph->subtitle->Set("(ganttex19.php)");

// Show day, week and month scale
$graph-
>ShowHeaders(GANTT_HDAY | GANTT_HWEEK | GANTT_HMONTH);

// Set table title
$graph->scale->tableTitle->Set("(Rev: 1.22)");
$graph->scale->tableTitle->SetFont(FF_FONT1,FS_BOLD);
$graph->scale->SetTableTitleBackground("silver");
$graph->scale->tableTitle->Show();

$graph->scale->divider->SetStyle('solid');
$graph->scale->divider->SetWeight(2);
$graph->scale->divider->SetColor('black');

$graph->SetBox(true,"navy",2);

// Use the short name of the month together with a
2 digit year
// on the month scale
$graph->scale->month-
>SetStyle(MONTHSTYLE_SHORTNAMEYEAR2);
$graph->scale->month->SetFontColor("white");

$graph->scale->month->SetBackgroundColor("blue");
// 0 % vertical label margin
$graph->SetLabelVMarginFactor(1);

// Format the bar for the first activity
// ($row,$title,$startdate,$enddate)
$activity = new GanttBar(0,"Project","2001-12-21","2002-01-07","[50%]");

// Yellow diagonal line pattern on a red background
$activity->SetPattern(BAND_RDIAG,"yellow");
$activity->SetFillColor("red");

// Set absolute height
$activity->SetHeight(10);

// Specify progress to 60%
$activity->progress->Set(0.6);
$activity->progress->SetPattern(BAND_HVCROSS,"blue");

// Format the bar for the second activity
// ($row,$title,$startdate,$enddate)
$activity2 = new GanttBar(1,"Project","2001-12-21","2002-01-02","[30%]");

// Yellow diagonal line pattern on a red background
$activity2->SetPattern(BAND_RDIAG,"yellow");
$activity2->SetFillColor("red");

// Set absolute height
$activity2->SetHeight(10);

// Specify progress to 30%
$activity2->progress->Set(0.3);
$activity2->progress-
>SetPattern(BAND_HVCROSS,"blue");

// Finally add the bar to the graph
$graph->Add($activity);
$graph->Add($activity2);
```

```
// Add a vertical line
$vline = new GanttVLine("2001-12-24","Phase 1");
$vline->SetDayOffset(0.5);
// $graph->Add($vline);
```

```
// ... and display it
$graph->Stroke();
?>
```

9.18 JpGraph のアンチエイリアス

JpGraph バージョン 1.2 からアンチエイリアスによるラインの描画をサポートしています。このセクションで説明するアンチエイリアスを使用するにはいくつかの注意点があります。

注意: アンチエイリアスの条件の性質上、アンチエイリアスは水平、垂直、または、45 度のラインの場合は使用できません。

9.18.6 アンチエイリアスを有効にする

アンチエイリアスを使用したいスクリプトで [Image](#) クラスの [SetAntiAliasing\(\)](#) メソッドを呼ぶことでアンチエイリアスを有効にします。

背景カラーとライン カラーの間の色を段階的に補完していくことで、アンチエイリアスはより滑らかに線の縁を描くことができます。

注意: アンチエイリアスに使用されるアルゴリズムはとてもシンプルです。実際の 2D のシグナルを処理することで、さらに良い結果を得ることができます。しかし、HTTP サーバー上で実際の 2D のシグナルを処理するのにはばかばかしいので、デザインはわざとシンプルに設計してあります。最も良のビジュアルを得るために明るい背景上で暗いライン カラーを使用するようにしてください。

以下の例は、このとてもシンプルなアルゴリズムが正確に動作することを示しています。ここではアンチエイリアスを有効にしたレーダー プロットと無効にしたレーダー プロットを表示します。

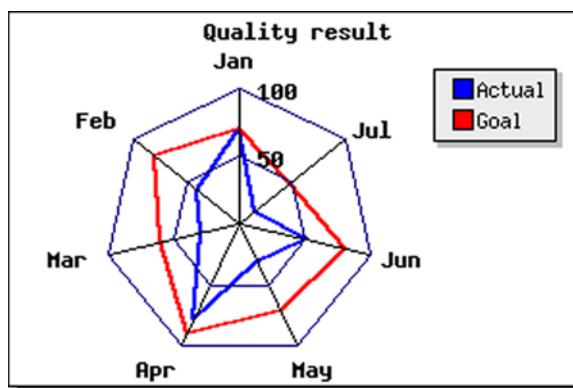


図 174: アンチエイリアスを無効にしたレーダープロット

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_radar.php");
```

```
// Create the basic radar graph
$graph = new RadarGraph(300,200,"auto");
// $graph->img->SetAntiAliasing();
```

```

// Set background color and shadow
$graph->SetColor("white");
$graph->SetShadow();

// Position the graph
$graph->SetCenter(0.4,0.55);

// Setup the axis formatting
$graph->axis->SetFont(FF_FONT1,FS_BOLD);

// Setup the grid lines
$graph->grid->SetLineStyle("solid");
$graph->grid->SetColor("navy");
$graph->grid->Show();
$graph->HideTickMarks();

// Setup graph titles
$graph->title->Set("Quality result");
$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->SetTitles($gDateLocale->GetShortMonth());

// Create the first radar plot

```

```

$pplot = new RadarPlot(array(70,80,60,90,71,81,47));
$pplot->SetLegend("Goal");
$pplot->SetColor("red","lightred");
$pplot->SetFill(false);
$pplot->SetLineWeight(2);

// Create the second radar plot
$pplot2 = new RadarPlot(array(70,40,30,80,31,51,14));
$pplot2->SetLegend("Actual");
$pplot2->SetLineWeight(2);
$pplot2->SetColor("blue");
$pplot2->SetFill(false);

// Add the plots to the graph
$graph->Add($pplot2);
$graph->Add($pplot);

// And output the graph
$graph->Stroke();

?>

```

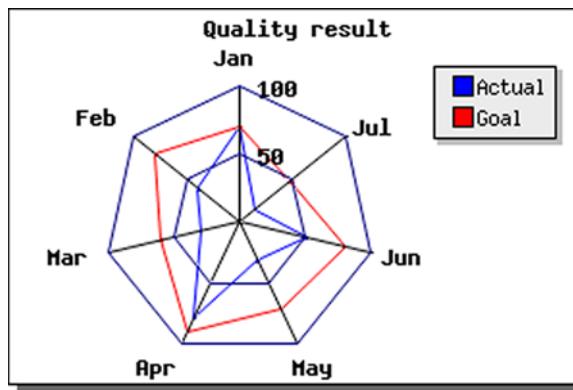


図 175: アンチエイリアスを有効にしたレーダープロット

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_radar.php");

// Create the basic radar graph
$graph = new RadarGraph(300,200,"auto");
$graph->img->SetAntiAliasing();

// Set background color and shadow
$graph->SetColor("white");
$graph->SetShadow();

// Position the graph
$graph->SetCenter(0.4,0.55);

// Setup the axis formatting
$graph->axis->SetFont(FF_FONT1,FS_BOLD);

// Setup the grid lines
$graph->grid->SetLineStyle("solid");
$graph->grid->SetColor("navy");
$graph->grid->Show();
$graph->HideTickMarks();

// Setup graph titles
$graph->title->Set("Quality result");

```

```

$graph->title->SetFont(FF_FONT1,FS_BOLD);

$graph->SetTitles($gDateLocale->GetShortMonth());

// Create the first radar plot
$pplot = new RadarPlot(array(70,80,60,90,71,81,47));
$pplot->SetLegend("Goal");
$pplot->SetColor("red","lightred");
$pplot->SetFill(false);
$pplot->SetLineWeight(2);

// Create the second radar plot
$pplot2 = new RadarPlot(array(70,40,30,80,31,51,14));
$pplot2->SetLegend("Actual");
$pplot2->SetLineWeight(2);
$pplot2->SetColor("blue");
$pplot2->SetFill(false);

// Add the plots to the graph
$graph->Add($pplot2);
$graph->Add($pplot);

// And output the graph
$graph->Stroke();

?>

```

アンチエイリアスを使用すると画像の生成に膨大な時間を費やしてしまう可能性があることを覚えておいてください。アンチエイリアスで線を描く場合、通常よりも約 8 倍ほど遅くなるので、この機能は慎重に使用してください。

さらに、アンチエイリアスの使用上の注意について、いくつか説明したいと思います。

1. アンチエイリアスは使用可能なカラー パレット以上の色数を消費します。使用される色の正確な数は線の角度に応じて変わり、水平に近い角度の線と垂直に近い角度の線の場合はかなり多くの色数を使用します(異なる角度を持つ線の数が多くなると、さらに多くなります)。カラーグラデーションと一緒にアンチエイリアスを使用する場合、使用可能なパレットの数が足りなくなり、アンチエイリアスを使用できなくなる可能性があります。通常のパレットは 256 前後の色をキープすることができます。これは、アンチエイリアスを使用する場合は、true カラーイメージを使用したほうがよいということです。
2. アンチエイリアスはそれぞれの線の端を同じ色で塗りつぶすので、背景イメージを同時に表示させる場合はうまく動作しません。より高度なアンチエイリアスの処理を行うと、単純にプロセッサのパワーが必要となります。
3. アンチエイリアス処理を行うと線の幅を指定しても無視されます。線の幅は常におおよそ 1 で表示されます。

9.19 グラフを回転させる

JpGraph は生成されたグラフを自由な角度で回転させることができます。これは実際のグラフ(軸、軸タイトル、ラベルなど)と、タイトルやフッターのような固定されていないエレメントにだけ影響します。

おそらくほとんどの場合、回転はグラフを 90 度回転するために使われます。たとえば、バープロットを水平に表示したい場合などです。

パフォーマンス ノート: 回転処理を追加すると、画像を出力する前にグラフの全てのポイントが変換処理を行うので、グラフの生成が遅くなります。JpGraph は予め計算された変換マトリックスを使用することでこの変換処理を効率的に行います。

デフォルトでは、回転の中心は画像全体の中心点が含まれるプロットエリアの中心点となります。回転をコントロールするには、2 つのメソッドを使用します。

- [Graph::image::SetAngle\(\)](#) 回転する角度を指定する。
- [Graph::image::SetCenter\(\)](#) ピクセルの絶対値で回転の中心を指定する。

使用例は以下のようになります。

```
$graph->image->SetAngle(45);
```

実際には、回転後にグラフを変換できる 3 つ目の関数があります。

これは、おそらくあまり使われないメソッドなので、これ以上説明はしませんが、詳しく知りたい場合は [Graph::image::SetTranslation\(\)](#) のクラス リファレンスを参照してください。

イメージを回転するときは、各軸のラベルは回転しないということに注意してください。

このように設計されたのは以下の 2 つの理由からです。

- a) ビット マップのフォントは回転できない
- b) 読みやすさを保つため

[Axis::SetLabelAngle\(\)](#) メソッドでラベルを回転することもできます。

デフォルトでは、ラベルのアンカー ポイントは角度が 0 度の場合に最適となるので、回転したグラフを綺麗に表示させるために軸のラベルのアンカー ポイントや直線を調整したい場合があるでしょう。これはメソッド [Axis::SetLabelAlign\(\)](#) を使用することで可能になります。この方法の詳細な説明は水平バーのグラフ ([Working with bar plots](#)) を参照してください。

以下のテーブルは、どのようにして角度を回転させるか、そして角度の中心点が異なると違った効果をもたらすことを示すための例です。左上のグラフはオリジナルのイメージです。

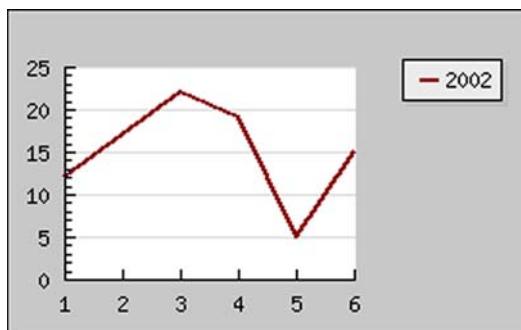


図 176: オリジナル イメージ

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

$ydata = array(12, 17, 22, 19, 5, 15);

$graph = new Graph(270, 170);
$graph->SetMargin(30, 90, 30, 30);
$graph->SetScale("textlin");

$line = new LinePlot($ydata);
$line->SetLegend('2002');
$line->SetColor('darkred');
$line->SetWeight(2);
$graph->Add($line);

// Output graph
$graph->Stroke();

?>
```

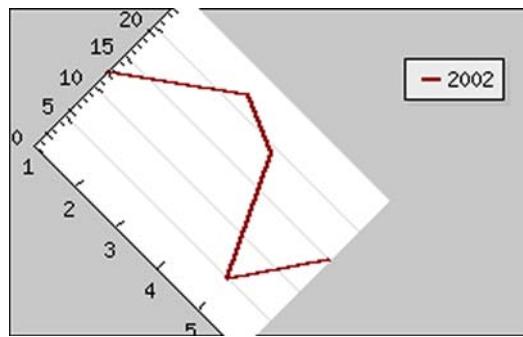


図 177: プロットエリアの中央を中心点として 45 度回転

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(12,17,22,19,5,15);

$graph = new Graph(270,170);
$graph->SetMargin(30,90,30,30);
$graph->SetScale("textlin");

$graph->img->SetAngle(45);
```

```
$line = new LinePlot($ydata);
$line->SetLegend('2002');
$line->SetColor('darkred');
$line->SetWeight(2);
$graph->Add($line);

// Output graph
$graph->Stroke();

?>
```

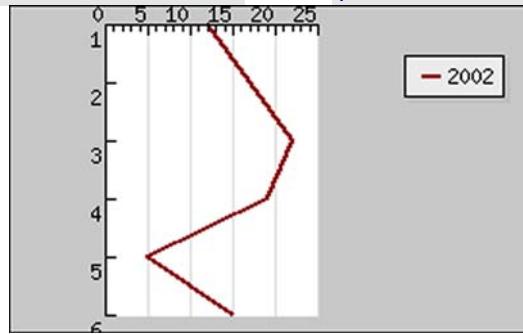


図 178: プロットエリアの中央を中心点として 90 度回転

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(12,17,22,19,5,15);

$graph = new Graph(270,170);
$graph->SetMargin(30,90,30,30);
$graph->SetScale("textlin");

$graph->img->SetAngle(90);
```

```
$line = new LinePlot($ydata);
$line->SetLegend('2002');
$line->SetColor('darkred');
$line->SetWeight(2);
$graph->Add($line);

// Output graph
$graph->Stroke();

?>
```

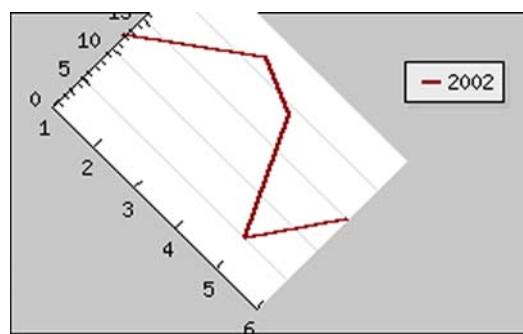


図 179: イメージの中央を中心点として 45 度回転

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(12,17,22,19,5,15);

$graph = new Graph(270,170);
$graph->SetMargin(30,90,30,30);
$graph->SetScale("textlin");

$graph->img->SetAngle(45);
$graph->img->SetCenter(floor(270/2),floor(170/2));
```

```
$line = new LinePlot($ydata);
$line->SetLegend('2002');
$line->SetColor('darkred');
$line->SetWeight(2);
$graph->Add($line);

// Output graph
$graph->Stroke();

?>
```

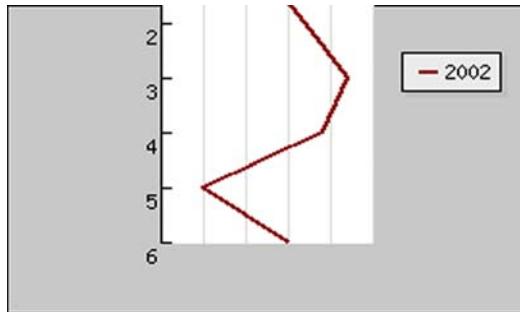


図 180: イメージの中央を中心点として 90 度回転

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(12,17,22,19,5,15);

$graph = new Graph(270,170);
$graph->SetMargin(30,90,30,30);
$graph->SetScale("textlin");

$graph->img->SetAngle(90);
$graph->img->SetCenter(floor(270/2),floor(170/2));
```

```
$line = new LinePlot($ydata);
$line->SetLegend('2002');
$line->SetColor('darkred');
$line->SetWeight(2);
$graph->Add($line);

// Output graph
$graph->Stroke();

?>
```

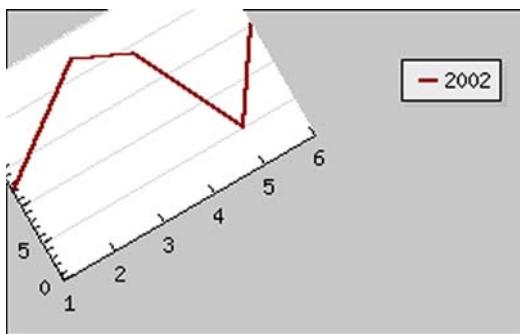


図 181: プロットエリアの左下を中心点として 30 度回転

```
<?php
include ("./jpgraph.php");
include ("./jpgraph_line.php");

$ydata = array(12,17,22,19,5,15);

$graph = new Graph(270,170);
$graph->SetMargin(30,90,30,30);
$graph->SetScale("textlin");

$graph->img->SetAngle(-30);
$graph->img->SetCenter(30,170-30);
```

```
$line = new LinePlot($ydata);
$line->SetLegend('2002');
$line->SetColor('darkred');
$line->SetWeight(2);
$graph->Add($line);

// Output graph
$graph->Stroke();

?>
```

これらの画像で示されるように、プロット エリアの中心以外のポイントを回転する場合、プロットが回転後イメージの外に配置されてしまうこともあります。

設計上画像の回転はプロットエリアにのみ影響があるので、余白のカラーが背景カラーと同じときに特に効果的です。

9.20 イメージと背景用の明るさおよびコントラストを調整する

以下のセクションはパレット イメージにのみ適応します。これは `true` カラーのイメージでは作動しないということです。

背景イメージが少し“色あせて”見えるようにしてグラフから注意をそらさないようにすることが、多くの場合は望ましいです。これを行うには 2 通りの方法があります。

1. 明るさのレベルとコントラストの強さを好みのレベルに調整する外部のイメージ エディタのあるイメージを準備して
2. コントラスト、明るさ、カラー彩度を調整する `JpGraph::s` を使用してください。

背景イメージの呼び出しを調整するため、明るさとコントラストの両方が範囲 $[-1, 1]$ での実際の数字です。たとえば、背景イメージだけ、または全体のイメージを調整するのに選ぶことができます。背景イメージ変更するには、適した値を指定するためメソッド [`Graph::AdjBackgroundImage\(\)`](#) を使用してください。このメソッドで何ができるかいくつかの例を見てみましょう。以下の例は、“utils/” ディレクトリにある小さいユーティリティ “adjimg.php” で生成されています。



Brightness=0, contrast=0, saturation = -1
(オリジナル イメージ)



Brightness=0, contrast=0, saturation = -1
(黒と白のイメージ)



Brightness=0.3, contrast=-0.3, saturation=0



Brightness=0.4, contrast=-0.7, saturation=0



Brightness=0.4, contrast=-0.7, saturation=-1



Brightness=0, contrast=0, saturation=1

9.21 グラフ生成の時間を計る

開発と最適化の間、フットノートとしてイメージを生成するにかかる実際の時間をとても手軽にします。以下の例はこの特徴の使用を表しています。

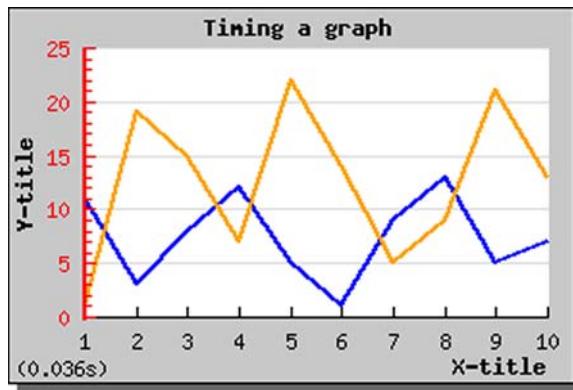


図 182:グラフのタイミング

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");

$ydata = array(11,3,8,12,5,1,9,13,5,7);
$ydata2 = array(1,19,15,7,22,14,5,9,21,13);

$gJpgBrandTiming=true;

// Create the graph. These two calls are always required
$graph = new Graph(300,200,"auto");
$graph->SetScale("textlin");

// Create the linear plot
$lineplot=new LinePlot($ydata);
$lineplot2=new LinePlot($ydata2);

// Add the plot to the graph
$graph->Add($lineplot);
$graph->Add($lineplot2);

$graph->img->SetMargin(40,20,20,40);
$graph->title->Set("Timing a graph");
$graph->xaxis->title->Set("X-title");
$graph->yaxis->title->Set("Y-title");

$graph->title->SetFont(FF_FONT1,FS_BOLD);
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD);
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD);

$lineplot->SetColor("blue");
$lineplot->SetWeight(2);

$lineplot2->SetColor("orange");
$lineplot2->SetWeight(2);

$graph->yaxis->SetColor("red");
$graph->yaxis->SetWeight(2);
$graph->SetShadow();

// Display the graph
$graph->Stroke();
?>

```

この特徴を有効にするため、2 つの方法で処理することができます。

1. グローバルの定義 BRAND_TIMING (jpgraph.php で) をトゥルーに設定する。これは生成されたすべてのグラフに時間の文字を追加します。
2. .. または、グローバル変数 \$gJpgBrandTiming を以下のように設定することで特定のグラフを有効にすることができます。

```
$gJpgBrandTiming=true;
```

スクリプトの最初に書きます。

方法を変更したい場合、BRAND_TIMING_FORMAT (jpgraph.php の) で定義された文字を設定することでタイミングはフォーマットされます。このストリングは標準の printf() フォーマット スtringingを表わします。

注意: JpGraph にはコードのタイミング部分で必要な JpgTimer と呼ばれるユーティリティクラスがあります。API は本当にシンプルです。クラスは複数回起動するタイマーをサポートし、Push() メソッドを呼ぶことでタイマーを起動します。新しいタイマーを起動し、積み重なったタイマーの上に置きます。タイマーを止めるには、積み重なったものからそれを動かし、タイミング値を返し Pop() を呼びます。

9.22 様々なコンテキストで国旗を使用する

JpGraph は 200 国の旗をサポートしています、たとえば、外部のイメージ定義なしでグラフで使用することができます。

国旗は主に 2 つの設定で使用できます。

1. ラインのイメージ記号と少量のグラフ
2. グラフの背景イメージとして
3. 任意の位置のグラフに追加できる特別なタイプのアイコン(IconPlot() を使用する)として次のセクションを見てください

適切な国旗を容易に見つけるためには、名前のすべてまたは部分、または数字のインデックスのどちらかで指定することができます。 JpGraph の手順は、特別な旗を指定する方法を見つけるのに十分“スマート”です。

マーカーとして国旗を指定するのに、MARK_FLAG1、MARK_FLAG2、MARK_FLAG3、または MARK_FLAG4 のうちどれか 1 つの特別な記号タイプを指定する必要があります。

旗は、マーク タイプの数によって示される 4 の異なるサイズに内部に格納されます。旗は、表示するとき勝手にスケールされる場合もあります。これは部分的に機能性をオーバーラップさせているので、なぜ旗が 4 つの異なる基準寸法に格納されるか非常によく尋ねるかもしれません。理由は当然パフォーマンスです。小さい旗が欲しいだけの場合、大きいイメージを小さく、小さいアイコンサイズにするのに処理時間がかかります。同時に、背景として使用される大きい旗では、小さいオリジナルのイメージを大きなサイズにする十分な詳細がありません。これが、4 つの異なるサイズに格納する理由です。

以下の例はマーカーとしての国旗の使用法を表しています。

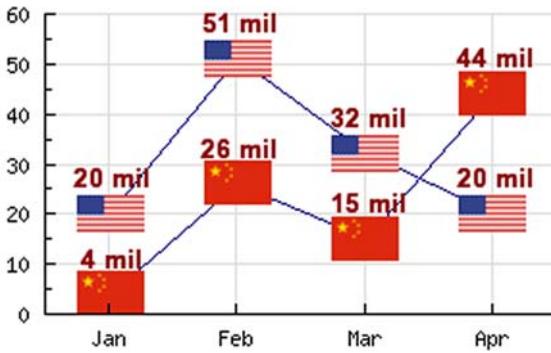


図 183: ラインプロットマーカーとして国旗を使用する

```

<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");
include ("../jpgraph_scatter.php");

$datay = array(
    array(4,26,15,44),
    array(20,51,32,20));

// Setup the graph
$graph = new Graph(300,200);
$graph->SetMarginColor('white');
$graph->SetScale("textlin");
$graph->SetFrame(false);
$graph->SetMargin(30,5,25,20);

// Enable X-grid as well
$graph->xgrid->Show();

// Use months as X-labels
$graph->xaxis->SetTickLabels($gDateLocale->GetShortMonth());

//-----
// Create the plots
//-----
$p1 = new LinePlot($datay[0]);
$p1->SetColor("navy");

// Use a flag
$p1->mark->SetType(MARK_FLAG1,'chin');

// Displays value on top of marker image
$p1->value->SetFormat('%d mil');
$p1->value->Show();
$p1->value->SetColor('darkred');
$p1->value->SetFont(FF_ARIAL,FS_BOLD,10);
// Increase the margin so that the value is printed above tje
// -----
// img marker
$p1->value->SetMargin(14);

// Incent the X-
// scale so the first and last point doesn't
// fall on the edges
$p1->SetCenter();
$graph->Add($p1);

//-----
// 2:nd plot
//-----
$p2 = new LinePlot($datay[1]);
$p2->SetColor("navy");

// Use a flag
$p2->mark->SetType(MARK_FLAG1,'united states');

// Displays value on top of marker image
$p2->value->SetFormat('%d mil');
$p2->value->Show();
$p2->value->SetColor('darkred');
$p2->value->SetFont(FF_ARIAL,FS_BOLD,10);
// Increase the margin so that the value is printed above tje
// -----
// img marker
$p2->value->SetMargin(14);

// Incent the X-
// scale so the first and last point doesn't
// fall on the edges
$p2->SetCenter();
$graph->Add($p2);

$graph->Stroke();

?>

```

背景として国旗を使うには、1つはメソッド `Graph::SetBackgroundCountryFlag()` を使用する必要があります。このメソッドで、どれくらいのイメージが詰まれ、どれくらいの旗が背景と組み合わされるかの両方を指定できます。

すべてのサポートされた国旗のリストを見るには、Example ディレクトリの “listallcountryflags.php” スクリプトを起動します。これはすべてのフラッグのテーブルを示しています。

9.23 グラフ上のアイコンを追加する

標準の背景イメージに加えて、グラフの背景に任意の数のアイコンを追加することもできます。これらのアイコンは特別なプロット クラス IconPlot で作成されます。

アイコンからのイメージはファイル、または組み込まれた国旗のうちの 1 つから取ったものです。

どれくらいのアイコンが割合 (1-100) の指定で背景に融合するかコンとロールする場合があります。下の例は、塗りつぶされたライン グラフの背景と写真 “Tux”との混み合わせの方法を現しています。
注意: この例はアルファの組み合わせを使用しているので GD2 が必要です。

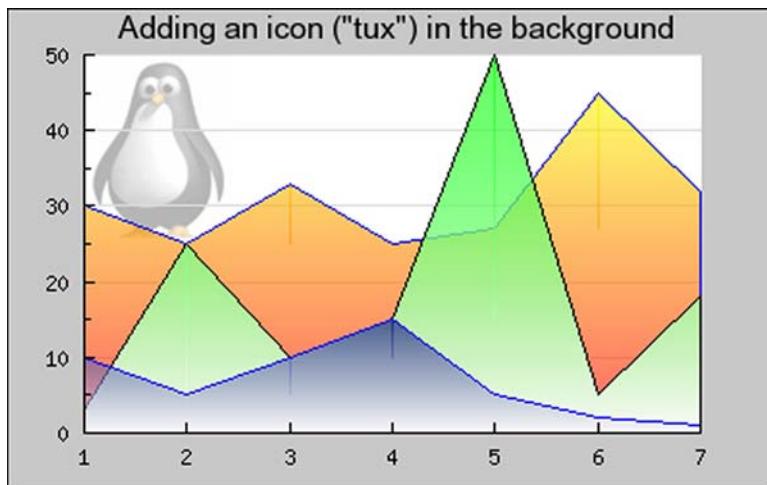


図 184: 背景にアイコンを追加する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");
include ("../jpgraph_icomplot.php");

// $datay = array(20,15,23,15,17,35,22);
$datay = array(30,25,33,25,27,45,32);
$datay2 = array(3,25,10,15,50,5,18);
$datay3 = array(10,5,10,15,5,2,1);

// Setup the graph
$graph = new Graph(400,250);
$graph->SetMargin(40,40,20,30);
$graph->SetScale("textlin");

$graph->title-
>Set('Adding an icon ("tux") in the background');
$graph->title->SetFont(FF_ARIAL,FS_NORMAL,12);

// $graph->SetBackgroundGradient('red','blue');

$graph->xaxis->SetPos('min');

$p1 = new LinePlot($datay);
$p1->SetColor("blue");
$p1->SetFillGradient('yellow@0.4','red@0.4');

$p2 = new LinePlot($datay2);
$p2->SetColor("black");
$p2->SetFillGradient('green@0.4','white');

$p3 = new LinePlot($datay3);
$p3->SetColor("blue");
$p3->SetFillGradient('navy@0.4','white@0.4');

$graph->Add($p1);
$graph->Add($p2);
$graph->Add($p3);

$icon = new IconPlot('penguin.png',0.2,0.3,1,30);
$icon->SetAnchor('center','center');
$graph->Add($icon);

// Output line
$graph->Stroke();

?>
```

アイコンとして 200 国の旗を指定するには、初め空のアイコンを指定して、適切なパラメータで IconPlot::SetCountryFlag() を呼びます。（クラス リファレンスを参照してください）。

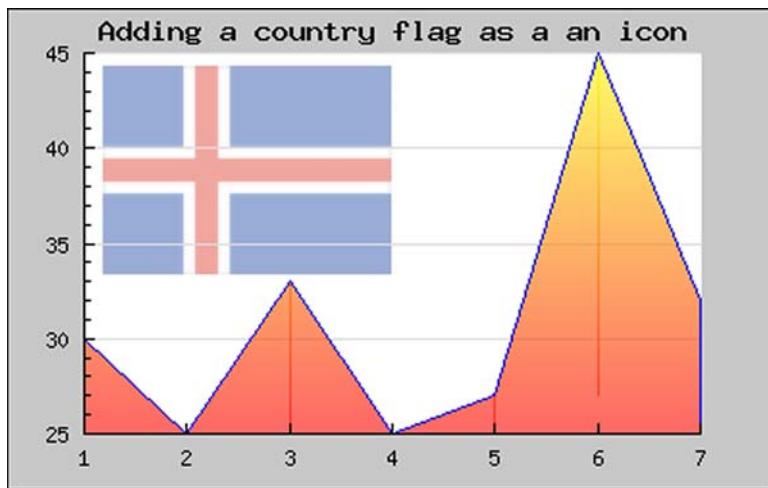


図 185: 背景に旗のアイコンを追加する

```
<?php
include ("../jpgraph.php");
include ("../jpgraph_line.php");
include ("../jpgraph_flags.php");
include ("../jpgraph_iconplot.php");

$datay = array(30,25,33,25,27,45,32);

// Setup the graph
$graph = new Graph(400,250);
$graph->SetMargin(40,40,20,30);
$graph->SetScale("textlin");

$graph->title-
>Set('Adding a country flag as a an icon');

$p1 = new LinePlot($datay);
$p1->SetColor("blue");
$p1->SetFillGradient('yellow@0.4','red@0.4');

$graph->Add($p1);

$icon = new IconPlot();
$icon->SetCountryFlag('iceland',50,30,1.5,40,3);
$icon->SetAnchor('left','top');
$graph->Add($icon);

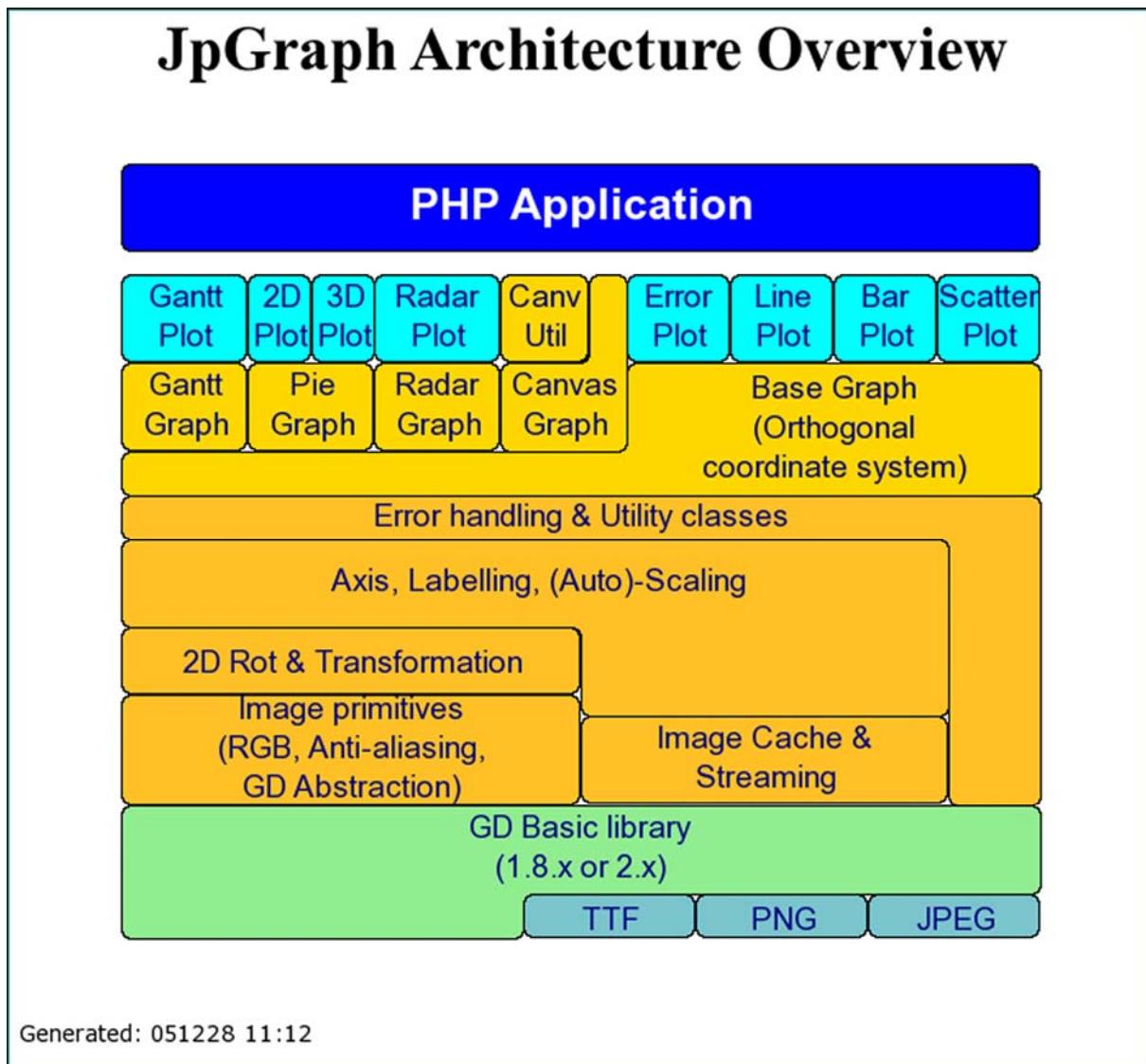
// Output line
$graph->Stroke();

?>
```

注意:内蔵 GD を使っている前 4.33 PHP のいくつかの古いバージョンは、混ざったイメージの読み込みで問題を起こします。もし、この問題を抱えたら、より新しい PHP バージョンにアップグレードする必要があります。

10 キャンバス グラフを使用する

キャンバス グラフは、より自由にグラフィックを描く必要がある場合にとても役に立つ機能です。キャンバスがどのような物かを伝えるために、下の例では JpGraph の設計構造をキャンバス グラフを用いて描いています。



Generated: 051228 11:12

図 186:キャンバスを使用した例

```
<?php
// $Id: canvas_jngarchex.php,v 1.3 2002/08/29 10:14:19 aditus Exp $
include "../jpgraph.php";
include "../jpgraph_canvas.php";
include "../jpgraph_canvtools.php";

// Scale we are using
$ymax=24;
$xmax=20;

// Setup the basic canvas
$g = new CanvasGraph(700,650,'auto');
$g->SetMargin(2,3,2,3);
$g->SetMarginColor("teal");
$g->InitFrame();

// ... and a scale
$scale = new CanvasScale($g);
$scale->Set(0,$xmax,0,$ymax);

// ... we need shape since we want the indented rectangle
```

```
$shape = new Shape($g,$scale);
$shape->SetColor('black');

// ... basic parameters for the overall image
$l = 2;           // Left margin
$r = 18;          // Row number to start the lowest line on
$width = 16;       // Total width

// Setup the two basic rectangle text object we will use
$tt = new CanvasRectangleText();
$tt->SetFont(FF_ARIAL,FS_NORMAL,14);
$tt->SetFillColor('');
$tt->SetColor('');
$tt->SetFontColor('navy');

$t = new CanvasRectangleText();
$t->SetFont(FF_ARIAL,FS_NORMAL,14);
$t->SetFillColor('goldenrod1');
$t->SetFontColor('navy');
```

キャンバス グラフを使用する

```
// Now start drawing the arch overview from the bott  
om and up  
// This is all pretty manual and one day I will write a  
proper  
// framework to make it easy to construct these type  
s of architecture  
// overviews. But for now, just plain old coordinates..  
  
// Line: GD Library and image libraries  
$h=3;  
$s = 3; $d=$l + $width-9;  
$t->SetFillColor('cadetblue3');  
$t->Set("TTF",$d,$r+2,$s,1);  
$t->Stroke($g->img,$scale);  
$t->Set("PNG",$d+$s,$r+2,$s,1);  
$t->Stroke($g->img,$scale);  
$t->Set("JPEG",$d+2*$s,$r+2,$s,1);  
$t->Stroke($g->img,$scale);  
$shape-  
>IndentedRectangle($l,$r,$width,$h,$s*3,1,2,'lightgreen');  
$t-  
>Set("GD Basic library\n(1.8.x or 2.x)",$l,$r,$width,$h-  
1);  
$tt->Stroke($g->img,$scale);  
  
// Area: Basic internal JpGraph architecture  
$t->SetFillColor('goldenrod1');  
$h = 2;  
$r -= $h; $d=8;  
$t->Set("Image primitives\n(RGB, Anti-  
aliasing,\nGD Abstraction)",$l,$r-0.5,$width*0.5,$h+0.5);  
$t->Stroke($g->img,$scale);  
$t-  
>Set("Image Cache &\nStreaming",$l+0.5*$width,$r,$wid-  
th*0.4,$h);  
$t->Stroke($g->img,$scale);  
  
$r -= $h; $d=8;  
$t->Set("2D Rot & Transformation",$l,$r,$width*0.5,$h-  
0.5); $t->Stroke($g->img,$scale);  
  
$r -= 2; $h = 4;  
$shape-  
>IndentedRectangle($l,$r,$width*0.9,$h,$d,2,3,'goldenrod1');  
$tt->Set("Axis, Labelling, (Auto)-  
Scaling",$l,$r,$width*0.9,$h-2); $tt->Stroke($g->  
img,$scale);  
  
$r -= 1;  
$shape-  
>IndentedRectangle($l,$r,$width,7,$width*0.9,6,3,'goldenro-  
d1');  
$tt-  
>Set("Error handling & Utility classes",$l,$r,$width,1); $t-  
>Stroke($g->img,$scale);  
  
// Area: Top area with graph components  
$t->SetFillColor('gold1');  
$r -= 3;  
$w = $width*0.55/4; $h = 2;  
$t->Set("Gantt\nGraph",$l,$r,$w,$h);  
$t->Stroke($g->img,$scale);  
  
$t->Set("Pie\nGraph",$l+$w,$r,$w,$h);  
$t->Stroke($g->img,$scale);  
  
$t->Set("Radar\nGraph",$l+$w*2,$r,$w,$h);  
$t->Stroke($g->img,$scale);  
  
$shape-  
>IndentedRectangle($l,$r,$width,3.4*$w,2,0,'gold1');  
$tt-  
>Set("Base Graph\n(Orthogonal\ncoordinate system)",$l  
+$w*4,$r,$width-$w*4,3);  
$tt->Stroke($g->img,$scale);  
  
$r -= 2;  
$d = 0.7;  
$shape-  
>IndentedRectangle($l+3*$w,$r,$w,4, $w*$d,2,0,'gold1');  
$t->Set("Canvas\nUtil",$l+3*$w,$r,$w*$d,$h); $t-  
>Stroke($g->img,$scale);  
$tt->Set("Canvas\nGraph",$l+3*$w,$r+2,$w,2); $tt-  
>Stroke($g->img,$scale);  
  
// Top line of plotting plugins  
$t->SetFillColor('cyan');  
$t->Set("Gantt\nPlot",$l,$r,$w,$h); $t->Stroke($g-  
>img,$scale);  
$t->Set("2D\nPlot",$l+$w,$r,$w/2,$h); $t-  
>Stroke($g->img,$scale);  
$t->Set("3D\nPlot",$l+$w+$w/2,$r,$w/2,$h); $t-  
>Stroke($g->img,$scale);  
$t->Set("Radar\nPlot",$l+2*$w,$r,$w,$h); $t-  
>Stroke($g->img,$scale);  
  
$wp = ($width - 4*$w)/4;  
$t->Set("Error\nPlot",$l+4*$w,$r,$wp,$h); $t-  
>Stroke($g->img,$scale);  
$t->Set("Line\nPlot",$l+4*$w+$wp,$r,$wp,$h); $t-  
>Stroke($g->img,$scale);  
$t->Set("Bar\nPlot",$l+4*$w+2*$wp,$r,$wp,$h); $t-  
>Stroke($g->img,$scale);  
$t-  
>Set("Scatter\nPlot",$l+4*$w+3*$wp,$r,$wp,$h); $t-  
>Stroke($g->img,$scale);  
  
// Show application top  
$r -= 2.5; $h=2;  
$t->SetFillColor('blue');  
$t->SetFontColor('white');  
$t->SetFont(FF_ARIAL,FS_BOLD,20);  
$t->Set("PHP Application",$l,$r,$width,$h); $t-  
>Stroke($g->img,$scale);  
  
// Stroke title  
$r = 0.5;  
$tt->SetFontColor('black');  
$tt->SetFont(FF_TIMES,FS_BOLD,28);  
$tt-  
>Set("JpGraph Architecture Overview",$l,$r,$width,1);  
$tt->Stroke($g->img,$scale);  
  
// Stroke footer  
$tt->SetFont(FF_VERDANA,FS_NORMAL,10);  
$tt-  
>Set("Generated: ".date("ymd H:m",time()),0.1,$ymax*0.  
95);  
$tt->Stroke($g->img,$scale);  
  
// .. and stream it all back  
$g->Stroke();  
  
?>
```

キャンバスを使用するには、プログラミングやチューニングだけではなく、JpGraph に関する深い知識も必要となります。

10.1 キャンバス グラフの使用方法

キャンバス グラフは実際にはグラフではありません。それはさまざまな形状を自由に描画することができる一枚の紙で、JpGraph の便利な機能を使用することもできます。

キャンバスはいくつかの違った方法で使用することができます。例えば、描画のための多くの基本要素を持つ `Image` クラスではそのまま動作させることが可能ですが、ピクセル座標の絶対値を使う必要があります。

キャンバス スケールを使用することで、少し使いやすくすることもできます。自分で指定したグリッド上で作動させることによって、キャンバスにより簡単に独自のスケールを定義することができます。スケールを変えるだけで、画像をとても簡単にスケールし直すこともできます。例えば、描画のサイズを半分にするには、ちょうど 2 倍のスケールを作ってください。

異なるキャンバスでの動作を助けるために、キャンバスを使用する際は “`jpgraph_canvtools.php`” ファイルをスクリプト上で呼び出してください。これは厳密には必要ありませんが、よい画像を作成する手助けになるでしょう。

キャンバスに関する別の例として、JpGraph で利用できるフォント スタイルのリストをキャンバスを使用して下の図に示しました。

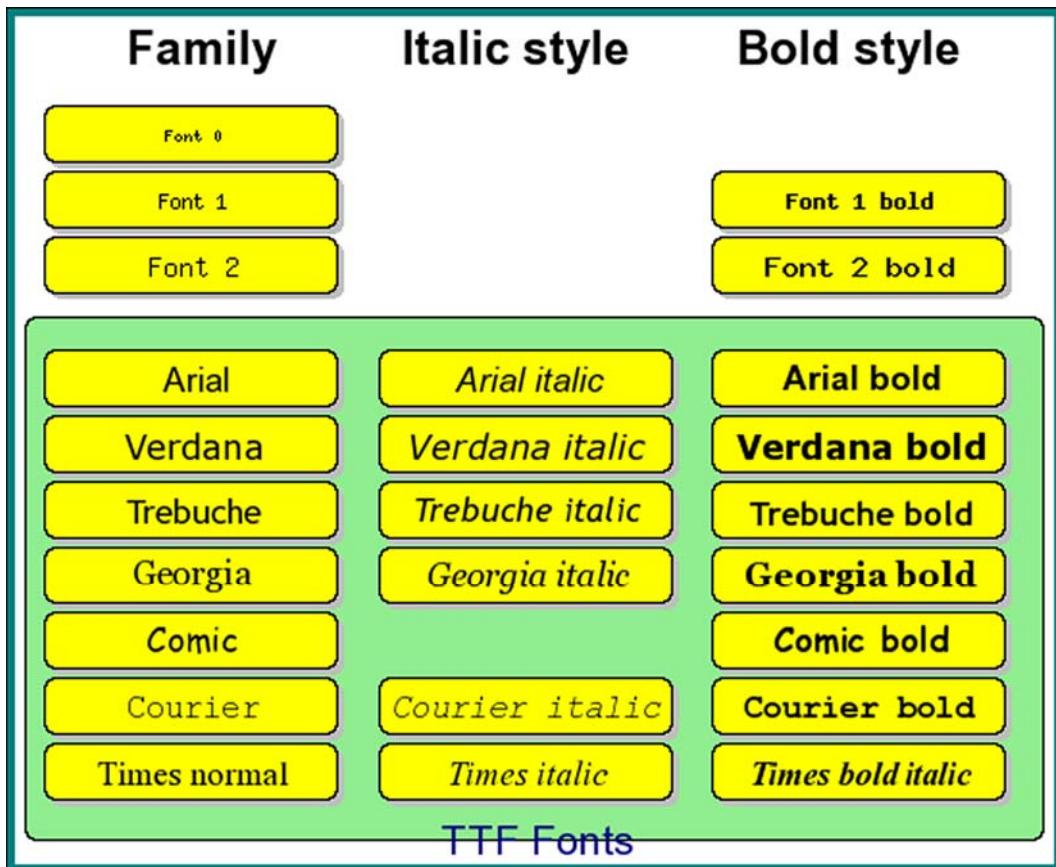


図 187:キャンバスを用いて多くのテキスト ボックスを描画する例

```

<?php
// $Id: listfontsex1.php,v 1.3 2002/10/25 22:44:15 aditus Exp $
include "../jpgraph.php";
include "../jpgraph_canvas.php";
include "../jpgraph_canvtools.php";

$g = new CanvasGraph(550,450,'auto');
$scale = new CanvasScale($g);
$scale->Set(0.27,0.53);
$g->SetMargin(5,6,5,6);
$g->SetColor('white');
$g->SetMarginColor("teal");
$g->InitFrame();

$t = new CanvasRectangleText();
$t->SetFillColor('lightgreen');
$t->SetFontColor('navy');
$t->SetFont(FF_ARIAL,FS_NORMAL,16);
$t-
>Set("￥n￥n￥n￥n￥n￥n￥n￥n￥n￥n￥n￥nTTF Fonts",0.5,19,26,3
2);
$t->Stroke($g->img,$scale);

$t = new CanvasRectangleText();
$t->SetFillColor('');
$t->SetFontColor('black');
$t->SetColor("");
$t->SetShadow("");
$t->SetFont(FF_ARIAL,FS_BOLD,18);

$t->Set('Family',1,1,8);
$t->Stroke($g->img,$scale);

$t->Set('Italic style',9,1,8);
$t->Stroke($g->img,$scale);

$t->Set('Bold style',17.5,1,8);
$t->Stroke($g->img,$scale);

$t->SetFillColor('yellow');
$t->SetFontColor('black');
$t->SetColor('black');
$t->SetShadow('gray');

$r=6;$c=1;$w=7.5;$h=3.5;

$fonts=array(
    array("Font 0",FF_FONT0,FS_NORMAL),
    array("",FF_FONT0,FS_ITALIC),
    array("",FF_FONT0,FS_BOLD),

    array("Font 1",FF_FONT1,FS_NORMAL),
    array("",FF_FONT1,FS_ITALIC),
    array("Font 1 bold",FF_FONT1,FS_BOLD),

    array("Font 2",FF_FONT2,FS_NORMAL),
    array("",FF_FONT2,FS_ITALIC),
    array("Font 2 bold",FF_FONT2,FS_BOLD),

    array("Arial",FF_ARIAL,FS_NORMAL),
    array("Arial italic",FF_ARIAL,FS_ITALIC),
    array("Arial bold",FF_ARIAL,FS_BOLD),

```

```

array("Verdana",FF_VERDANA,FS_NORMAL),
array("Verdana italic",FF_VERDANA,FS_ITALIC),
array("Verdana bold",FF_VERDANA,FS_BOLD),

array("Trebuchet",FF_TREBUCHET,FS_NORMAL),
array("Trebuchet italic",FF_TREBUCHET,FS_ITALIC),
array("Trebuchet bold",FF_TREBUCHET,FS_BOLD),

array("Georgia",FF_GEORGIA,FS_NORMAL),
array("Georgia italic",FF_GEORGIA,FS_ITALIC),
array("Georgia bold",FF_GEORGIA,FS_BOLD),

array("Comic",FF_COMIC,FS_NORMAL),
array("Comic italic",FF_COMIC,FS_ITALIC),
array("Comic bold",FF_COMIC,FS_BOLD),

array("Courier",FF_COURIER,FS_NORMAL),
array("Courier italic",FF_COURIER,FS_ITALIC),
array("Courier bold",FF_COURIER,FS_BOLD),

array("Times normal",FF_TIMES,FS_NORMAL),
array("Times italic",FF_TIMES,FS_ITALIC),
array("Times bold italic",FF_TIMES,FS_BOLDITALIC),
);

```

```

$n=count($fonts);

for( $i=0; $i < $n; ++$i ) {

    if( $i==9 ) $r += 3;

    if( $fonts[$i][0] ) {
        $t->SetTxt($fonts[$i][0]);
        $t->SetPos($c,$r,$w,$h);
        $t->SetFont($fonts[$i][1],$fonts[$i][2],13);
        $t->Stroke($g->img,$scale);
    }

    $c += $w+1;
    if( $c > 30-$w-2 ) {
        $c = 1;
        $r += 4;
    }
}

$g->Stroke();
?>

```

10.2 簡単なキャンバスの作成

キャンバス グラフを作成するためには、標準の “jpgraph.php” ファイルに加えて “jpgraph_canvas.php” ファイルをインクルードする必要があります。また、便利なサポート クラスにアクセスするために “jpgraph_canvtools.php” をインクルードすることもあるかもしれません。

キャンバスの作成で、“白い”紙に好きな形を自由に描画することができます。最初に、テキスト ボックスを描くだけの簡単な例を紹介しましょう。最初にこれから説明するコードを紹介します。

```
(File: canvasex01.php)
<?php
// $Id: canvasex01.php,v 1.3 2002/10/23 08:17:23
aditus Exp $
include "../jpgraph.php";
include "../jpgraph_canvas.php";

// 基本キャンバスの設定
$g = new CanvasGraph(400,300,'auto');
$g->SetMargin(5,11,6,11);
$g->SetShadow();
$g->SetMarginColor("teal");

// テキストが上書きされてしまうため、テキストを追加する前に
// プロットエリアとマージンを出力する必要がある
$g->InitFrame();

// 中央にテキスト ボックスを描く
$txt="This is na TEXT!!!";

```

```

$t = new Text($txt,200,100);
$t->SetFont(FF_ARIAL,FS_BOLD,40);

// テキストボックスが座標をどのように解釈するか
$t->Align('center','top');

// パラグラフをどのように位置揃えするか
$t->ParagraphAlign('center');

// テキストの周りに、白い塗りつぶし、黒い縁取り、灰色の影のボックスを追加する
$t->SetBox("white","black","gray");

// テキストを出力する
$t->Stroke($g->img);

// グラフを出力する
$g->Stroke();
?>

```

上の例は 400 x 300 のサイズのイメージを作成する部分から始まります。その後イメージの周りの余白を設定します。キャンバスでは、座標を入力する方法に余白の影響はありません。左上は (0,0) で、右下(余白と影を含む)が最大値です。この場合、X 座標は 0~399 Y 座標は 0~199 となります。

その後、余白とプロットエリアをグラフに出力する [InitFrame\(\)](#) メソッドを呼び出します。全てのグラフィカル オブジェクトはコマンドが呼び出される順番に出力されるので、最も手前に表示したいオブジェクトは最後に出力するように気を付けてください。JpGraph の通常の動作とは異なり、オブジェクトを追加すると全てその順番で反映されるので、正しい順番で出力されているか確認するようにしてください。

次に、[Text](#) オブジェクトを作成し、テキストを表示させたい場所をスクリーンの絶対位置で指定し、それを出力します。名前が非常に紛らわしい関数 [Text::Align\(\)](#) の詳細な説明が必要になるかと思います。このメソッドは、テキストの座標をどのように解釈させるか を決定します。例えば、テキストパラグラフの座標として (200, 10) を指定された場合、これが左上の角を意味するのか、左下を意味するのか、それともそれ以外で、ボックスの範囲を示す値なのか、といったことを判断させます。上記のコードでは、X 座標がボックスの中心を意味し、Y 座標がトップを意味するように解釈させました。したがって、パラグラフを囲むボックスのトップラインの中央が (200, 100) に設定されます。

[Text::ParagraphAlign\(\)](#) を使用して中央に寄せられているパラグラフの中にもラインを指定することができます。テキストの周りにボックスを表示させる場合は、ボックスの塗りつぶす色、ボーダーカラー、シャドウカラーを設定することができる[Text::SetBox\(\)](#) が役に立ちます。

ここで、キャンバス上でテキストを出力する準備をします。そのためには、使用したい基本イメージの描画クラスを指定しなければなりません。これについてはこれ以上の解説はせず、Graph クラスの `img` プロパティとして存在する `image` クラスについてのみ記述します。

これでようやくブラウザにキャンバスを出力する準備ができました。以下の画像でこのコード結果を全て見ることができます。



図 188:中央にテキストボックスを置いた簡単なキャンバスの例

```
<?php  
// $Id: canvases01.php,v 1.3 2002/10/23 08:17:23 aditus Exp $
```

```
include "../jpgraph.php";  
include "../jpgraph_canvas.php";
```

```
// Setup a basic canvas we can work
$g = new CanvasGraph(400, 300, 'auto');
$g->SetMargin(5, 11, 6, 11);
$g->SetShadow();
$g->SetMarginColor("teal");

// We need to stroke the plotarea and margin before
// we add the
// text since we otherwise would overwrite the text.
$g->InitFrame();

// Draw a text box in the middle
$txt="This is a TEXT!!!";
$t = new Text($txt, 200, 10);
$t->SetFont(FF_ARIAL, FS_BOLD, 40);

// How should the text box interpret the coordinates?
$g->TextAlign('center', 'top');

// How should the paragraph be aligned?
$t->ParagraphAlign('center');

// Add a box around the text, white fill, black border and gray shadow
$t->SetBox("white", "black", "gray");

// Stroke the text
$t->Stroke($g->img);

// Stroke the graph
$g->Stroke();

?>
```

10.3 キャンバスに線と長方形を追加する

キャンバスは円や線などの基本的な形を簡単に表示させることができます。覚えておかなければならることは、スクリーン座標は絶対値で使用することと、全ての描画オブジェクトは Graph クラスのプロパティである [Image](#) クラスで発見できるということです。例えば、座標 (0,0) と (100,100) の間に線を引くには、線オブジェクトを追加しなければなりません。

```
$graph->img->Line(0,0,100,100);
```

以下の例で、Image クラスからアクセス可能ないつかの描画オブジェクトを表示しています。

```
(File: canvasex02.php)
<?php
// $Id: canvasex02.php,v 1.1 2002/08/27 20:08:57
aditus Exp $
include "../jgraph.php";
include "../jgraph_canvas.php";

// 基本キャンバスの設定
$g = new CanvasGraph(400,200,'auto');
$g->SetMargin(5,11,6,11);
$g->SetShadow();
$g->SetMarginColor("teal");

// テキストが上書きされてしまうため、テキストを追加する前に
// プロットエリアとマージンを出力する必要がある
$g->InitFrame();

// 黒い線を追加する
$g->img->SetColor('black');
$g->img->Line(0,0,100,100);
```

```
// .. 円(x,y,直径)を追加する
$g->img->Circle(100,100,50);

// .. 塗りつぶした円(x,y,直径)の追加
$g->img->SetColor('red');
$g->img->FilledCircle(200,100,50);

// .. 長方形を追加する
$g->img->SetColor('black');
$g->img->Line(10,10,50,50);

// .. 塗りつぶした角の丸い長方形を追加する
$g->img->SetColor('green');
$g->img->FilledRoundedRectangle(300,30,350,80,10);
// 暗いボーダーを使用
$g->img->SetColor('darkgreen');
$g->img->RoundedRectangle(300,30,350,80,10);

// グラフを出力する
$g->Stroke();

?>
```

Graph クラスの img プロパティを通してこれらのルーチンにアクセスする方法に注目してください。また、座標は絶対値であるということも覚えておいてください。

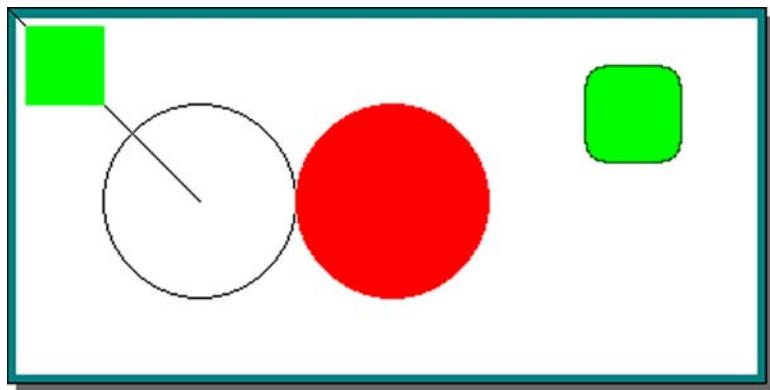


図 189:キャンバス オブジェクトの例

```
<?php
// $Id: canvasex02.php,v 1.1 2002/08/27 20:08:57 aditus Exp $
include "../jpgraph.php";
include "../jpgraph_canvas.php";

// Setup a basic canvas we can work
$g = new CanvasGraph(400,200,'auto');
$g->SetMargin(5,11,6,11);
$g->SetShadow();
$g->SetMarginColor("teal");

// We need to stroke the plotarea and margin before
we add the
// text since we otherwise would overwrite the text.
$g->InitFrame();

// Add a black line
$g->img->SetColor('black');
$g->img->Line(0,0,100,100);

// .. and a circle (x,y,diameter)
$g->img->Circle(100,100,50);

// .. and a filled circle (x,y,diameter)
$g->img->SetColor('red');
$g->img->FilledCircle(200,100,50);

// .. add a rectangle
$g->img->SetColor('green');
$g->img->FilledRectangle(10,10,50,50);

// .. add a filled rounded rectangle
$g->img->SetColor('green');
$g->img->FilledRoundedRectangle(300,30,350,80,10);
// .. with a darker border
$g->img->SetColor('darkgreen');
$g->img->RoundedRectangle(300,30,350,80,10);

// Stroke the graph
$g->Stroke();

?>
```

10.4 キャンバス スケールを使用する

これまでの関数は座標を絶対値で指定してきました。しかし、その方法だと画像の倍率を簡単に変更することはできず、低い解像度の場合はほとんど不可能です。特に基本的な形状を描画したい場合などに起こります。

これを防ぐためには、キャンバスにスケールを使用します。これは“ワークスペース”を定義させます。例えば、座標を X:0-10 と Y:0-10 の間に設定できるようになります。これにより、キャンバス内のオブジェクトの配置がより簡単になります。また、さらに 2 つの利点があります。

- キャンバスのサイズを大きくする場合、すべてのオブジェクトは比率が変更されることなく自動的に拡大されます。
- 別のスケールを使うだけで、(イメージではなく)描画オブジェクトを縮小・拡大することができます。例えば、(0:10, 0:10)のスケールでオリジナルのイメージを描画する場

合、(0:20, 0:20)のスケールに変更すると描画オブジェクトは半分のサイズに“縮小”されます。

このタイプのスケールを使用するには、“`jpgraph_canvtools.php`” をインクルードしていることを確認してください。スケーリングクラスに加えて、[Shape](#) クラスなどの他の便利なユーティリティ クラスもいくつか存在します。

スケールの使用はとても簡単です。まず、パラメーターとしてグラフを渡すスケール オブジェクトをインスタンス化し、使用したいスケールを指定します。

```
$scale = new CanvasScale($g);
setScale->Set(0,$xmax,0,$ymax);
```

そしてあなたのワールド座標とスクリーン座標を変換する `CanvasScale` クラスのから、変換メソッドを 1 つ(例えば、[CanvasScale::Translate\(\)](#))使うことができます。上記の例のコードを用いるだけで可能になります。

```
list($x1,$y1) = $this->scale->Translate($x1,$y1);
list($x2,$y2) = $this->scale->Translate($x2,$y2);
$g->img->Line($x1,$y1,$x2,$y2);
```

このパターンは、描画するあらゆるオブジェクトで繰り返されるので、個別のクラスでこれをまとめるのはよい考えです。また、これはまさにキャンバスツールファイルが [Shape](#) と呼ばれるユーティリティ クラスを持っている理由です。このクラスは、`Image` クラスの非常によく使われる関数群のラッパーです。Shape クラスに関する詳細はクラス リファレンスを参照してください。

```
$shape = new Shape($g,$scale);
```

これで `Shape` クラスの全てのメソッドを使う準備ができました。前の例を手本としてスケールを使用すると、以下のようなソースになります。

```
(File: canvasex03.php)
<?php
// $Id: canvasex03.php,v 1.1 2002/08/27 20:08:57
aditus Exp $
include "../jpgraph.php";
include "../jpgraph_canvas.php";
include "../jpgraph_canvtools.php";

// ワーク スペースを定義する
$xmax=20;
$ymax=20;

// キャンバスの設定
$g = new CanvasGraph(400,200,'auto');
$g->SetMargin(5,11,6,11);
$g->SetShadow();
$g->SetMarginColor("teal");

// テキストが上書きされてしまうため、テキストを追加する前に
// プロットエリアとマージンを出力する必要がある
$g->InitFrame();

// 新しいスケールを作成する
$scale = new CanvasScale($g);
$scale->Set(0,$xmax,0,$ymax);

// Shape クラスは、Image クラスのラッパー
```

```
$shape = new Shape($g,$scale);
$shape->SetColor('black');

// 黒いラインの追加
$shape->SetColor('green');
$shape->FilledRectangle(0,0,20,20);

// 円 (x,y,直径) の追加
$shape->Circle(5,14,2);

// .. 塗りつぶした円 (x,y,直径) の追加
$shape->SetColor('red');
$shape->FilledCircle(11,8,3);

// .. 長方形の追加
$shape->SetColor('green');
$shape->FilledRectangle(15,8,19,14);

// .. 塗りつぶした角の丸い長方形の追加
$shape->SetColor('green');
$shape->FilledRoundedRectangle(2,3,8,6);
// .. 暗いボーダーを使用
$shape->SetColor('darkgreen');
$shape->RoundedRectangle(2,3,8,6);

// グラフを出力する
$g->Stroke();
```

?>

上記のソースは以下の結果をもたらします。

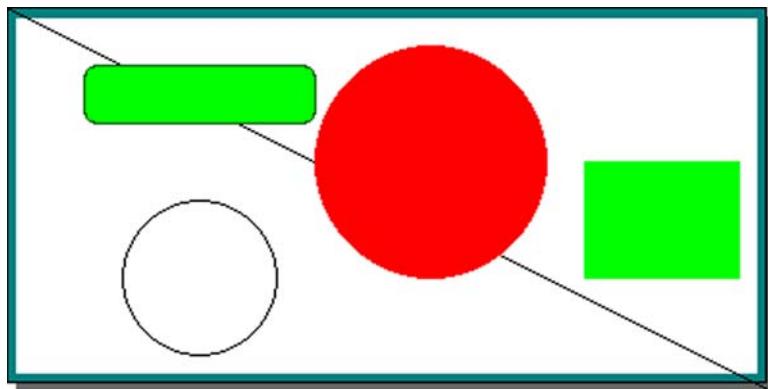


図 190:スケールを使用してキャンバスに形を描写する

```
<?php
// $Id: canvasex03.php,v 1.1 2002/08/27 20:08:57 aditus Exp $
include "../jpgraph.php";
include "../jpgraph_canvas.php";
include "../jpgraph_canvtools.php";

// Define work space
$xmax=20;
$ymax=20;

// Setup a basic canvas we can work
$g = new CanvasGraph(400,200,'auto');
$g->SetMargin(5,11,6,11);
$g->SetShadow();
$g->SetMarginColor("teal");

// We need to stroke the plotarea and margin before
we add the
// text since we otherwise would overwrite the text.
$g->InitFrame();

// Create a new scale
$scale = new CanvasScale($g);
$scale->Set(0,$xmax,0,$ymax);

// The shape class is wrapper around the Imgae class
// which translates
// the coordinates for us
$shape = new Shape($g,$scale);
```

```
$shape->SetColor('black');

// Add a black line
$shape->SetColor('black');
$shape->Line(0,0,20,20);

// .. and a circle (x,y,diameter)
$shape->Circle(5,14,2);

// .. and a filled circle (x,y,diameter)
$shape->SetColor('red');
$shape->FilledCircle(11.8,3);

// .. add a rectangle
$shape->SetColor('green');
$shape->FilledRectangle(15.8,19,14);

// .. add a filled rounded rectangle
$shape->SetColor('green');
$shape->FilledRoundedRectangle(2,3,8,6);
// .. with a darker border
$shape->SetColor('darkgreen');
$shape->RoundedRectangle(2,3,8,6);

// Stroke the graph
$g->Stroke();

?>
```

より小さなイメージを作成したい場合は、イメージ サイズを変更するだけで、コードを全く変更することなく再構成することができます。たとえば、画像のサイズを半分にすると以下のような画像が output されます。

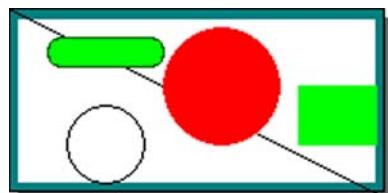


図 191:スケーリングがそのオブジェクトの相対的な位置を維持するので、
イメージのサイズを半分に 縮小することは簡単です

```
<?php
// $Id: canvasex04.php,v 1.1 2002/08/27 20:08:57 a
ditus Exp $

include "../jpgraph.php";
include "../jpgraph_canvas.php";
include "../jpgraph_canvtools.php";

// Define work space
$xmax=20;
$ymax=20;

// Setup a basic canvas we can work
$g = new CanvasGraph(200,100,'auto');
$g->SetMargin(5,11,6,11);
$g->SetShadow();
$g->SetMarginColor("teal");

// We need to stroke the plotarea and margin before
we add the
// text since we otherwise would overwrite the text.
$g->InitFrame();

// Create a new scale
$scale = new CanvasScale($g);
$scale->Set(0,$xmax,0,$ymax);

// The shape class is wrapper around the Imgae class
// which translates
// the coordinates for us
$shape = new Shape($g,$scale);

$shape->SetColor('black');

// Add a black line
$shape->SetColor('black');
$shape->Line(0,0,20,20);

// .. and a circle (x,y,diameter)
$shape->Circle(5,14,2);

// .. and a filled circle (x,y,diameter)
$shape->SetColor('red');
$shape->FilledCircle(11,8,3);

// .. add a rectangle
$shape->SetColor('green');
$shape->FilledRectangle(15,8,19,14);

// .. add a filled rounded rectangle
$shape->SetColor('green');
$shape->FilledRoundedRectangle(2,3,8,6);
// .. with a darker border
$shape->SetColor('darkgreen');
$shape->RoundedRectangle(2,3,8,6);

// Stroke the graph
$g->Stroke();

?>
```

イメージ サイズを変更せず、オブジェクトの形状を半分にするには単にスケールを 2 倍の大きさにするだけです。

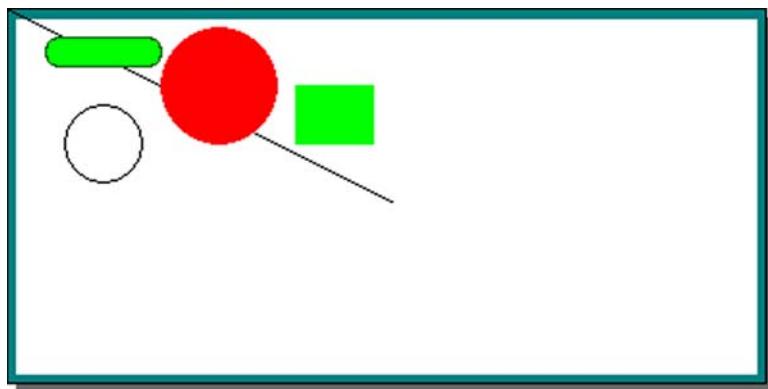


図 192:スケールを 2 倍の大きさにすることで、グラフィック オブジェクトを縮小します

```
<?php
// $Id: canvasex05.php,v 1.1 2002/08/27 20:08:57 a
ditus Exp $

include "../jpgraph.php";
include "../jpgraph_canvas.php";
include "../jpgraph_canvtools.php";

// Define work space
$xmax=40;
$ymax=40;

// Setup a basic canvas we can work
$g = new CanvasGraph(400,200,'auto');
$g->SetMargin(5,11,6,11);
$g->SetShadow();
$g->SetMarginColor("teal");

// We need to stroke the plotarea and margin before
we add the
// text since we otherwise would overwrite the text.
$g->InitFrame();

// Create a new scale
$scale = new CanvasScale($g);

$shape->SetColor('black');

// Add a black line
$shape->SetColor('black');
$shape->Line(0,0,20,20);

// .. and a circle (x,y,diameter)
$shape->Circle(5,14,2);

// .. and a filled circle (x,y,diameter)
$shape->SetColor('red');
$shape->FilledCircle(11,8,3);

// .. add a rectangle
$shape->SetColor('green');
$shape->FilledRectangle(15,8,19,14);

// .. add a filled rounded rectangle
$shape->SetColor('green');
$shape->FilledRoundedRectangle(2,3,8,6);
// .. with a darker border
$shape->SetColor('darkgreen');
$shape->RoundedRectangle(2,3,8,6);

// Stroke the graph
$g->Stroke();

?>
```

```
$scale->Set(0,$xmax,0,$ymax);

// The shape class is wrapper around the Image class
// which translates
// the coordinates for us
$shape = new Shape($g,$scale);
$shape->SetColor('black');

// Add a black line
$shape->SetColor('black');
$shape->Line(0,0,20,20);

// ... and a circle (x,y,diameter)
$shape->Circle(5,14,2);

// ... and a filled circle (x,y,diameter)
$shape->SetColor('red');

$shape->FilledCircle(11,8,3);

// ... add a rectangle
$shape->SetColor('green');
$shape->FilledRectangle(15,8,19,14);

// ... add a filled rounded rectangle
$shape->SetColor('green');
$shape->FilledRoundedRectangle(2,3,8,6);
// ... with a darker border
$shape->SetColor('darkgreen');
$shape->RoundedRectangle(2,3,8,6);

// Stroke the graph
$g->Stroke();

?>
```

Shape クラスは、1 つの例外を除けば Image クラスのラッパーであるということを述べました。では、例外とは何でしょう。例外には、"デコボコの長方形"を描画する追加メソッドがあります。デコボコの長方形は、4 つの角のうち 1 つが長方形の内側に移動した先の長方形です。[Shape::IndentedRectangle\(\)](#) などの関数を呼びだす、デコボコの長方形を作成することができます。

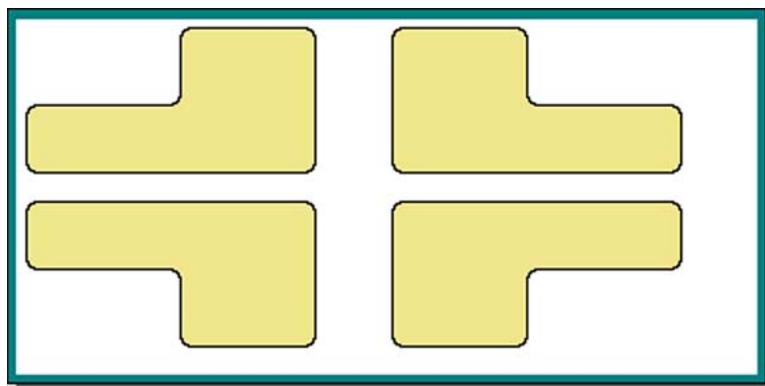


図 193:塗りつぶされたデコボコの長方形の例

```
<?php
// $Id: canvasex06.php,v 1.1 2002/08/27 20:08:57 a
ditus Exp $
include "../jpgraph.php";
include "../jpgraph_canvas.php";
include "../jpgraph_canvtools.php";

// Define work space
$xmax=40;
$ymax=40;

// Setup a basic canvas we can work
$g = new CanvasGraph(400,200,'auto');
$g->SetMargin(5,11,6,11);
$g->SetShadow();
$g->SetMarginColor("teal");

// We need to stroke the plotarea and margin before
we add the
// text since we otherwise would overwrite the text.
$g->InitFrame();

// Create a new scale
$scale = new CanvasScale($g);
```

```
$scale->Set(0,$xmax,0,$ymax);

// The shape class is wrapper around the Image class
// which translates
// the coordinates for us
$shape = new Shape($g,$scale);
$shape->SetColor('black');

$shape-
>IndentedRectangle(1,2,15,15,8,8,CORNER_TOPLEFT,'khaki');

$shape-
>IndentedRectangle(1,20,15,15,8,8,CORNER_BOTTOMLEFT,
'khaki');

$shape-
>IndentedRectangle(20,2,15,15,8,8,CORNER_TOPRIGHT,'khaki');

$shape-
>IndentedRectangle(20,20,15,15,8,8,CORNER_BOTTOMRIGHT,
'khaki');
```

```
// Stroke the graph  
$g->Stroke();
```

?>

最後の注意として、角が丸みを帯びた長方形にテキストを追加できる[CanvasRectangleText](#) クラスについて述べます。利用できるフォントが全て描画されたサンプル画像はこのクラスを使用しています。詳しくは説明しませんが、クラス リファレンスと 'listfontsex1.php' を参照してください。

10.5 サンプル アプリケーション:DB schema を描画する

最後の例として、キャンバス タイプのグラフを DDDA 構造の DB Schema 生成でどう使用するかについて簡単に述べます。

“utils/misc/imgdbschema.php” ライブドリの PHP ファイルは、より簡単にテーブル スキーマを描画するユーティリティ クラスを含む配置にあります。2 つの基本クラス、ImgDBTable クラスと ImgDBSchema クラスがあります。最初のクラスは、1 つのテーブルを描くイメージの描き方を理解します。2 つ目のクラスは、完全な DB スキーマを描画する DB から、全ての関連情報を自動的に抽出するのに信頼性があります。

先に進む前に、この例に似ているものを説明します。

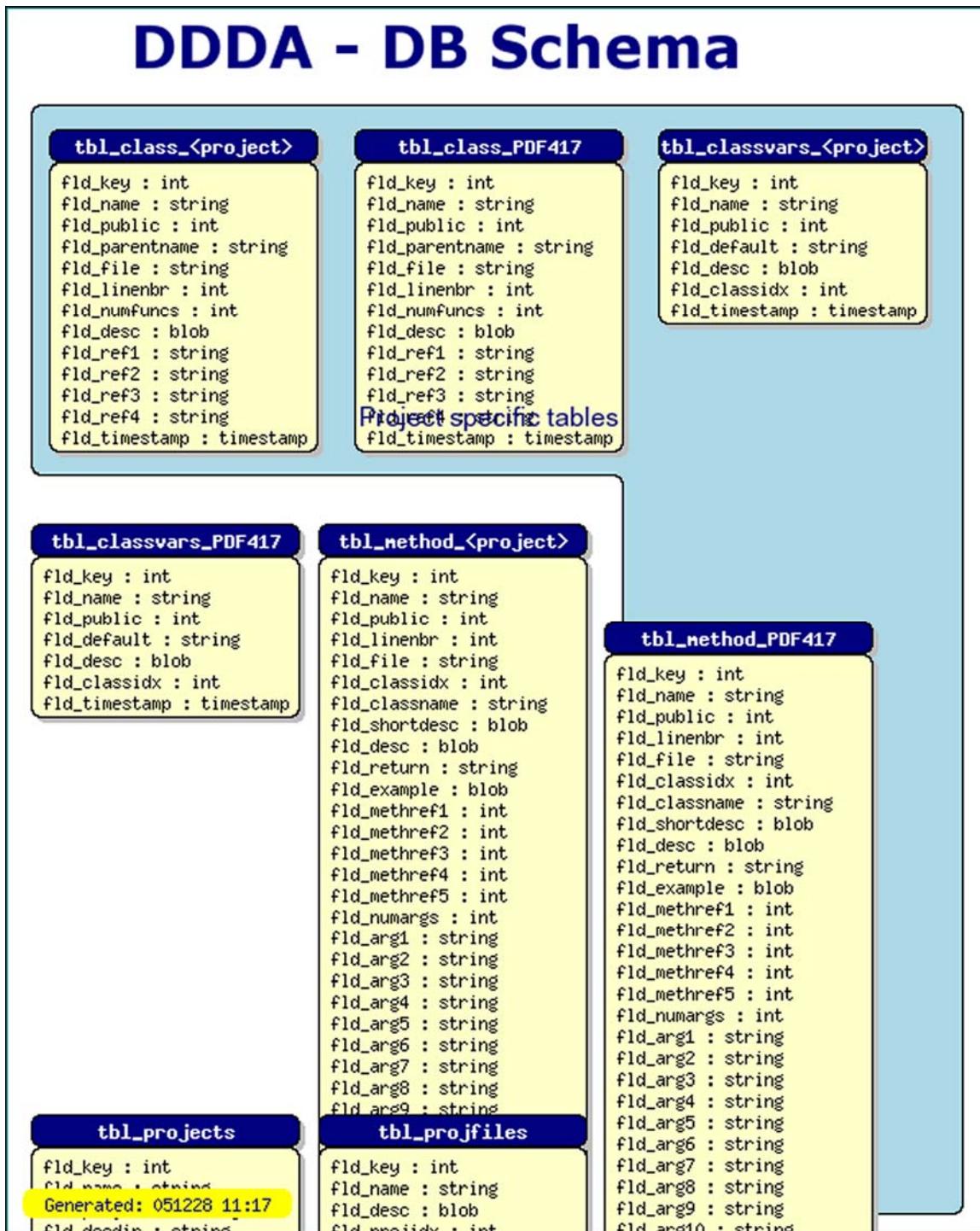


図 194: DB スキーマを半自動的に生成する imgdbschema.php ライブライと
一緒にキャンバス グラフスタイルを使用する例

```

<?php
/*=====
=====
// File:      DBSCHEMAEX1.PHP
// Description: Draw a DB schema of the DDDA architecture
// Created:    2002-08-25
// Author:    Johan Persson (johnp@aditus.nu)
// Ver:       $Id: dbschemaex1.php,v 1.1 2002/08/27 20:08:57 aditus Exp $

```

```

// License: This code is released under QPL
//           Copyright (C) 2001,2002 Johan Persson
// Note: The actual drawing of the tables are
//       semi-automatically
//       but you can easily adjust the individual tables position
//       with the 'tblposadj' array.
// 

```

```

//=====
/*
include "../jpgraph.php";
include "../jpgraph_canvas.php";
include "../jpgraph_cantools.php";
include "../utils/misc/imgdbschema.inc";
include "../utils/jpdocgen/jpdb.php";

// Global callback to format the table header names
function FormatTblName($aName) {
    // We want to replace any specific references to the
    // 'JpGraph' project with the generic '<project>'
    return str_replace('JpGraph','<project>', $aName);
}

// Global callback to format each field name in the table
function FormatFldName($aName,$aTable) {
    return $aName;
}

class Driver {

    var $ig, $img, $iscale, $ishape;
    var $iymax,$ixmax;
    var $iwidth,$iheight;

    function Driver() {

        // Define Image size and coordinate grid space to work within
        $this->iwidth = 600;
        $this->iheight= 750;
        $this->iymax = 50;
        $this->ixmax = 55;

        // Setup a basic canvas
        $this->ig = new CanvasGraph($this->iwidth,$this->iheight,'auto');
        $this->img = $this->ig->img;

        // Define the scale to be used
        $this->iscale = new CanvasScale($this->ig);
        $this->iscale->Set(0,$this->ixmax,0,$this->iymax);
        $this->ishape = new Shape($this->ig,$this->iscale);

        // A small frame around the canvas
        $this->ig->SetMargin(2,3,2,3);
        $this->ig->SetMarginColor("teal");
        $this->ig->InitFrame();
    }

    function Run() {

        $leftm=1.5;      // Left margin (for table schemes)
        $topm=5;         // Top margin (for table schemes)
        $tblwidth=15;    // Individual table width
        $tlo=1;          // Offset for top line

        // Add the background color for the project specific tables
        $this->ishape->IndentedRectangle($leftm,$topm-1,3*$tblwidth+$tlo+6,45,
                                          $tlo+2*$tblwidth+2,30,CORNER_B
                                          OTTOMLEFT,
                                          'lightblue');

        // Stroke the tables (series of x,y offsets, If == 1 then use the
        // automatic positioning
        $tblposadj=array($tlo,0,$tblwidth+$tlo+2,0,2*$tblwidth+$tlo+4,
                         0,-1,16,-1,16);
        $dbschema = new ImgDBSchema('jpgraph_doc','FormatTblName','FormatFldName');
        $dbschema->SetMargin($leftm,$topm);
        $dbschema->SetTableWidth($tblwidth);
        $dbschema->Stroke($this->img,$this->iscale,$tblposadj);

        $tt = new CanvasRectangleText();
        $tt->SetFillColor('');
        $tt->SetColor('');
        $tt->SetFontColor('navy');

        // Add explanation
        $tt->SetFont(FF_ARIAL,FS_NORMAL,12);
        $tt-
        >Set('Project specific tables',$tblwidth+$leftm+3,16,15);
        $tt->Stroke($this->img,$this->iscale);

        // Add title
        $tt->SetColor('');
        $tt->SetFont(FF_VERDANA,FS_BOLD,26);
        $tt->Set('DDDA - DB Schema',9,0,5,30);
        $tt->Stroke($this->img,$this->iscale);

        // Add a version and date
        $tt->SetFillColor('yellow');
        $tt->SetFont(FF_FONT1,FS_NORMAL,10);
        $tt-
        >Set("Generated: ".date("ymd Hi",time()),1,$this->iymax*0.96,15);
        $tt->Stroke($this->img,$this->iscale);

        $this->ig->Stroke();
    }
}

$driver = new Driver();
$driver->Run();
?>

```

先に進む前に、DDDA 構造の 'jpdb.php' ファイルで利用可能なアブストラクション レイヤーの後に、DB アブストラクション レイヤー モデルを通して、アクセス可能な DB を仮定する ImgDBSchema に注意してください。このアブストラクション レイヤーは、下部で MySQL データベースを仮定します。特別なアブストラクション レイヤーの依存状態は、一般的なキャンバス ツール ファイルにこれらのクラスがインクルードされていない理由を意味しています。

第二に注意すべきことは、ライブラリーは完璧な自動的なレイアウトエンジンを含んでいるのではなく、むしろ他に何も指定されていないのなら、ただ長方形のマス目にテーブルを入れるとも簡単な自動的なシステムを含んでいます。完全なグラフ レイアウト エンジンは、この状況での書き込みを非常に簡単にします。これはとても難しい最適化の問題もあり、今までのところ、この問題に挑んだどんなプログラムも手動の調停無しで満足できるレイアウトを達成できていません。

以上のグラフをもたらすためのコードでの重要なラインです。

```
$tblposadj=array($tlo,0,$tblwidth+$tlo+2,0,2*$tblwidth+$tlo+4,0,-1,16,-1,16);
$dbschema = new ImgDBSchema("jpgraph_doc","FormatTblName","FormatFldName");
$dbschema->SetMargin($leftm,$topm);
$dbschema->SetTableWidth($tblwidth);
$dbschema->Stroke($this->img,$this->iscale,$tblposadj);
```

ファイルの中でのコードの休止は、ただキャンバスをセットアップするためで、いくつかのテーブルに分類するためのデコボコの長方形を加えて、このイメージがもたらした日時のあるフッターを作ります。

第一に、ラインが'jpgraph_doc'というデータベースのためのイメージを描くように、それに要求する新しい ImgDBSchema レイアウトエンジンを例示します。下記の 2 つの引数は、ヘッダーやテーブルのそれぞれのフィールドのテキストをフォーマットするため 2 つのコールバック ファンクションを指定します。

第二に、ラインがテーブルの描画がスタートされるべき左上余白を指示します。

第三に、ラインが一つのテーブルの幅を指示します。最後に、ラインはエンジンをはじめ、キャンバスのデータベースでの全てのテーブルに描写します。この最後の引数は、いくつかの前の意味で要求します。これは、各々のテーブルが配置されるべきである左上端からの(x,y)の補正です。もし値があるのならば、-1はデフォルト値が使用されるべきであることを示します。もしこの配列が指定されたら、テーブルはラインによって簡単に配置されるでしょう。

この DB schema の例を描くための 完全なソースコードは、以下に示されます。

```
(File: dbschemaex1.php)
<?php
/*=====
=====
// ファイル: DBSCHEMAEX1.PHP
// 説明: DDA 構造の DB スキーマを描画する
// 作成日: 2002-08-25
// 著者: Johan Persson (johanp@aditus.nu)
// バージョン: $Id: dbschemaex1.php,v 1.1 2002/08/27
20:08:57 aditus Exp $
//
// ライセンス: // このコードは QPL のもとリリースされました
// Copyright (C) 2001,2002 Johan Persson
// 注意: 実際のテーブルの描画は半自動的ですが、簡単に
// ック
function FormatTblName($aName) {
// 特定のリファレンスを一般の '<project>' を含む 'JpGraph'
プロジェクト
// に置き換える場合
return str_replace('JpGraph','<project>', $aName);
}

// テーブルの各フィールド名をフォーマットするグローバルコ
ールバック
function FormatFldName($aName,$aTable) {
```

```
それぞれのテーブルの
// 位置を調整できます
//
// =====
// =====
/*
include "../jpgraph.php";
include "../jpgraph_canvas.php";
include "../jpgraph_canvtools.php";
include "../utils/misc/imgdbschema.inc";
include "../utils/jpdocgen/jpdb.php";

// テーブルのヘッダ名をフォーマットするグローバルコールバ
return $aName;
}

class Driver {

var $ig, $img, $iscale, $ishape;
var $iymax,$ixmax;
var $iwidth,$iheight;

function Driver() {
```

```
// イメージサイズと座標のグリッドスペースを定義します
$this->iwidth = 600;
$this->iheight= 750;
$this->iymax = 50;
$this->ixmax = 55;

// 基本キャンバスを設定する
$this->ig = new CanvasGraph($this->iwidth,$this->iheight,'auto');
$this->img = $this->ig->img;

// 使用するスケールを定義する
$this->iscale = new CanvasScale($this->ig);
$this->iscale->Set(0,$this->ixmax,0,$this->iymax);
$this->ishape = new Shape($this->ig,$this->iscale);

// A small frame around the canvas
$this->ig->SetMargin(2,3,2,3);
$this->ig->SetMarginColor("teal");
$this->ig->InitFrame();

}

function Run() {

$leftm=1.5; // 左の余白(テーブル キーマ用)
$topm=5; // トップの余白(テーブル キーマ用)
$tblwidth=15; // 各々のテーブル幅
$tlo=1; // トップ ラインをオフセットする

// プロジェクト指定のテーブルに背景カラーを追加する
$this->ishape->IndentedRectangle($leftm,$topm-1,3*$tblwidth+$tlo+6,45,
$tlo+2*$tblwidth+2,30,CORNER_BOTTOMLEFT,
'lightblue');

// テーブルをストロークする(x、y、オフセット、=-1 の場合、
自動配置を使用する)
$tblposadj=array($tlo,0,$tblwidth+$tlo+2,0,2*$tblwidth+$tlo+4,
```

```
0,-1,16,-1,16);
$dbschema = new
ImgDBSchema('jpgraph_doc','FormatTblName','FormatFldNa
me');
$dbschema->SetMargin($leftm,$topm);
$dbschema->SetTableWidth($tblwidth);
$dbschema->Stroke($this->img,$this->iscale,$tblposadj);

$tt = new CanvasRectangleText();
$tt->SetFillColor('');
$tt->SetColor('');
$tt->SetFontColor('navy');

// 説明の追加
$tt->SetFont(FF_ARIAL,FS_NORMAL,12);
$tt->Set('Project specific
tables',$tblwidth+$leftm+3,16,15);
$tt->Stroke($this->img,$this->iscale);

// タイトルの追加
$tt->SetColor('');
$tt->SetFont(FF_VARDANA,FS_BOLD,26);
$tt->Set('DDDA - DB Schema',9,0,5,30);
$tt->Stroke($this->img,$this->iscale);

// バージョンと日付の追加
$tt->SetFillColor('yellow');
$tt->SetFont(FF_FONT1,FS_NORMAL,10);
$tt->Set("Generated: ".date("Y.m.d H:i",time()),1,$this-
>iymax*0.96,15);
$tt->Stroke($this->img,$this->iscale);

$this->ig->Stroke();
}
}

$driver = new Driver();
$driver->Run();

?>
```

11 アンチスパム

JpGraph には簡単にアンチスパム画像を生成するモジュールがあります。アンチスパム画像とは、OCR プログラムでも判別することが難しい数字と文字から構成される画像のことです。これは、とても効果的にロボット(スパム ボット)によるサインアップを防ぐので、掲示板、オンラインメールシステム、またはメールリストへのサインアップなどを安全に保つために使用されます。

この JpGraph のモジュールは `jpgraph_antispam.php` を読み込んで作成します。

以下の例はアンチスパム画像がどのように表示されるかの例です。



図 195:アンチスパムのシンプルな例

```
<?php
// Antispam example using a random string
require_once "../jpgraph_antispam.php";

// Create new anti-spam challenge creator
// Note: Neither '0' (digit) or 'O' (letter) can be used to avoid confusion
$spam = new AntiSpam();

// Create a random 5 char challenge and return the st
    ring generated
$chars = $spam->Rand(5);

// Stroke random challenge
if( $spam->Stroke() === false ) {
    die('Illegal or no data to plot');
}

?>
```

アンチスパムは機能が非常に少なく、それによりこのユーティリティの正確さと使いやすさが保たれています。制限としては、生成されたイメージは常に JPEG イメージになるということです。たとえば、PNG などに変更することはできません。

11.1 アンチスパム画像を作成する

アンチスパム画像を生成するほうほうは 2 通りあります。

1. 使用される文字列を送る
2. ランダムに文字列を生成する これを選択する場合、モジュールに作成された文字列を保存し、ユーザが入力した内容と一致するかどうかを確認してください。

アンチスパムを生成するスクリプトを書くには、4 つのステップがあります。

1. ライブラリファイル `jpgraph_antispam.php` をインクルードする。すべての機能がこのライブラリファイルに含まれているので、“`jpgraph.php`”をインクルードする必要はありません。

```
require_once "jpgraph_antispam.php";
```

2. `AntiSpam` クラスの新しいインスタンスを作成します

```
$spam = new AntiSpam();
```

3. アンチスパムで使用される文字列を指定します。 文字列を指定しない場合、自動的に文字列が生成されます。

```
// 生成する文字列の長さを決定する  
$chars = $spam->Rand(5);
```

使用する文字列を指定する場合、この文字列は AntiSpam() インスタンスの作成時、または Set() 関数の呼び出しで指定します。

```
$spam->Set("aui8k");
```

文字の '0' と数字の '0' (ゼロ) は非常によく似ていて間違えやすいので、これらの文字は無視されるということに注意してください。

4. AntiSpam クラスのインスタンスで Stroke() 関数を呼ぶことで画像を出力します。

```
if( $spam->Stroke() === false ) {  
die ("Illegal or no data to plot");  
}
```

他のグラフ タイプと同様に、Stroke() にファイル名を引数として送ることで生成されたイメージをファイルに書き出すことができます。

12 エンタープライズ版スタンダードで作成できるグラフ

12.1 オドメータ(走行距離積算計)

このモジュールはエンタープライズ版のライブラリでのみご利用可能です。

オドメータ(走行距離計)グラフは車の旧式スピードメーターのようなもので、特定のパラメーターをざっと見るために使われます。この種の比喩を使うとたいていの人には簡単に理解できます。オドメータに色の表示を追加することで、特定の値が表示できるレンジに収まるかそうでないかがすぐわかります。

次に表示するのはライブラリで作成した典型的なオドメータの一部です。

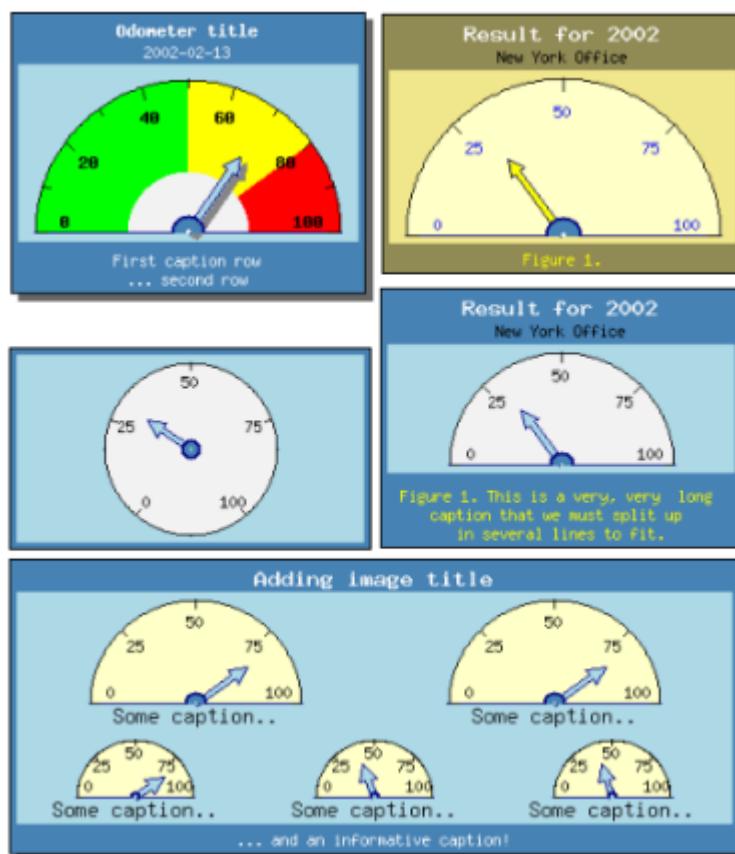


図 12.1.1 様々なオドメータ

12.1.1 オドメータ・モジュールの機能

このパッケージで作成したオドメータは幅広い機能を持ち、あらゆるレイアウトを様々な方法でカスタマイズすることができます。また、簡単に使えるように、ほとんどのパラメタには初期設定値があり、単純なオドメータなら数行のコードで作成できます。

他にもオドメータは次のような機能をサポートしています。

- 半円と円のオドメータ
- 目盛りやサイズを自由にカスタマイズ
- 様々な指針のサイズや形
- スケールのカラー表示
- 同じグラフ・エリアでの複数のオドメータ表示
- ライブラリの標準フォントとフォント・カラーの完全サポート
- 同一グラフ・エリアに複数プロットを表示するための自動レイアウト・エンジン。ピクセル座標指定なしで適切な位置設定をサポート。

12.1.2 基本オドメータ・グラフの作成とフォーマット

オドメータ・グラフを使うには「jpgraph_odo.php」という拡張モジュールをスクリプト内に記述する必要があります。これは通常の「jpgraph.php」コード・モジュールの追加としてです。

それ以外は、オドメータの作成でも通常の手順どおりグラフ・エリアを作成し、そのキャンバス上に1つまたは複数のプロットを追加します。

1. 基本キャンバス・グラフを OdoGraph クラスのインスタンスとして作成します。
2. 1つまたは複数のオドメータ・プロットのインスタンスを、OdoPlot クラスのインスタンスとして作成し、グラフ・キャンバスに追加します。
3. オドメータ・グラフを OdoGraph::Stroke()メソッドでブラウザに返します。

次のスクリプトは原則に従った記述です。

```
<?php
// 300x200 ピクセルの新規キャンバスを作成。
$graph = new OdoGraph(300,200);

// パラメータを必要に応じて設定。
$graph->title->Set(...);
$graph->caption->Set(...)

// 1つまたは複数のオドメータを作成。
$odo1 = new Odometer();
$odo2 = new Odometer();

...

// オドメータのパラメーター、色、その他を設定。
$odo1->needle->Set(21);
$odo2->needle->Set(47);

...
...
```

// オドメータ・プロットを縦に並べる。
\$li = new LayoutVert(array(\$odo1, \$odo2, ...));

// オドメータを設定済みのレイアウトでキャンバスに追加する。
\$graph->Add(\$li);

// オドメータが1つだけの場合は次のように記述することも可能。
// \$graph->Add(\$odo1);

// 出来上がった画像をブラウザに返す。
\$graph->Stroke();

?>

12.1.2.1 基本オドメータを作成する

初期設定値のみを使用した一番簡単なオドメータのスクリプトは下記のようになります。

例 12.1.2.1 基本オドメータ

```
<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_odo.php');

// 新規オドメータ・グラフ・エリアを作成。(幅 250, 高さ 200
ピクセル)
$graph = new OdoGraph(250,140);

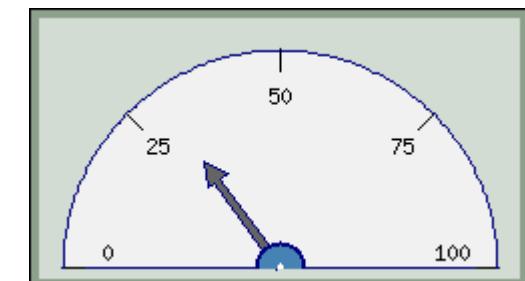
// オドメータを作成しグラフ・エリアに追加。
// 初期設定によりスケール(目盛り)は 0 から 100。
$odo = new Odometer();
```

```
// オドメータの指針が指す値を設定。
$odo->needle->Set(30);
```

```
// オドメータをグラフ・エリアに追加。
$graph->Add($odo);
```

```
// ... 最後に、完成した画像をブラウザに返す。
$graph->Stroke();
```

```
?>
```



- 図 12.1.2.1 基本オドメータ

オドメータのサイズは与えられた画像に最適に適合するよう自動的に設定されます。オドメータのサイズはどのようなオドメータ・キャプション、グラフ・タイトル、グラフ・キャプションにでも適応するように自動設定されるので、手動で調整する必要な滅多にありません。

しかし、手動で半径の絶対ピクセル数か画像サイズに対する比率でオドメータのサイズを手動設定することも可能です。

オドメータの半径に値を設定するには次のメソッドを使います。

- OdoPlot::SetSize(\$aRadius)

OdoGraph クラスは他のすべての基本グラフの機能を受け継いでいるので、コードを追加することで、例えば上の図のオドメータ・グラフにタイトルを付けることも可能です。

```
$graph->title->Set('A suitable graph title');
```

この後の章ではオドメータ・グラフのサイズや形、見栄えを変更するためのオプションを見ていきます。

Tips: オドメータは他のすべての基本グラフの機能を有しているので、画像キャッシュなどもサポートしています。

-

12.1.3 サンプル集

オドメータは円か半円で作成します。特に指定されていない場合は初期設定により半円で作成されます。

円か半円のどちらのタイプにするかは次の2つの定数でコントロールします。

- ODO_FULL 圓形のオドメータを作成
- ODO_HALF 半円のオドメータを作成(初期設定値)

圓形のオドメータ・プロット作成は次のように記述します。

```
$odo = new Odometer(ODO_FULL);
```

半円のオドメータ・プロット作成は次のように記述します。

```
$odo = new Odometer(ODO_HALF);
```

例 12.1.3_1 圓形オドメータの作成

```
<?php // content="text/plain; charset=utf-8"
require_once ("jpgraph/jpgraph.php");
require_once ("jpgraph/jpgraph_odo.php");

// 新規オドメータ・グラフ・エリアを作成(幅 250、高さ 200 ピクセル)
$graph = new OdoGraph(250,140);

// オドメータ・グラフ・エリアに入れるためのオドメータを作成する。
// 初期設定により目盛り(Scale)は 0 から 100。
$odo = new Odometer(ODO_FULL);

// 指針がオドメータ上で指す値を設定。
$odo->needle->Set(30);

// オドメータをグラフ・エリアに追加。
$graph->Add($odo);

// ... 最後に、完成した画像をブラウザに返す。
$graph->Stroke();

?>
```

次のように表示されます。

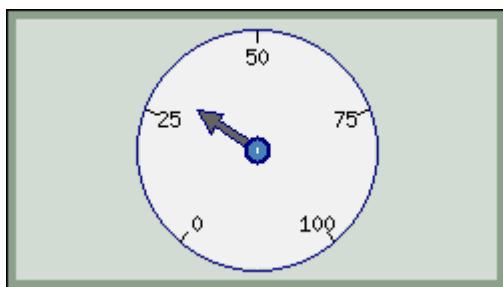


図 12.1.3_1 圓形基本オドメータ

タイトルやサブタイトル、キャプションを上のサンプルに追加するには下記のように記述します。

例 12.1.3_2 タイトルとキャプションの追加

```
<?php // content="text/plain; charset=utf-8"
require_once ("jpgraph/jpgraph.php");
require_once ("jpgraph/jpgraph_odo.php");

// 新規オドメータ・グラフ・エリアを作成(幅 250、高さ
200 ピクセル)
$graph = new OdoGraph(250,180);

// タイトル、サブタイトル、キャプションを設定
$graph->title->Set("Result for 2002");
$graph->title->SetColor("white");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,14);
$graph->subtitle->Set("New York Office");
$graph->subtitle->SetColor("white");
$graph->caption->Set("Figure 1. Branch results.");
$graph->caption->SetColor("white");

// オドメータ・グラフ・エリアに入れるためのオドメータ
を作成する。
// 初期設定により目盛り(Scale)は 0 から 100。
$odo = new Odometer();

// 指針がオドメータ上で指す値を設定。
$odo->needle->Set(30);
```

```
// オドメータをグラフ・エリアに追加。
$graph->Add($odo);
```

```
// ... 最後に、完成した画像をブラウザに返す。
$graph->Stroke();
?>
```



図 12.1.3_2 タイトルとキャプションの追加

例 12.1.3_3 複数行のキャプションを追加する

```
<?php // content="text/plain; charset=utf-8"
require_once ("jpgraph/jpgraph.php");
require_once ("jpgraph/jpgraph_odo.php");

// 新規オドメータ・グラフ・エリアを作成(幅 250、高さ
200 ピクセル)
$graph = new OdoGraph(250,200);

// タイトル、サブタイトル、キャプションを設定
$graph->title->Set("Result for 2002");
$graph->title->SetColor("white");
$graph->title->SetFont(FF_ARIAL,FS_BOLD,14);
$graph->subtitle->Set("New York Office");
$graph->subtitle->SetColor("white");
$graph->caption-
>Set("Figure 1.This is a very, very long text with
multiples
lines
that are added as a caption.");
$graph->caption->SetColor("white");

// オドメータ・グラフ・エリアに入れるためのオドメータ
を作成する。
// 初期設定により目盛り(Scale)は 0 から 100。
$odo = new Odometer();

// 指針がオドメータ上で指す値を設定。
$odo->needle->Set(30);
```

```
// オドメータをグラフ・エリアに追加。
$graph->Add($odo);
```

```
// ... 最後に、完成した画像をブラウザに返す。
$graph->Stroke();
?>
```

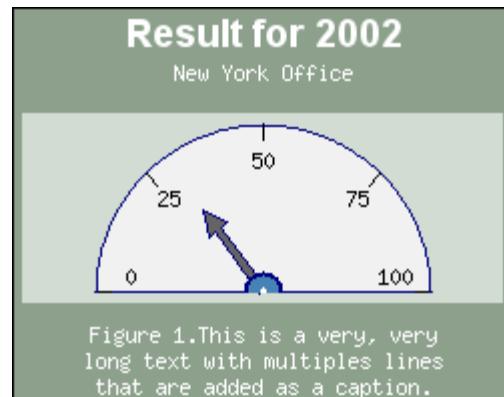


図 12.1.3_3 複数行のキャプションを追加

12.2 風配図

このモジュールは JpGraph エンタープライズ版でのみ使用できます。

風配図プロットは通常、コンパス方位に関する値を表示するために使われます。例えば一定時間に複数方位で測定された風力を表すことに使われます。次の図は一つの風配図を一つの風配図プロットにした例です。

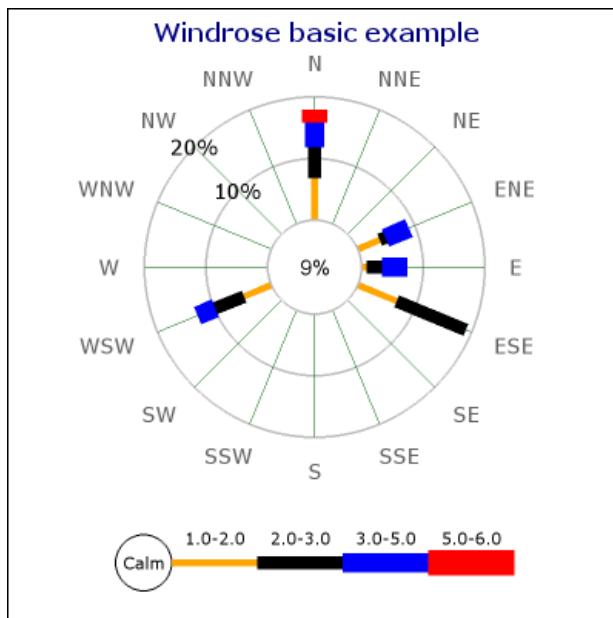


図 12.2.1 風配図基本プロット

風配図プロットではレンジ(つまり下の表の各「バケット番号」)に対応するカンマで区切られたレンジ)にデータを格納します。バケット内のデータポイントの数値は一定の方位にあるバケットの始点と終点を表現します。入力したデータは特定の方位の中心からの長さに対するパーセント比率を特定します。表 21.1 では初期設定では8つのバケットがそのレンジ表示されています。凡例の初期設定レンジは風力表示に使われる標準値です。凡例のバケット数の数字は仮の数値で、実際の表示はバケット番号を特定するためだけに入力した値であり絶対値ではありません。「JpGraph の設定を変更する」をご参照ください。

バケット番号	バケット境界値
0	[0,1]
1	[1,2]
2	[2,3]
3	[3,5]
4	[5,6]
5	[6,10]
6	[10,13.5]
7	[13.5,99]

表 12.2.1 風配図バケットの初期設定値

注意:風配図の初期設定フォントは TTF です。つまり、この章のサンプル図と全く同じように表示するには TTF フォントをインストールする必要があります。TTF が使えない場合はビルトイン・フォントのどれかに変更する必要があります。

風配図プロットは画像の境界部や斜線を完全にスムーズにするアンチエイリアス機能をサポートしています。上の図の画像はこのアンチエイリアス機能を使用しています。

警告: アンチエイリアスを使うには TTF フォントを有効にして使用する必要があります。

12.2.1 風配図モジュールの機能

風配図拡張モジュールは、通常のライブラリ・グラフでできること、つまりフレーム、シャドウ、カラー、タイトル、キャッシュなどのすべての機能をサポートしています。他にも下記の特徴を備えています。

- グラフをスムーズにみせるアンチエイリアス機能の強化
- マニュアルか自動かを選択できる目盛り
- 表示するコンパス方位の数を4、8、16の中から選ぶためのコンパス・タイプをカスタマイズ可能
- 自由な角度のグラフとコンパス方位によるグラフ、2つの基本グラフ・タイプから選択可能
- コンパス方位と角度の両方でデータを特定
- プロットのどこでもフォントのカスタマイズ自由
- 目盛りとラベルの自動ポジショニングによりデータ位置の衝突回避。マニュアルでラベルのポジショニングするときは停止することもできます。
- ラベルはフォーマット機能が豊富で背景色や角を丸くしたボックス、任意のフォントも使用可能
- 同じ画像内で複数の風配図を使用可能
- 個々の風配図のサイズや位置設定を自由にカスタマイズ
- 風配図内の個々のプロットのサイズは絶対ピクセル数でもグラフサイズの比率でも指定可能
- マニュアルと自動、どちらでも色指定可能
- マニュアルと自動、どちらでもメモリ設定指定可能
- クライアント指定の配置調整を行う場合の方角ラベルのインテリ・ポジショニング
- 風配図に任意のテキスト・ブロック追加可能

12.2.2 風配図の種類

風配図プロットには以下の2つの基本タイプがあります。

- **コンパス・タイプ**

コンパス・タイプの風配図プロットでは、コンパス方位の軸数は 16 本、8 本、4本の中から選べます。入力データの方位はプロット内のコンパス方位軸の 1 本ごとに限定されます。方位は方位軸の番号で特定します。

方位軸のラベルの初期設定はコンパス方位の短縮形です。例えば「E」「N」「W」「S」。コンパス方位の初期設定名の変更の仕方はこれによりご想像がつくと思いますが、東、つまり3時の方位から時計と反対回りに番号がついています。

• フリー・タイプ

フリー・タイプの風配図プロットは事前に設定された方位軸はありません。入力データのディレクション定数で任意の角度が設定できます。（例えば「NW」などのように指定できます。方位ゼロは東、つまり3時の方角です。

プロット・タイプは次の方法で指定します。

`$aDirection` は4つの初期設定のコンパス・タイプ定数から選びます。

- `WINDROSE_TYPE4` は4本の方位軸のコンパス・タイプ
- `WINDROSE_TYPE8` は8本の方位軸のコンパス・タイプ
- `WINDROSE_TYPE16` は16本の方位軸のコンパス・タイプ
- `WINDROSE_TYPEFREE`, フリー・タイプ。データ方位は任意の角度となり、事前に設定されたラベルや軸はありません。

下図 12.2.2_1 から 12.2.2_4 は基本となるタイプ別風配図プロットを表しています。

図 12.2.2_1 WINDROSE_TYPE 4

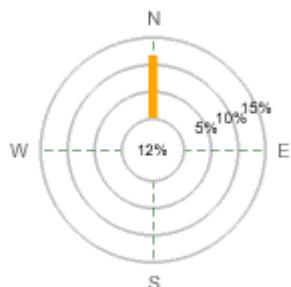


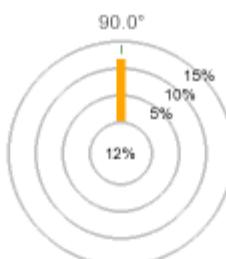
図 12.2.2_2 WINDROSE_TYPE 8



図 12.2.2_3 WINDROSE_TYPE 16



図 12.2.2_4 WINDROSE_TYPEFREE



次のコードでは 16 軸のコンパス方位がある風配図を指定しています。

```
<?php  
$wp = new WindrosePlot($data);  
$wp->SetType(WINDROSE_TYPE16);  
?>
```

注意: 風配図プロット・タイプが指定されていない場合、初期設定は 16 軸のコンパス方位となります。

入力データでタイプを指定：

入力データは配列の形をとります。配列関数への入力がそれぞれ一方位に対するデータを表す配列となります。入力データはある方位とバケットに属する 全データに対するパーセント数と解釈します。つまり、すべての方位に対するすべての入力値の合計は 100 以下であることになります。データが損失した場合は 100 未満になります。

最初のバケット、「バケット 0(ゼロ)」は 0 値、即ちデータのその部分は計測無し(または無風状態)を指定します。無風状態では方位も存在しないため、バケット 0 はすべて内側の円にまとめて表示されます。

凡例に表示される初期設定の各バケット境界値は次の通りです。

0-1, 1-2, 2-3, 3-5, 5-6, 6-10, 10-13.5, 13-99
0-1, 1-2, 2-3, 3-5, 5-6, 6-10, 10-13, 13-99

最初の要素「0-1」は 0 から 1 示度数のパーセント値を指定しており、風配図プロットの中心に置かれます。次の要素「1-2」は 1 から 2 パーセントの測定値、「2-3」は 2 から 3 パーセントの測定値…となります。

方位はその要素を特定するカギとなります。方位の特定方法は次のいずれかで行います。

- 「N」「SN」など、表示されたコンパス方位を指定する String を使う。
- 配列の使用例：

```
$data = array ( array('E' => <east_bucket0>,<east_bucket1> ... <east_bucketN> ),  
               array('ESE' => <eastsoutheast_bucket0>, ... <eastsoutheast_bucketN> ),  
               ...  
               array('NE' => <northeast_bucket0>, ... <northeast_bucketN> ),  
               array('ENE' => <eastnortheast_bucket0>, ... <eastnortheast_bucketN> ));
```
- 東の方方位から時計回りに軸を表示する通常のプロットに対し、0 から 15 の番号（整数）を使う。表示されている軸だけを数えることに注意。ゼロは東の方方位を指定。

```
$data = array ( array(0 => <east_bucket0>,<east_bucket1> ... <east_bucketN> ),  
               array(1 => <eastsoutheast_bucket0>, ... <eastsoutheast_bucketN> ),  
               ...
```

```
array(2 => <northeast_bucket0>, ... <northeast_bucketN> ),
array(3 => <eastnortheast_bucket0>, ... <eastnortheast_bucketN> ));
```

- 角度が「34.5」のような数値の場合、分数ではなく String 変数として認識できる小数点を指定する必要があります。小数であればライブラリは有効な角度として自動的に認識し、グラフ上で想定通りに扱います。（これは浮動小数点型は連想配列の中で使用できないためです。）

```
$data = array ( array('12.5' => <bucket0>,<bucket1> ... <bucketN> ),
array('22.1' => <bucket0>, ... <bucketN> ),
```

同じデータセットの中ではスタイルを混合することも可能です。

警告: ライブラリは、すべてのバケット値の合計が 100% より大きいかどうか以外はデータの検証を行いません。指定されたデータが理にかなったものであるかはブラウザでの確認が必要です。

通常のコンパス方位風配図プロットを使用中に表示できない角度を指定するとそれを知らせるエラーメッセージが出ます。

この機能がどう働くかを下記の例で説明します。

例 12.2.2_1 通常の風配図コンパス方位での入力データの例

(16 本の軸がすべて表示されていると仮定。)

```
$data = array( 'N' => array(2,5,6) );
```

このデータ配列では「北」という1つの方位のみ指定しており、バケット値の範囲が 0 では 2%、0~1 では 5%、1~2 では 6% を表します。

(16 本の軸がすべて表示されていると仮定。)

```
$data = array( 1 => array(2,5,6), 3 => array(6,3), 'NW' => array(3,2,2,2) )
```

このデータ配列では3つの方位を指定しています。ご覧のように、方位を示す軸番号や「NW」のような方位名を混合させることができます。しかし同時に使用するときはひとつの方法に決めておくほうが明確でいいでしょう。

例 12.2.2_2 フリー風配図プロット用の入力データ例

フリータイプの特別な点は 16 本のコンパス方位に限らず任意の方位を指定できることです。

```
$data = array( 10 => array(2,5,6), 24 => array(6,3), 137 => array(3,2,2,2) )
```

このデータ配列では、10 度、24 度、137 度の 3 つの方位のバケットの数値を指定しています。

```
$data = array( '21. 5' => array(2,5,6), 'N' => array(6,3), 137 => array(3,2,2,2) )
```

このデータ配列は、21.5 度、「北」(90 度)、137 度の 3 方位の測定値を指定しています。分数になりそうな数値を持つ角度であれば「21.5」のように String 値として扱われる角度を指定します。それによってライブラリが自動的に適切に取り扱います。

12.2.3 基本風配図の作成とフォーマット

風配図グラフの作成には最初にライブラリ拡張モジュールである「jpgraph_windrose.php」を含めることが必要です。

最初に含めない場合の風配図の作成は、グラフを作成してから1つあるいは複数の風配図プロットをキャンバスに加えていくという、ライブラリの通常のステップをいくつか踏むことになります。

1. 基本キャンバスグラフを WindroseGraph クラスのインスタンスとして作成する。
2. 一つまたは複数の風配図プロットのインスタンスを WindrosePlot クラスとして作成し、目盛りと外観の設定を行い、それらをグラフのキャンバスに追加します。
3. 「WindroseGraph::Stroke()」を呼び出して、グラフをクライアントに返します。いつものようにこれは、グラフをクライアント(つまりブラウザ側)に送り返すか、ファイルにグラフを書き込むために使います。ファイル名は WindroseGraph::Stroke() の第 1 引数として指定します。

図 12.2.3_1 の例はすべてのパラメタに初期設定値のみ使用した場合の風配図です。

例 12.2.3_1 16 方位の基本風配図

```
(File:windrose_ex0.php)
<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');

// データは軸番号か方位ラベルの両方で指定可能。
$data = array(
    0 => array(5.5,5,8),
    1 => array(3,4,1,4),
    'WSW' => array(1,5,5,3),
    'N' => array(2,3,8,1,1),
    15 => array(2,3,5));

// まず新規の風配図をタイトルを付けて作成。
$graph = new WindroseGraph(400,400);
$graph->title->Set('A basic Windrose graph');

// 風配図プロットを作成。
$wp = new WindrosePlot($data);

// グラフを追加してブラウザに返す。
$graph->Add($wp);
$graph->Stroke();
?>
```

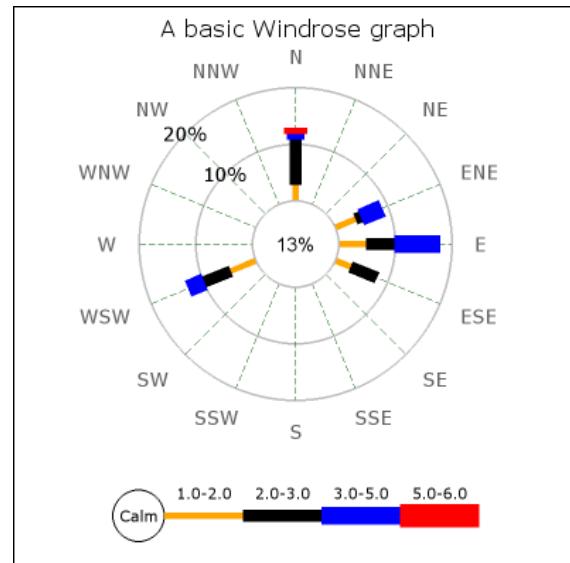


図 12.2.3_1 方位の基本風配図[\[ソース\]](#)

他のタイプのグラフ同様、風配図でも1つ以上の風配図プロットを自由に追加したり位置設定したりできます。位置は絶対数でも、グラフ全体の幅・高さに対する比率でも指定できます。

各風配図プロットをグラフに追加するには次のようにレイアウト・クラスのオブジェクトを呼び出すメソッドを使用します。

- WindroseGraph::Add(\$aObject)

すべてのプロットが真ん中で衝突するのを避けるため、前のセクションで説明した位置設定とサイズ調整する方法を使用してください。

次のスクリプトは2つのプロットがどのようにひとつのグラフに表示されるかを記述しています。

例 12.2.3_1 16 方位の基本風配図

```
(File:windrose_2plots_ex1.php)
<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');

// データは軸番号でも方位ラベルでも指定可能。
$data = array(
    1 => array(10,10,13,7),
    2 => array(2,8,10),
    4 => array(1,12,22),
);

$data2 = array(
    4 => array(12,8,2,3),
    2 => array(5,4,4,5,2),
);

// 新規の風配図を作成
$graph = new WindroseGraph(660,400);
$graph->SetShadow();

$graph->title->Set('Two windrose plots in one graph');
$graph->stroke->SetBox();

$graph->title->SetFont(FF_ARIAL,FS_BOLD,14);
$graph->subtitle->Set('Using Box() for each plot');

$wp = new WindrosePlot($data);
$wp->SetType(WINDROSE_TYPE8);
$wp->SetSize(0.42);
$wp->SetPos(0.25,0.55);
$wp->SetBox();

$wp2 = new WindrosePlot($data2);
$wp2->SetType(WINDROSE_TYPE16);
$wp2->SetSize(0.42);
$wp2->SetPos(0.74,0.55);
$wp2->SetBox();
$wp2->SetRangeColors(array('green','yellow','red','brown'));

$graph->Add($wp);
$graph->Add($wp2);

$graph->Stroke();
?>
```

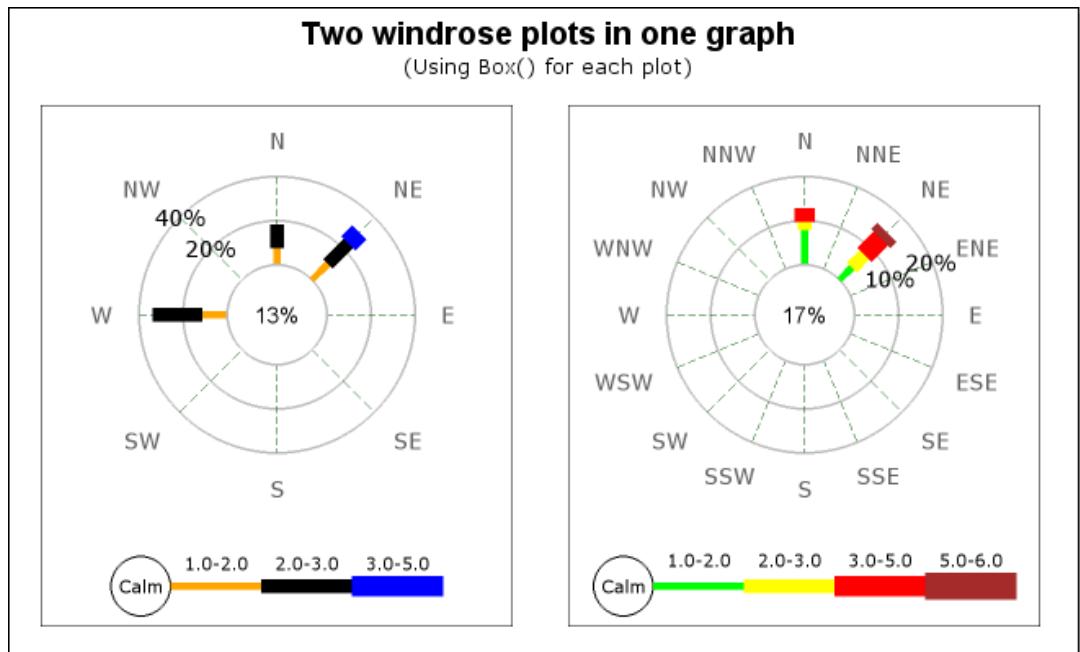


図 12.2.3_1 同じグラフに2つの風配図プロットを追加

12.2.3.1 風配図の目盛りを指定する

目盛りは各プロットのプロパティでインスタンス変数「WindrosePlot::scale」によって操作できます。目盛りは同心円として風配図プロット上で放射状に広がっています。円と円の間の幅は手動で自動でもデータの値によりライブラリから操作できます。

(説明文を継続するため、このセクションでは例を全部表示せずに「サンプル集」に回します。)

円の目盛りを手動で設定するには次の方法を使います。

- `WindrosePlotScale::SetStepSize($aMax,$aStepSize=null)`
 \$aMax, The maximum scale value
 \$aStepSize, The step between each scale circle

ライブラリの初期設定により5と2の倍数が綺麗な目盛間隔になるよう表示されます。

目盛りを調整するための他の手法は次の通りです。

- `WindrosePlotScale::SetFont($aFontFamily,$aFontStyle=FS_NORMAL,$aFontSize=10)`
 中心の0値ラベル以外の全ての目盛りのフォントを指定します。
- `WindrosePlotScale::SetZFont($aFontFamily,$aFontStyle=FS_NORMAL,$aFontSize=10)`
 中心の0値ラベルのフォントを指定します。
- `WindrosePlotScale::SetFontColor($aColor)`
 中心の0値ラベル以外の全ての目盛りのフォントカラーを指定します。
- `WindrosePlotScale::SetZFontColor($aColor)`
 中心の0値ラベルのフォントカラーを指定します。
- `WindrosePlotScale::SetLabelFillColor($aBkgColor,$aBorderColor=false)`
 全てのラベルの背景色を指定します。
- `WindrosePlotScale::SetLabelAlign($aAlign)`
 ラベルの文字の配置を指定します。
- `WindrosePlotScale::SetLabelFormat($aFmt)`
 ラベルのフォーマットを指定します。フォーマットString変数「printf()」を使います。
- `WindrosePlotScale::Hide($aFlg)`
- 目盛りラベルを非表示にします。
 これらの方法を使用した例は次のセクションに記載しています。

フォントとフォントカラーを指定する

フォントは目盛り上のラベル全てに対して指定することも、プロットの中心のラベルを別に指定することもできます。それをどのように行うかは次のコードの抜粋をご覧いただけます。(ここでは「windplot」という名前の風配図プロットが既に作成してあると仮定します。)

```
// 目盛りラベルのフォントとフォントカラー
$windplot->scale->SetFont(FF_VERDANA,FS_NORMAL,10);
$windplot->scale->SetFontColor('navy');

// 中心の0値ラベルの目盛りラベルのフォントとフォントカラー
$windplot->scale->SetZFont(FF_VERDANA,FS_NORMAL,10);
$windplot->scale->SetZFontColor('navy');
```

目盛りを任意で指定する

ライブラリの初期設定により、目盛りの範囲と間隔はデータの最大値、プロットの大きさ、目盛りラベルのフォントの大きさを考慮して自動的に決まります。

しかし、Set(\$a.MaxValue,\$aStepSize)を呼び出すことで強制的に手動で行うことができます。ここでは目盛りの最大値と間隔を指定します。目盛り間隔は任意です。目盛り間隔が指定されていない場合は自動で指定されます。

```
// 目盛り最大値と間隔を指定。
$windplot->scale->Set(40,8);

...
// 目盛りの最大値のみを指定。
$windplot->scale->Set(40);
```

この特徴については、この章の「サンプル集」の図 12.2.4_7にも紹介しています。

注意: 自動的に生成される目盛りは目盛りのラインに適した5か2の倍数の刻み幅を使っています。

注意: 目盛りラベルのフォントが変更されるときに目盛りの間隔が変化する場合は驚かれるかもしれません、そういう設計になっています。各刻み幅に適用するラベルには限られたスペースしかありません。大きいフォントがマス目のスペース内に収まるように、それぞれの目盛りに当たる円の大きさが調整されます。

目盛りラベルの角度の指定

目盛りラベルは風配図プロットの中心から放射状に伸びる方位軸線に沿って置かれます。ラベルの角度は任意でも自動でも設定できます。

ラベルの自動位置設定を選択した場合、目盛りラベルは 16 本のコンパス方位のどれかに沿って置かれます。ライブラリの初期設定ではデータが一番少ない方位を選ぶようになっています。

角度の手動設定の場合は、SetAngle(\$aAngle)を呼び出し、度数を引数に入れます。

次のように記述します。

```
// ラベルを 45 度の角度で表示する  
$windplot->scale->SetAngle(45);
```

初期設定によりライブラリが角度の自動設定を行いますが、これを明示するには次のように「'auto'」という文字列を使います。

```
// ライブラリの適切な角度を設定させる  
$windplot->scale->SetAngle('auto');
```

目盛りラベルの数字フォーマットを指定する

他のライブラリの目盛り同様、「printf()」フォーマット文字列を指定してラベルの表示を調整することができます。初期設定では次のようにになっています。

```
$windplot->scale->SetLabelFormat("%d%");
```

12.2.3.2 方位ラベルを指定する

ここでは「方位ラベル」とは風配図プロットのまわりのコンパス方位の名前のこと指します。初期設定では 16 の方位はそれぞれ英語の方位名の省略形になっています。これもご使用になる地域・言語に合わせて変更が可能です。

方位軸ラベルのフォントとフォントカラーの指定

軸ラベルのフォントは次の関数に通常の引数を入れて指定します。

- WindrosePlot::SetFont()
- WindrosePlot::SetFontColor()

下は引数を入れた記述例です。

```
$windplot->SetFont(FF_TIMES,FS_BOLD,12);  
$windplot->SetFontColor('darkgreen');
```

風配図プロットの表示は図 12.2.4_4 の風配図プロットのようになります。

データの方針の任意文字列の設定

フリー・タイプの風配図プロットを使う場合、ラベルの初期設定はデータの角度と同じ方位になります。これで次のようにカスタマイズできます。

- WindrosePlot::SetLabels(\$aLabels)

この方法による入力データでは、連想配列を使い、キーに方位、カッコ内には表示するテキストを記述します。

通常のコンパス方位の風配図プロットでは方位ラベルは変更できないことにご注意ください。変更は「WINDROSE_TYPEFREE」の風配図プロットでのみ可能です。

次の例では、角度が 50 度の位置でテキスト「Point #7315」のラベルのデータポイントを指定する例です。

ヒント:ラベルには改行を行う「\n」を含むこともできます。

```
$data = array( 50 => array(12,12,2) );
$labels = array( 50 => 'Point #7315' );
...
$windplot = new WindrosePlot();
$windplot->SetLabels($labels);
```

このサンプルは以下の図のようになります。(使っているバケットの角度は異なります。)

配置と余白を調整する

プロットのまわりのコンパス・ラベルの位置を調整するには以下の 2 通りの方法があります。

- ラベルとプロットの外周の間の余白を調整する。
- ラベルのアンカー・ポイント(基準点)を調整する。

まず、プロットのまわりに見えない円があると想像してみてください。余白のマージンはこの想像上の外円を、いちばん外の目盛りとなる円からどれくらい離れた場所に置くかによって指定します。

プロットの外周の円の中のどこにラベルを置くかについては、テキストのどの点を外円に合わせるかを選択することによって、いくつかの方法があります。一番簡単な方法はテキストの真ん中、(つまり縦横両方向でみた中央)を選択することです。すべてのラベルの大きさがほぼ同じ場合はこの方法でうまくいきます。最大ラベルと最小ラベルのサイズに大きな違いがある場合は余白を大きくとって、テキストがプロットと衝突するのを避ける必要があるかもしれません。

もっといいのはテキスト上でプロットに最短となるポイントを決め、そのポイントを円の上で設定する方法です。この最短ポイントはどのコンパス方位に ラベルを置くかによって異なります。例えば「東」に置くラベルならテキストの左端の中央が最短ポイントとなり、「西」での最短点はテキスト右端の中央になります。

ライブラリは下記の方法によって両方の位置設定をサポートします。

- WindrosePlot::SetLabelPosition(\$aPosition)

\$aPosition に有効な値は LBLPOSITION_CENTER と LBLPOSITION_EDGE です。

余白と位置の両方を指定するには次のように記述します。

```
$windplot->SetLabelMargin(30);  
$windplot->SetLabelPosition(LBLPOSITION_EDGE);
```

下の図の中の赤い円は、上で説明した想像上のラベルを置くための円です。プロット上の一一番外の円からラベルのアンカー・ポイントまでの幅が余白マージンです。

図 12.2.3.2_1

LBLPOSITION_CENTER による位置設定

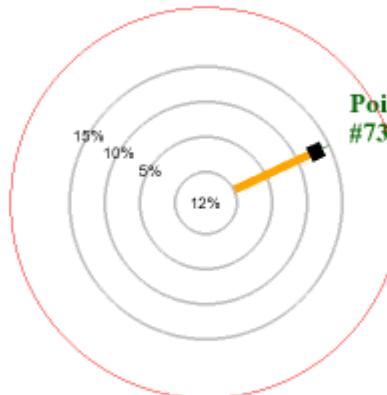
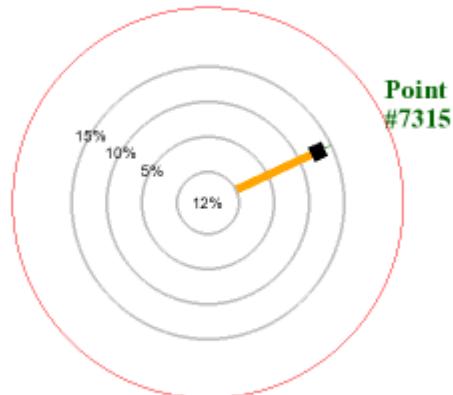


図 12.2.3.2_1

LBLPOSITION_EDGE による位置設定。



この方法で描かれたラベルの例はサンプル集の[図 12.2.4_6](#)をご覧ください。

12.2.4 サンプル集

例 12.2.4_1

1. ほぼすべて初期設定値を使用した初步的な風配図プロットの表示
2. タイトルのフォーマットを追加
3. 凡例テキストの追加
4. 軸番号と方位名の両方でコンパス方位を指定する方法の説明

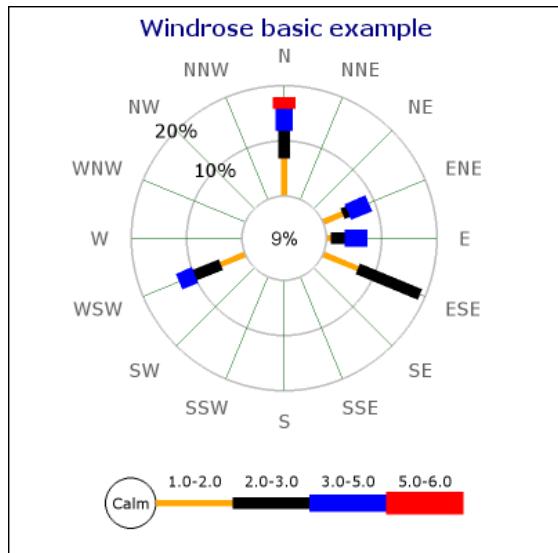


図 12.2.4_1 [ソース]

```
<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');
```

```
$data = array(
    0 => array(1,1,2.5,4),
    1 => array(3,4,1,4),
    'ws' => array(1,5,5,3),
    'N' => array(2,7,5,4,2),
    15 => array(2,7,12));

// まず新規の風配図グラフ・エリアを作成する。
$graph = new WindroseGraph(400,400);

// タイトルを設定。
$graph->title->Set('Windrose basic example');
$graph->title->SetFont(FF_VARDANA,FS_BOLD,12);
$graph->title->SetColor('navy');

// 風配図プロットを作成。
$wp = new WindrosePlot($data);
$wp->SetRadialGridStyle('solid');
$graph->Add($wp);

// グラフをブラウザに返す。
$graph->Stroke();
?>
```

例 12.2.4_2

1. 風配図バケットの幅をカスタマイズする方法
2. 8 本のコンパス方位軸のみを表示する方法
3. プロットを特定のピクセル・サイズで設定する方法

4. 中心円のサイズを調整する方法
5. 目盛りラベルとフォントやフォントカラーの調整方法
6. コンパス方位のフォントの調整方法

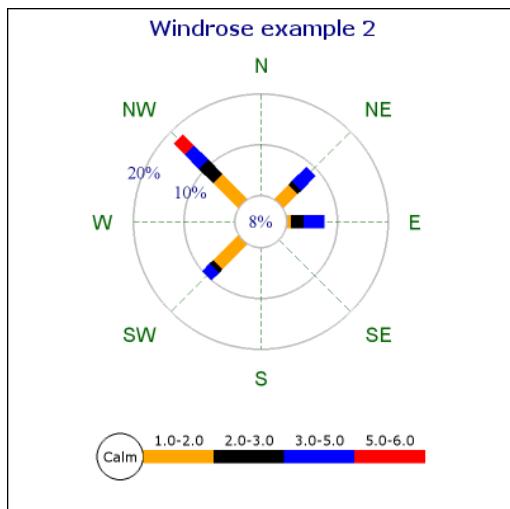


図 12.2.4_2 [ソース]

```
<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');

// データは軸番号と方位ラベルのどちらでも指定できます。
$data = array(
    0 => array(1, 1, 2.5, 4),
    1 => array(3, 4, 1, 4),
    3 => array(2, 7, 4, 4, 3),
    5 => array(2, 7, 1, 2));

// 風配図グラフ・エリアを作成。
$graph = new WindroseGraph(400, 400);
```

```
// タイトルを設定。
$graph->title->Set('Windrose example 2');
$graph->title->SetFont(FF_VERDANA, FS_BOLD, 12);
$graph->title->SetColor('navy');

// 風配図プロットを作成。
$wp = new WindrosePlot($data);

// 8 本のコンパス方位軸に設定。
$wp->SetType(WINDROSE_TYPE8);

// レンジの異なるバケットに一定の太さを設定する
$weights = array_fill(0, 8, 10);
$wp->SetRangeWeights($weights);

// 目盛りラベルのフォントとフォント・カラーを設定する。
$wp->scale->SetFont(FF_TIMES, FS_NORMAL, 11);

$wp->scale->SetFontColor('navy');

// プロットの直径を 200 ピクセルに設定する。
$wp->SetSize(200);

// 中心円をプロット・サイズの 20%に設定する。
$wp->SetZCircleSize(0.2);

// コンパス方位のフォントとフォント・カラーを設定する。
$wp->SetFont(FF_ARIAL, FS_NORMAL, 12);
$wp->SetColor('darkgreen');

// 作成されたグラフをクライアント(ブラウザ)に返す。
$graph->Add($wp);
$graph->Stroke();
?>
```

例 12.2.4.3

1. 同じグラフ・エリア内に複数のプロットを追加する方法
2. サブタイトルを追加し、グラフ・エリアの縁に影を付ける方法
3. プロットのサイズと位置を変更する方法
4. 凡例テキストの追加方法

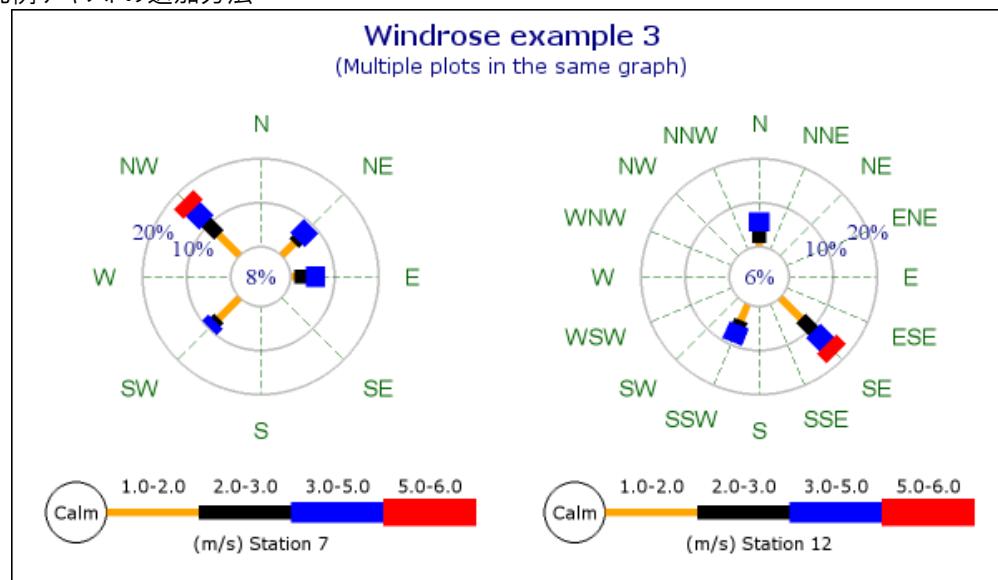


図 12.2.4_3 [ソース]

```
<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');

// データは方位軸番号でも方位ラベルでもどちらでも指定できます。
$data[0] = array(
    0 => array(1,1,2,5,4),
    1 => array(3,4,1,4),
    3 => array(2,7,4,4,3),
    5 => array(2,7,1,2));

$data[1] = array(
    "n" => array(1,1,2,5,4),
    "ssw" => array(3,4,1,4),
    "se" => array(2,7,4,4,3));

// 複数のプロット作成を容易にするため、
// 各プロットの位置やサイズを配列に格納する。
// レイアウト・データのためのフォーマットは、
// (風配図タイプ,Y 軸位置,X 軸位置,プロットサイズ,
// 中心円のサイズ)となります。
setLayout = array(
    array(WINDROSE_TYPE8,0.25,0.55,0.4,0.25),
```

```

array(WINDROSE_TYPE16,0.75,0.55,0.4,0.25));

$legendtxt = array('(m/s) Station 7',(m/s) Station 12');

//影付き(dropshadow)の風配図グラフ・エリアを作成。
$graph = new WindroseGraph(600,350);
$graph->SetShadow('darkgray');

//タイトル・サブタイトルを設定。
$graph->title->Set('Windrose example 3');
$graph->title->SetFont(FF_VERDANA,FS_BOLD,12);
$graph->title->SetColor('navy');
$graph->subtitle-
>Set('Multiple plots in the same graph');
$graph->subtitle->SetFont(FF_VERDANA,FS_NORMAL,9);
$graph->subtitle->SetColor('navy');

// 2つの風配図プロットを作成。
for( $i=0; $i < count($data); ++$i ) {
    $wp[$i] = new WindrosePlot($data[$i]);

    // 8本方位軸タイプに設定。
    $wp[$i]->SetType($layout[$i][0]);

    // 目盛りラベルのフォントとフォント・カラー
    // を設定。
}

// wp[$i]->SetType($layout[$i][0]);
// wp[$i]->SetFont(FF_VERDANA,FS_NORMAL,9);
// wp[$i]->SetColor('navy');

// wp[$i]->SetFont(FF_TIMES,FS_NORMAL,10);
// wp[$i]->SetFontColor('navy');

// プロットの位置を X 軸 Y 軸で設定。
$wp[$i]->SetPos($layout[$i][1],$layout[$i][2]);

// プロットの直径をグラフ・エリアの幅のピク
セル数の 40%に設定。
$wp[$i]->SetSize($layout[$i][3]);

// 中心円の直径をプロットの直径の 25%に設定。
$wp[$i]->SetZCircleSize($layout[$i][4]);

// コンパス方位のフォントとフォント・カラ
ーを設定。
$wp[$i]->SetFont(FF_ARIAL,FS_NORMAL,10);
$wp[$i]->SetFontColor('darkgreen');

// 凡例テキストを追加。
$wp[$i]->legend->SetText($legendtxt[$i]);

$graph->Add($wp[$i]);
}

// 作成したグラフをブラウザに返す。
$graph->Stroke();
?>

```

例 12.2.4_4

1. 任意のテキストブロック(段落)をグラフ・エリアに追加する方法
2. 軸線・同心円のグリッド間隔を手動設定する方法
3. 軸線・同心円の色の設定方法
4. 表示されるレンジのフォーマットのカスタマイズ方法
5. 中心円に任意のテキストブロック(段落)を追加する方法
6. 凡例の余白を調整する方法
7. コンパス方位に対して余白を拡大する方法

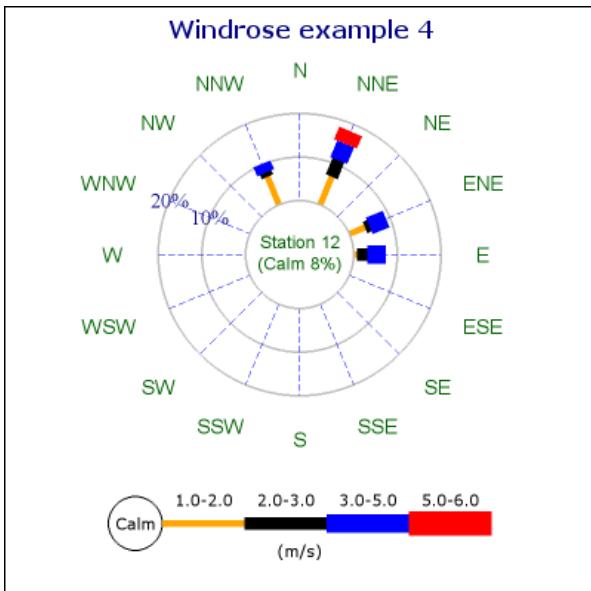


図 12.2.4_4

```

<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');

// データと方位ラベルは、方位軸番号と方位ラベルのどち
らでも指定できます。
$data = array(
    0 => array(1,1,2,5,4),
    1 => array(3,4,1,4),
    3 => array(2,7,4,4,3),
    5 => array(2,7,1,2));

// 新規の風配図グラフ・エリアを作成。
$graph = new WindroseGraph(400,400);

// グラフのタイトルを設定。
$graph->title->Set('Windrose example 4');
$graph->title->SetFont(FF_VERDANA,FS_BOLD,12);
$graph->title->SetColor('navy');

```

```
// 風配図プロットを作成。
$wp = new WindrosePlot($data);

// 目盛りラベルのフォントとフォント・カラーを設定。
$wp->scale->SetFont(FF_TIMES,FS_NORMAL,11);
$wp->scale->SetFontColor('navy');

// プロットの直径と位置を設定。
$wp->SetSize(190);

// 中心円の直径をプロットの直径の 38%に設定。
$wp->SetZCircleSize(0.38);
$wp->scale->SetZeroLabel("Station 12¥n(Calm %d%)");

// 中心円内のテキストのフォントとフォント・カラーを設定。
$wp->scale->SetZFont(FF_ARIAL,FS_NORMAL,9);
$wp->scale->SetZFontColor('darkgreen');

// コンパス方位ラベルのフォントとフォント・カラーを設定。
$wp->SetFont(FF_ARIAL,FS_NORMAL,10);
$wp->SetFontColor('darkgreen');

// コンパス方位ラベルの余白を設定。
$wp->SetLabelMargin(50);

// Grid を構成する同心円と方位軸の色を設定。
$wp->SetGridColor('silver','blue');

// テキスト「(m/s)」を凡例に追加。
// 凡例のマージンを設定。
$wp->legend->SetText('(m/s)');
$wp->legend->SetMargin(20,5);

// 作成したプロットをグラフ・エリアに追加し、グラフをブラウザに返す。
$graph->Add($wp);
$graph->Stroke();
?>
```

例 12.2.4_5

1. グラフ・エリアに任意のテキストブロックを追加する方法
2. Grid を構成する同心円と方位軸の手動設定の方法
3. 各プロットのまわりの枠線を追加する方法
4. コンパス方位ラベルのローカライズ方法
5. バケット・レンジにカスタマイズした色を設定する方法

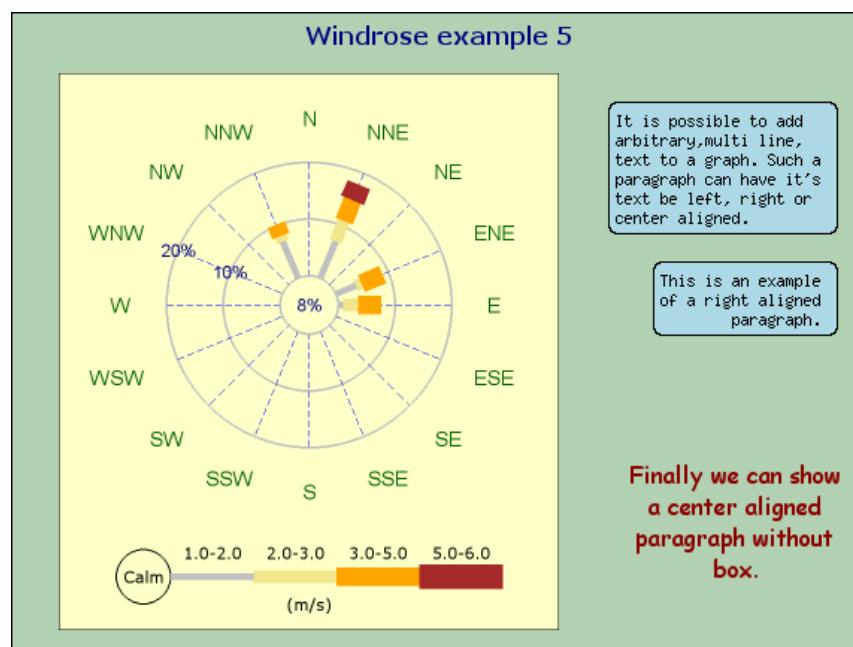


図 12.2.4_5

```

<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');

// データと方位ラベルは、方位軸番号と方位ラベルのどちらでも指定できます。

$data = array(
    0 => array(1,1,2,5,4),
    1 => array(3,4,1,4),
    3 => array(2,7,4,4,3),
    5 => array(2,7,1,2));

// テキストを配列に格納。
$txt=array();
$txt[0] = "It is possible to add arbitrary,multi line, text to a graph. ";
$txt[0] .= "Such a paragraph can have it's text be left, right or center ";
$txt[0] .= "aligned.";
$txt[1] = "This is an example of a right aligned paragraph.";
$txt[2] = "Finally we can show a center aligned paragraph without box.";

// 配列に各テキストボックスのレイアウトを格納し
// コードを整理する。
$txtdict = array(
    array(0.97,0.15,25,'left','black','lightblue'),
    array(0.97,0.4,20,'right','black','lightblue'),
    array(0.97,0.7,20,'center','darkred',false,FF_COMIC,F
S_NORMAL,12),
);

// レンジの色を配列に格納。
$rangeColors = array('silver','khaki','orange','brown','blue',
'navy','maroon','red');

// 新規の風配図グラフ・エリアを作成し、タイトルを設定する。
$graph = new WindroseGraph(570,430);
$graph->title->Set('Windrose example 5');
$graph->title->SetFont(FF_VARDANA,FS_BOLD,12);
$graph->title->SetColor('navy');

// グラフの背景色を設定。
$graph->SetColor('darkgreen@0.7');

// 定義済みのテキストボックス全てに対して設定を行う。
$n = count($txt);
for( $i=0; $i < $n; ++$i ) {
    $txtdict[$i] = new Text($txt[$i]);
    $txtdict[$i]->
>SetPos($txtdict[$i][0],$txtdict[$i][1],'right');
    $txtdict[$i]->SetWordwrap($txtdict[$i][2]);
}

```

```

$txtdict[$i]->SetParagraphAlign($txtdict[$i][3]);
$txtdict[$i]->SetColor($txtdict[$i][4]);
$txtdict[$i]->SetBox($txtdict[$i][5]);
if( count($txtdict[$i]) > 6 )
    $txtdict[$i]->SetFont($txtdict[$i][6],$txtdict[$i][7],$txtdict[$i][8]);
}
$graph->Add($txtdict);

// 風配図プロットを作成する。
Create the windrose plot.
$wp = new WindrosePlot($data);

// プロット・エリアの背景色を設定する。
$wp->SetColor('lightyellow');

// プロットの周りの枠線を追加する。
$wp->SetBox();

// 各パケット・レンジの色を設定する。
$wp->SetRangeColors($rangeColors);

// 目盛りラベルのフォントとフォント・カラーを設定する。
$wp->scale->SetFont(FF_ARIAL,FS_NORMAL,9);
$wp->scale->SetFontColor('navy');

// プロットの直径と位置を設定する。
$wp->SetSize(190);
$wp->SetPos(0.35,0.53);

$wp->SetZCircleSize(0.2);

// コンパス方位ラベルのフォントとフォント・カラーを設定する。
$wp->SetFont(FF_ARIAL,FS_NORMAL,10);
$wp->SetFontColor('darkgreen');

// コンパス方位のマージンを設定する。
$wp->SetLabelMargin(50);

// Grid を構成する同心円と方位軸の色を設定する。
$wp->SetGridColor('silver','blue');

// テキスト「(m/s)」を凡例に追加。
// 凡例のマージンを設定。
$wp->legend->SetText('(m/s)');
$wp->legend->SetMargin(20,5);

// プロットをグラフ・エリアに追加しグラフをブラウザに返す。
$graph->Add($wp);
$graph->Stroke();
?>

```

例 12.2.4_6

1. パケット・レンジの色を設定
2. 同心円と軸線の色を設定
3. ラベルのマージンの設定
4. 表示する凡例の値のフォーマットの変更

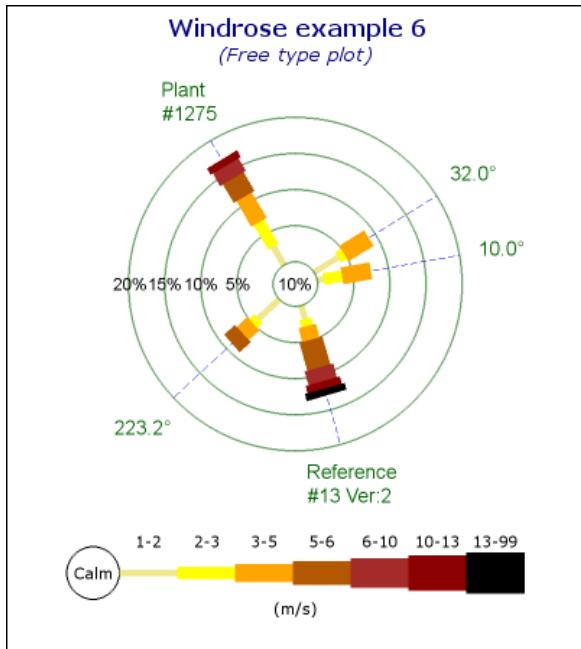


図 12.2.4_6

```
<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');

// データは方位軸番号でも方位ラベルでもどちらでも指定できます。
(ここで使われているのは「角度」では？)
$data = array(
    '10' => array(1,1,2.5,4),
    '32.0' => array(3,4,1,4),
    '120.5' => array(2,3,4,4,3,2,1),
    '223.2' => array(2,4,1,2,2),
    '285.7' => array(2,2,1,2,4,2,1,1)
);

// 方位ラベルのテキストを指定する。
$labels = array( '120.5' => "Plant\n#1275",
                 '285.7' => "Reference\n#13 Ver:2");

// パケット・レンジに使う色を配列に格納する。
$rangeColors = array('khaki','yellow','orange','orange:0.7','brown','darkred','black');

// 風配図グラフ・エリアを作成する。
First create a new windrose graph with a title
$graph = new WindroseGraph(400,450);
```

```
// タイトルを設定する
$graph->title->Set('Windrose example 6');
$graph->title->SetFont(FF_VARDANA,FS_BOLD,12);
$graph->title->SetColor('navy');

// サブタイトルを設定する
$graph->subtitle->Set('Free type plot');
$graph->subtitle->SetFont(FF_VARDANA,FS_ITALIC,10);
$graph->subtitle->SetColor('navy');

// 風配図プロットを作成する。
$wp = new WindrosePlot($data);

// フリー・タイプのプロットに設定する。
$wp->SetType(WINDROSE_TYPEFREE);

// ラベルを設定する。
$wp->SetLabels($labels);
$wp->SetLabelPosition(LBLPOSITION_CENTER);
$wp->SetLabelMargin(30);

// パケット・レンジの色を設定する。
$wp->SetRangeColors($rangeColors);

// 目盛りラベルのフォントとフォント・カラーを設定する。
$wp->scale->SetFont(FF_ARIAL,FS_NORMAL,9);

// プロットの直径と位置を設定する。
$wp->SetSize(230);
$wp->SetZCircleSize(30);

// コンパス方位ラベルのフォントとフォント・カラーを設定する。
$wp->SetFont(FF_ARIAL,FS_NORMAL,10);
$wp->SetTextColor('darkgreen');

// Grid を構成する同心円と方位軸の色を設定する。
$wp->SetGridColor('darkgreen@0.7','blue');

// テキスト「(m/s)」を凡例に追加。
$wp->legend->SetText('(m/s)');

// 凡例の値を小数点以下無しで表示する。
$wp->legend->SetFormat('%d');

// 作成したプロットをグラフ・エリアに追加しブラウザに返す
$graph->Add($wp);
$graph->Stroke();
?>
```

例 12.2.4_7

1. 方位ラベルの変更方法
2. パケット・レンジの調整方法
3. 表示している凡例の値のフォーマットの変更方法
4. 目盛りとなる同心円のレンジと間隔を手動で設定する方法

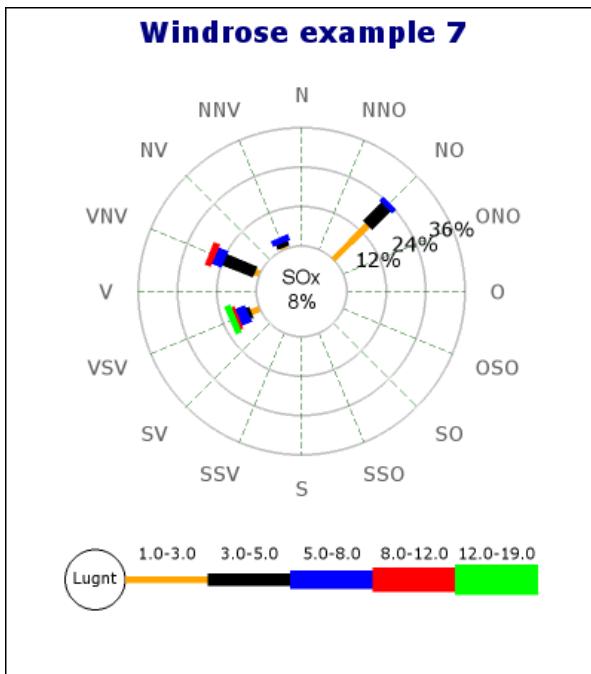


図 12.2.4_7

```
<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');

$data = array(
    2 => array(1,15,7.5,2),
    5 => array(1,1,1.5,2),
    7 => array(1,2,10,3,2),
    9 => array(2,3,1,3,1,2),
);

// 新規の風配図を作成し、タイトルを設定する。
$graph = new WindroseGraph(400,450);
$graph->title->Set('Windrose example 7');
$graph->title->SetFont(FF_VERDANA,FS_BOLD,14);
$graph->title->SetColor('navy');
```

```
// フリー・タイプの風配図プロットを作成する。
$wp = new WindrosePlot($data);
$wp->SetType(WINDROSE_TYPE16);

// 任意の文字列を中心円に追加する。
$wp->scale->SetZeroLabel("SOx¥n8%");

// コンパス方位ラベルをスウェーデン語にローカライズする。
// 注: コンパス方位ラベルはコンパス方位名と一致している必要があります。
$se_CompassLbl = array('O','ONO','NO','NNO','N','NNV','NV','VNV',
    'V','VSV','SV','SSV','S','SSO','SO','OSO','OSO');

$wp->SetCompassLabels($se_CompassLbl);

// 中心円の中の「Calm」という文字列をスウェーデン語の「Lugnt」に変更。
// 中心円を初期設定より少し大きく設定。
$se_calmtext = 'Lugnt';
$wp->legend->SetCircleText($se_calmtext);
$wp->legend->SetCircleRadius(20);

// 表示するパケット・レンジを調整。
$ranges = array(1,3,5,8,12,19,29);
$wp->SetRanges($ranges);
// $wp->SetAntiAlias(true);

// 同心円の目盛り間隔が必ず 12 で最大 30 になるよう設定。
$wp->scale->Set(30,12);

// プロットをグラフ・エリアに追加しグラフをブラウザに返す。
$graph->Add($wp);
$graph->Stroke();
?>
```

例 12.2.4_8

1. 方位軸 8 本タイプとは異なる軸線スタイルの設定
2. 部分的なラベルの追加

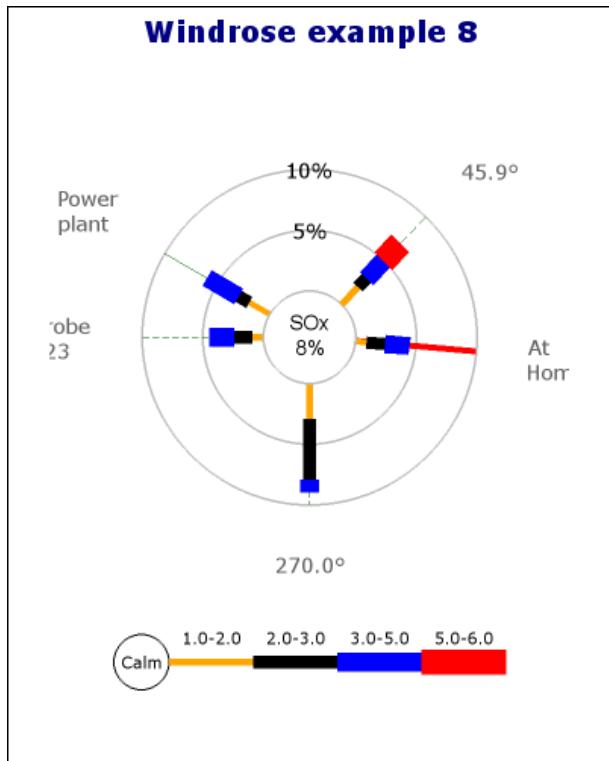


図 12.2.4_8 [ソース]

```

<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');

// データは方位軸番号でも方位ラベルでもどちらでも指定
// できます。
$data = array(
    '45.9' => array(3,2,1,2,2),
    355 => array(1,1,1,5,2),
    180 => array(1,1,1,5,2),
    150 => array(1,2,1,3),
    'S' => array(2,3,5,1),
);

```

```

// 方位のいくつかにラベルを追加。
$labels = array(355=>"At¥nHome base",180=>"Probe¥n
123",150=>"Power¥nplant");

// 方位軸線のいくつかに色、太さ、線の種類を定義する。
$axiscolors = array(355=>"red");
$axisweights = array(355=>8);
$axisstyles = array(355=>'solid',150=>'solid');

// 新規の風配図を作成し、タイトルを設定する。
First create a new windrose graph with a title
$graph = new WindroseGraph(400,500);
$graph->title->Set('Windrose example 8');
$graph->title->SetFont(FF_VERDANA,FS_BOLD,14);
$graph->title->SetColor('navy');

// フリー・タイプの風配図プロットを作成。
$wp = new WindrosePlot($data);
$wp->SetType(WINDROSE_TYPEFREE);

// 方位軸線の色、太さ、線の種類を指定する。
$wp->SetRadialColors($axiscolors);
$wp->SetRadialWeights($axisweights);
$wp->SetRadialStyles($axisstyles);

// ラベルのいくつかを追加する。
$wp->SetLabels($labels);

// 任意のテキストを中心円に追加する。
$wp->scale->SetZeroLabel("SOx¥n8%");

// プロットをグラフ・エリアに追加してグラフをブラウザに
// 返す。
$graph->Add($wp);
$graph->Stroke();
?>

```

例 12.2.4_9

16 本方位軸タイプに設定

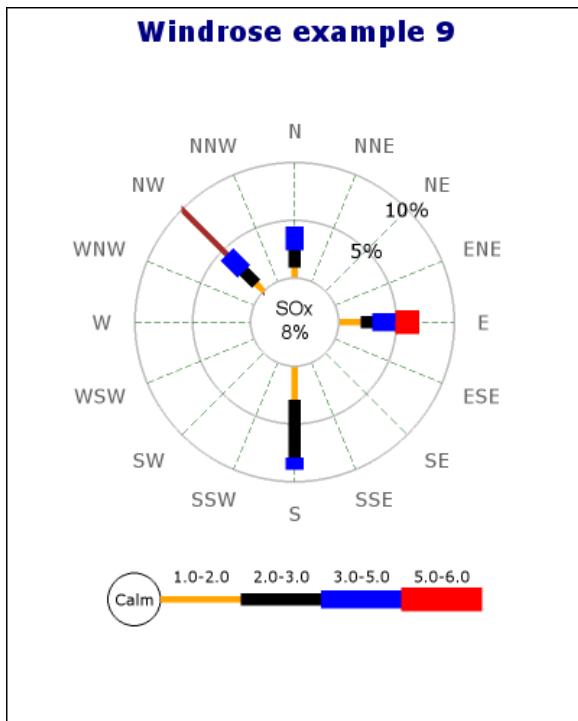


図 12.2.4_9

```

<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');

// データは方位軸番号でも方位ラベルでもどちらでも指定できます。
$data = array(
    'E' => array(3,2,1,2,2),
    'N' => array(1,1,1,5,2),
    'nw' => array(1,1,1,5,2),
    'S' => array(2,3,5,1),
);
// 方位軸線のいくつかに個別に色、太さ、線の種類を定義する。
// 方位軸線はローカライズしたラベルでも方位軸番号でもどちらでも指定可能。
// 注:プロット上の軸線を何本にするかによって、軸番号の最大値は
// その本数から1引いた数になります。(0 からスタート。)
$axiscolors = array('nw'=>'brown');
$axisweights = array('nw'=>8); // 6 => 8 でも可能
$axisstyles = array('nw'=>'solid');

// 新規の風配図を作成し、タイトルを設定する。
$graph = new WindroseGraph(400,500);
$graph->title->Set('Windrose example 9');
$graph->title->SetFont(FF_VERDANA,FS_BOLD,14);
$graph->title->SetColor('navy');

// フリー・タイプの風配図プロットを作成する。
$wp = new WindrosePlot($data);
$wp->SetType(WINDROSE_TYPE16);

// 方位軸線のいくつかに個別に色、太さ、線の種類を指定する
$wp->SetRadialColors($axiscolors);
$wp->SetRadialWeights($axisweights);
$wp->SetRadialStyles($axisstyles);

// 中心円に任意のテキストを追加する。
$wp->scale->SetZeroLabel("SOx¥n8%");

// 作成したプロットをグラフ・エリアに追加しブラウザに返す
$graph->Add($wp);
$graph->Stroke();
?>

```

例 12.2.4_10

国旗の画像を背景に追加する

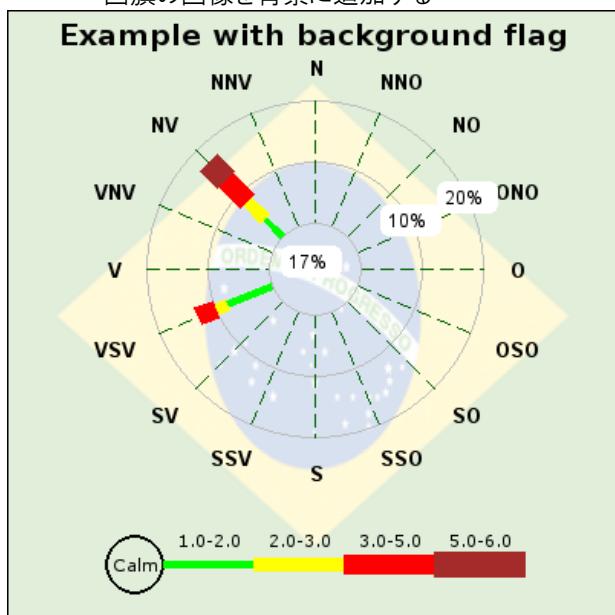


図 12.2.4_10

```

<?php // content="text/plain; charset=utf-8"
require_once ('jpgraph/jpgraph.php');
require_once ('jpgraph/jpgraph_windrose.php');
require_once ('jpgraph/jpgraph_flags.php');

// データは方位軸番号でも方位ラベルでもどちらでも指定できます。
$data2 = array(
    'vsv' => array(12,8,2,3),
    6 => array(5,4,4,5,4),
);

$se_CompassLbl = array('O','ONO','NO','NNO','N','NNV','NV',
    'VNV','V','VSV','SV','SSV','S','SSO','SO','OSO');

// 新規の小さい風配図グラフ・エリアを作成する。
$graph = new WindroseGraph(400,400);
$graph->SetMargin(25,25,25,25);
$graph->SetFrame();

$graph->title->Set('Example with background flag');
$graph->title->SetFont(FF_VERA,FS_BOLD,14);

// $graph-
>SetBackgroundImage('bkgrndimg.jpg',BGIMG_FILLFRAME);

// $graph->SetBackgroundImageMix(90);
$graph->SetBackgroundCFlag(28,BGIMG_FILLFRAME,15);

$wp2 = new WindrosePlot($data2);
$wp2->SetType(WINDROSE_TYPE16);
$wp2->SetSize(0.55);
$wp2->SetPos(0.5,0.5);
$wp2->SetAntiAlias(false);

$wp2->SetFont(FF_ARIAL,FS_BOLD,10);
$wp2->SetFontColor('black');

$wp2->SetCompassLabels($se_CompassLbl);
$wp2->legend->SetMargin(20,5);

$wp2->scale->SetZFont(FF_ARIAL,FS_NORMAL,8);
$wp2->scale->SetFont(FF_ARIAL,FS_NORMAL,9);
$wp2->scale->SetLabelFillColor('white','white');

$wp2-
>SetRangeColors(array('green','yellow','red','brown'));

$graph->Add($wp2);
$graph->Stroke();
?>

```

12.3 バーコード

このモジュールはエンタープライズ版のライブラリにのみ含まれています。

バーコード拡張モジュールでは、一般に使われているコードはどれでも使用してバーコードを作成することができます。ここでは一次元バーコードについて説明します。完成したバーコードは PNG 画像、JPEG 画像、または高解像度プリンタ印刷用の Postscript ファイルとして生成されます。

バーコード拡張モジュールは大規模なデータ検証を可能にし、作成されたバーコードは公式な適用企画に確実に対応しています。

欧洲でほとんどサポートされていない CODE11 を除き、JpGraph が作成するすべてのバーコードは Metrologic CCD-47 スキャナによる広範な検証を経てています。

バーコードを作成するには「jpgraph_barcode.php」モジュールを含む必要があります。

バーコードには数種類ありますが、すべて同じ原則に従います。コードの種類がいくつか存在する理由は、ひとつには、「数値データのみ」や「数値とアルファベットの混合データ」など異なる要求に対応するためであり、また、同様の課題に対して複数のメーカーが開発・製造したことにもあります。ライブラリは一般的な一次元バーコードのほとんどに対応しています。これらのタイプのバーコードはたいていの消費財についているものと同様です。

異なる種類のバーコードはシンボロジーというバーコードの種類として分類・参照されます。

よく使われるシンボロジーのほとんどは ISO 標準や IEC 標準に昇格しました。ISO 標準として認可されていないバーコードの一部は AIM(<http://www.aimglobal.org>)をご参照ください。米国では Uniform Code Council (UCC) という機関がバーコードに使用される小売業界のコード発行を管理しています。

これらの標準は広範囲のバーコードをカバーしている一方、エンド・ユーザにはほとんどメリットはありません。バーコードを選択するとき、エンド・ユーザにとって最重要課題は次のようなことが考えられます。

1. このコードでどんな文字が暗号化できるのか。
2. 入力値に対してバーコードはどれくらいの大きさになるのか(どれくらい効率的か)。
一般的に、一次元バーコードはアルファベットや数値の文字列を最大 20 文字に暗号化します。

注意：多くの種類のバーコードは理論的には無限の文字数に対応できますが、実際にはバーコード・リーダーやスキャナの読み取りができるサイズによって限界があります。バーコード・リーダーのほとんどは 10cm より長いバーコードの場合、信頼できるレベルでの読み取りができません。

3. 物理的損傷に対してどれくらいの強度があるか。つまり、バーコードが何パーセントくらいまで破壊されてもまだスキャナに読み取れるのか。

一次元バーコードの許容値はたいてい低いものです。バーが1本でも読めなくなると、バーコード全部が読み取り不可能にできなくなります。バーコードの種類によってはチェック・ディジットを含んだものもありますが損傷データを再作成するには十分ではなく、データが編さんされていないかどうかを証明するのみです。それさえ、2つのエラーが偶然、本来エラーを示すはずのチェック・ディジットを正しい数字にしてしまう可能性もあります。

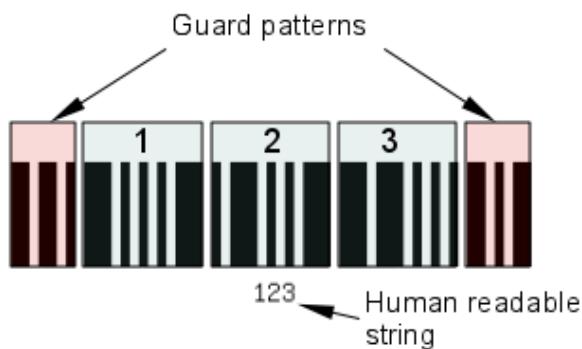
一次元バーコードの強度を上げるには通常、物理的にバーコードを大きくし、高さを持たせます。

一次元バーコードは今でもレガシーとして広く使われていますが、新しいアプリケーションで一次元バーコードを使っているものはほとんどありません。上にあげたような、強度や柔軟性に対する限界がその理由の一部です。新しいアプリケーションのほとんどはRFIDタグか二次元バーコードを使っています。データ許容量と強度が遙かに優れているからです。

12.3.1 バーコードの仕組み

一次元バーコードはサポートされている文字が変換されて、幅の異なる棒線の組合せから成るバーコードとして表示されるように作成されます。バーコードの最初と最後をスキャナが識別するために特別な「ガード」パターンが使われ、どのタイプのバーコード(シンボロジー)かも同時に識別されます。

下の図は「Code25」シンボロジーを使用したバーコードを拡大・修正表示したもので、数時の「123」を暗号化しています。最初と最後のガード・パターンを色づけて、3桁の数字がどのように暗号化されたかが解るようにしてみました。



こでは理解しやすいようにCode25のバーコードを使っています。各文字はスペースで区切られた5本の黒い棒でできています。2本の太い棒と3本の細い棒、即ち2種類の棒5本の並び方ということで25と呼ばれます。一番細い棒の幅はモジュール幅とも呼ばれます。他のコードはもっと

複雑なパターンで高密度、即ち同じ幅により多くの文字が暗号化されていますが、原則は同じです。

典型的なスキャナは光を送って反射させ、バーコードを左から右へ読み取ります。白黒の領域の相対的なサイズを計測し、暗号化された文字を変換し直します。

人が読んで理解できる文字列は、バーコードの下、厳密にはバーコードには含まれない場所に表示されます。これはバーコードの内容を人が理解できるように付けられているだけです。

バーコードによっては一番後ろにチェック・ディジットを含んでいます。チェック・ディジットは変換されたデータが正しく読み込まれているかを検証するために使われます。

12.3.2 バーコードの種類(シンボロジー)

12.3.2.1 数字のみのシンボロジー

以下のコードは、「-」「+」「:」「\$」のような記号の一部と数値データのみサポートする種類のバーコードです。各バーコードの詳細は Section 23.9 をご参照ください。

- **Codabar** 米国の一図書館以外では現在あまり使用されていない古いコード。
- **Code 11** 主にラベリング電気通信機器に使用
- **EAN-13 (ISO/IEC 15420)** 13 行の欧洲統一商品番号。国際的小売商品コード。米国では「UPC-13」と知られる。
- **EAN-8** 8 行の欧洲統一商品番号。EAN コードの圧縮版でサイズの小さい商品に使用。
- **Industrial 2 of 5** 低密度の古いコード。現在は一般には使われていない。
- **Interleaved 2 of 5 (ISO/IEC 16390)** コンパクトな数字コードで、産業界、運転免許証、輸送など幅広く使用される。
- **UPC-A** 米国とカナダのほぼすべての小売商品で見られるユニバーサルな商品コード。
- **UPC-E** UPC-A コードの圧縮版でサイズの小さい商品に使用。
- **Bookland** 国際標準図書番号である ISBN コードで、書籍・雑誌に使われる。

12.3.2.2 アルファベットと数字のシンボロジー

以下のコードは、「-」「+」「&」「#」「!」のような記号の一部とアルファベット、数値データをサポートするバーコードのシンボロジーです。

- **Code 128 (ISO/IEC 15417)** 幅広く使われる柔軟で大容量のコード。Code 128 の変形も存在し(EAN-128 など)、厳密な構成のデータを暗号化する。
- **EAN 128 (ISO/IEC 15420)** 構成化された Code 128 の変形。バーコードのコードそのものではなく、Code 128 に入力するデータのフォーマット構成。
- **Code 39 (ISO/IEC 16388)** 一般目的のコードで世界中で使用されている。

12.3.2.3 使用するシンボロジーの選択

通常はアプリケーションが「標準」を決定し、それ以外は適当な業界標準に従うしか選択肢はありません。

しかし用途が内部使用のみに厳密に限られる場合はどのようなコードでも選択できます。一般的な用途であれば Code 39 か Code 128 を推奨します。柔軟性の富み、長い文字列に対応し、データサイズに対する必要スペースの比率がかなり効率的なためです。

12.3.3 バーコード・モジュールで使える機能

以下に JpGraph バーコード拡張モジュールで使える機能をまとめました。

- 対応しているバーコード(データ妥当性検証を含むシンボロジー)の種類は下記のとおりです。
 1. UPC A
 2. UPC E
 3. EAN 128
 4. EAN 13
 5. EAN 8
 6. CODE 11 (USD-8)
 7. CODE 39
 8. CODE 128
 9. Industrial 2 of 5
 10. Interleaved 2 of 5
 11. Codabar
 12. Bookland (ISBN)
- 入力データは各コードの公定規格に対して検証され、無効なバーコードの作成はできないように設定されています。
- 出力フォーマット
 1. UPC A
 2. UPC E
 3. EAN 128
- 選択可能なフォーマット

1. ユーザ指定のモジュール幅。バーコードの中で一番小さい単位の幅をコントロールします。
2. チェックサムが選択できるバーコードの場合、チェックサムの自動計算を行う。
3. ユーザ指定のバーコード目盛り設定
4. 水平および垂直のバーコード・レイアウト
5. 読解できるテキストの抑制
6. 読解できるテキストのフォント選択

コードの種類を指す「シンボロジー」という言葉は、暗号化方法と特定のバーコード標準によるバーのレイアウトとの組み合わせを指します。上に挙げた標準コード(シンボロジー)は現在でも広く使用されています。バーコードの種類の多くは機能が重複しており、どのコードを使用すべきかを指定する厳格な標準が存在しない限り、ひとつのアプリケーションで複数のコードを使用することも可能かもしれません。

重複する機能を持つコードが数多く存在する理由は、特許を取得したバーコードで多くの企業が市場シェアを競い、時には顧客を特定のブランドのスキャナに縛り付けようとしたことに起因していると考えられます。初期のバーコードが単独では新規の需要を十分に満たせなかった結果ともいえます。

12.3.4 バーコードを作成する - クイック・スタート

一次元バーコードの作成はすべて、一律の手順に従います。

1. 選択したコードの暗号化インスタンスを作成する。
2. 選択したアウトプット・フォーマット(画像かポストスクリプト)のバックエンド・インスタンスを作成する。
3. データを暗号化し、バーコードを生成する。

通常、基本バーコードを作成するには3行のコードしか必要としません。

例えば、次のスクリプトは、「CODE 39」というシンボロジーを使用し文字列「ABC123」を表す画像を暗号化したバーコードを作成するものです。

```
<?php
require_once('jpgraph_barcode.php');

$symbology = BarcodeFactory::Create(ENCODING_CODE39);
$barcode = BackendFactory ::Create(BACKEND_IMAGE, $symbology);
$barcode ->Stroke('ABC123');
?>
```

作成されるバーコードは次のようにになります。



ABC123

図 12.3.2 CODE39 で暗号化した「ABC123」

このことかスクリプトから解るように、ライブラリへの基本インターフェースはシンボロジーによる暗号化とバックエンド出力という2つの理論上の「ファクトリー」を利用します。この設計はライブラリのエンド・ユーザーにとって、わかりやすい新しい出力フォーマットと新しいシンボロジーを追加することになります。

「CODE 128」というバーコードを使用してデータ文字列を暗号化する場合、最初の行だけを変更し、下記のように記述します。

```
<?php
require_once('jpgraph_barcode.php');

$symbology = BarcodeFactory::Create(ENCODING_CODE128);
$barcode = BackendFactory ::Create(BACKEND_IMAGE, $symbology);
$barcode ->Stroke('ABC123');
?>
```

作成されるバーコードは次のようにになります。



図 12.3.3 CODE 128 で暗号化した「ABC123」

上の2つの例で見たように、記号定数を使うことにより、バックエンド画像とシンボロジーの両方を指定できます。ライブラリがサポートしているシンボロジーを指定する記号定数は下記のとおりです。

1. ENCODING_EAN128
2. ENCODING_EAN13
3. ENCODING_EAN8
4. ENCODING_UPCA
5. ENCODING_UPCE
6. ENCODING_CODE39
7. ENCODING_CODE128
8. ENCODING_CODE25
9. ENCODING_CODE125
10. ENCODING_CODABAR
11. ENCODING_CODE11
12. ENCODING_BOOKLAND

12.3.5 エラー・ハンドリング

バーコード・モジュールはライブラリの標準エラー・ハンドリングを使用します。エラーが発生した場合(多くはデータ検証が失敗した場合)に備えて例外処理を設けます。エラーは次の2つの方法で処理されます。

1. try { ... } catch { ... }ステートメントでスクリプトを囲む方法。
2. 通常の初期設定によるエラー・ハンドラー、set_exception_handler()をインストールする方法。
ライブラリがエラーに反応したら、JpGraphException クラスのインスタンスで指定されたエラー・ハンドラーが呼び出されます。

次のコードは try-catch ステートメントの使用例です。

```
<?php
try {
    $encoder = BarcodeFactory::Create(ENCODING_CODE39);
    $e = BackendFactory::Create(BACKEND_IMAGE,$encoder);
    $e->Stroke('abc123');
} catch( JpGraphException $e ) {
    echo 'Error: ' . $e->getMessage()."\\n";
}
?>
```

```
JpGraphError::Raise($e->getMessage());
```

このコードは次のような画像エラーを表示します。



図 12.3.3 画像エラー：バーコードのデータ検証に失敗した例

次に、コマンド・ライン・バーコード・ユーティリティで、他の初期設定エラー・ハンドラーをインストールする別の方法を紹介します。

12.3.6 コマンドラインを使用してバーコードを作成する

ライブラリはコマンドライン・ユーティリティをライブラリ使用方法の追加例として含んでいます。

- mkbarcode.php

このユーティリティは「barcode/」ディレクトリの中に入り、次のようにコマンドラインで使います。

```
$>php mkbarcode.php -b code39 ABC123 > barcode.png
```

このコマンドラインは Code 39 シンボロジーを使用してデータ文字列「abc123」のバーコードを作成し、生成された画像を「barcode.png」ファイルに保存します。

このコマンドライン・ユーティリティはクライアント側にもコマンドラインの PHP が存在していることを前提としています。

このユーティリティは次のような構文を含みます。

```
mkbarcode.php -b <symbology> [-r -h -c -o <output format> -m <width> -s <scale>]
-y <height> -f <filename> ] datastring
```

-b	使用するシンボロジーは下記より1つ選択(大文字・小文字区別なし) UPCA UPCE EAN128 EAN13 EAN8 CODE11 CODE39 CODE128 CODE25 CODEI25 CODABAR BOOKLAND
-c	チェックディジットに対応するシンボロジーにはディジットを追加
-o	出力フォーマット。0=Image, 1=PS, 2=EPS
-m	モジュールの
-s	スケール・ファクター
-h	このヘルプを表示
-f	書き込むファイル名
-r	バーコードを 90 度回転させる
-y height	高さをピクセルで設定
-x	読み解可能なテキストを非表示にする
-silent	サイレント。エラー・メッセージを表示しない

ファイル名が指定されていない場合、画像をファイルに変換するように促す表示が返されます。

12.4 テーブル

このモジュールはエンタープライズ版のライブラリにのみ含まれています。

データを視覚化するときには、全体の傾向がつかめる簡単なグラフィック表示と実際の数値が詳細に記載されたテーブル・ビューの両方があれば便利でしょう。テーブル・モジュールはほぼ無限にバリエティに富んだテーブルの作成をサポートします。

テーブル・ライブラリ・モジュールによって必要な行数や列数、サイズの選択、セルの結合、個別セルの背景色、フォント、枠線の設定など、ほとんどどのような表でも作成することができます。

テーブルはオブジェクトとして作成されるため、サンプル・アイコンやテキスト・オブジェクトとほとんど同じように、通常のグラフ・エリアに追加できます。

下に表示した様々なテーブルは、ライブラリに備わった機能を示すものです。この章では使用可能な API を豊富なサンプルによって紹介します。

テーブルは CSIM(クライアント・サイド・イメージ・マップ)を完全にサポートしているので、ユーザはローンチ・パッドで更に詳細な情報も得ることができます。

データ・テーブルをグラフ・エリアに追加するだけでなく、単独でテーブルだけを作成することもできます。HTML テーブルとは異なりフォーマットを維持したままグラフィック・テーブルのコピーができるという面で優れています。

	w631	w632	w633	w634	w635	w636
Critical (sum)	13	17	15	8	3	9
High (sum)	34	35	26	20	22	16
Low (sum)	41	43	49	45	51	47
Sum	88	95	90	73	76	72

	w631	w632	w633	w634	w635	w636
Critical (sum)	13	17	15	8	3	9
High (sum)	34	35	26	20	22	16
Low (sum)	41	43	49	45	51	47
Sum:	88	95	90	73	76	72

2007						
	Q1		Q2			
	Jan	Feb	Mar	Apr	May	
Min	15.2	12.5	9.9	70.0	22.4	21.5
Max	23.9	14.2	18.6	71.3	66.8	42.6

	1	2	3	4
5	6	7	8	
6	8	10	12	

	1.0	2.0	3.0	4.0
5.0	6.0	7.0	8.0	
6.0	8.0	10.0	12.0	

	Jan	Feb	Mar	Apr	May	Jun
Min	15.2	12.5	9.9	70.0	22.4	21.5
Max	23.9	14.2	18.6	71.3	66.8	42.6
Sum:	39.1	26.7	28.5	141.3	89.2	64.1

Areas					
USA	UK	France	Denmark	Iceland	Canada
Feb	13	17	15	8	3
Mar	34	35	26	20	22
Apr	41	43	49	45	51
Sum:	88	95	90	73	76

	Jan	Feb	Mar	Apr	May	Jun
Team 1	15.2	12.5	9.9	70.0	22.4	21.5
Team 2	23.9	14.2	18.6	71.3	66.8	42.6
Sum:	39.1	26.7	28.5	141.3	89.2	64.1

	Feb	Mar	Apr	May	Jun
Team 1	12.5	9.9	70.0	22.4	21.5
Team 2	14.2	18.6	71.3	66.8	42.6
Sum:	26.7	28.5	141.3	89.2	64.1

図 12.4_1 スタンドアローンのテーブル例

上に表示したスタンドアローンのテーブルの他に、グラフとテーブルを組合せて使うこともできます。その場合、下の図のように表示されます。

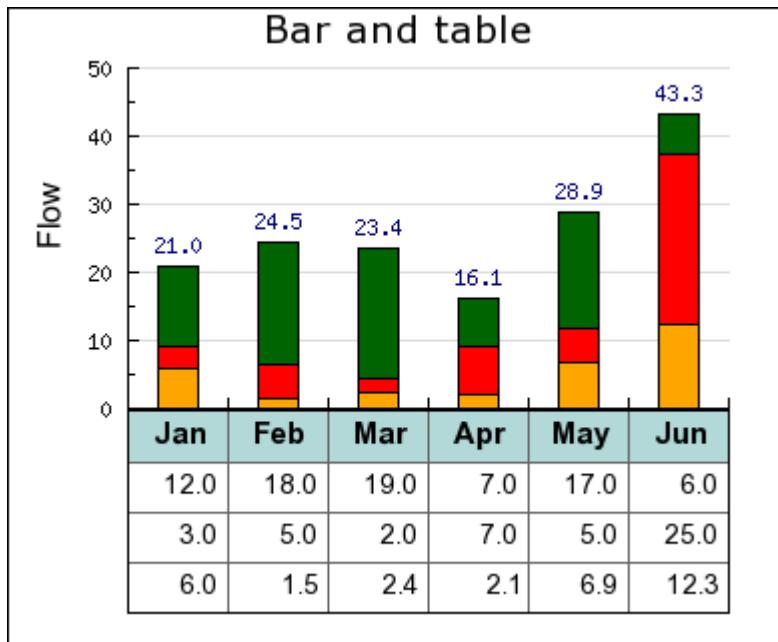


図 12.4_2 グラフィック・テーブルと棒グラフの組合せ

12.4.1 テーブルの構成

ここではテーブルの基本的なコンセプトを紹介し、利用可能な基本フォーマットについて説明します。

12.4.1.1 基本テーブルを作成する

テーブルは GTextTable クラスのインスタンスとして作成され、サンプル・アイコンや、テキスト、その他のプロット・オブジェクトとほぼ同様に操作できます。Graph::Add メソッドによって、X 軸 Y 軸の座標のグラフに追加することもできます。

テーブルの作成は「jpgraph_table.php」ファイルを呼び出すことから始めます。このファイルには GTextTable クラスのクラス定義が含まれています。テーブルを作成するには、通常のグラフ・エリアに使われる Graph クラスのインスタンスをまず作成するか、それとも CanvasGraph クラスのインスタンスで単体のテーブルを作成するかを決める必要があります。

どちらにするかを決めれば、新規テーブルの作成は、GTextTable クラスのインスタンスを作成し、行や列のサイズを決定するための初期設定メソッドを呼び出すだけできます。単体のテーブルを表示するには、グラフ・エリアである CanvasGraph も作成してテーブルをグラフに追加する必要があります。

```
<?php
require_once "jpgraph/jpgraph.php";
```

```

require_once 'jpgraph/jpgraph_canvas.php';
require_once "jpgraph/jpgraph_table.php";

// 基本となるキャンバス・グラフを設定する。
$graph = new CanvasGraph(630,600);

// 基本グラフ・エリアを作成する。
$table = new GTextTable();
$table->Init(5,7); // Create a 5 rows x 7 columns table

...

// テーブルをグラフ・エリアに追加する。
$graph->Add($table);

// グラフ・エリアをクライアントに返す。
$graph->Stroke();
?>

```

これまで見てきたように、テーブルの実際のサイズを指定することは（可能ですが）必要ではありません。実際のテーブル・サイズはテーブルのコンテンツによって自動的に決定されます。

テーブル内のセルには連番が付加されます。 $(0,0)$ は一番上の左端、 $(n-1, m-1)$ は一番下の右端で、 n が行数、 m が列数です。

次に、テーブルにデータを入力していきます。各セル内の内容は次のメソッドの中から指定します。

- GTextTable::Set(\$aRow, \$aCol, \$aText)
Set the specified text in cell (\$aRow, \$aCol)
GTextTable::Set(\$aData)
二次元配列内のデータ、\$aData をテーブルにコピーします。
- GTextTable::SetImage(\$aRow, \$aCol, \$aText, \$alimage)
セル(\$aRow, \$aCol)の中の画像を設定します。
- GTextTable::SetCellCountryFlag(\$aRow,\$aCol,\$aFlag,\$aScale=1.0,\$aMix=100,\$aStdSize=3)
指定した国旗をセル(\$aRow, \$aCol)に設定します。

The Set()メソッドは2つの異なる方法で呼び出すことができます。

ひとつは、ゼロで始まる行・列番号で個別のセルの値を指定する方法です。例えば、上の例を継続すると、一番上の行(0)の1列目と2列目の2つのセルに次のように値を入れます。

```

...
$table->Set(0,0,1);
$table->Set(0,1,1);
...

```

もうひとつは（もう少し簡潔な方法かもしれません）、適切な値を含む二次元配列をそれぞれ、最初でかつ最後の引数として Set() メソッドに格納する方法で、次のように記述します。

```
$data = array( array( 12, 7), array( 10, 5)); $table->Set($data)
```

テーブルをマトリックスとして直接指定するなら、マトリックス自体のサイズが自動的に決定するので最初の Init() メソッドを呼び出す必要はありません。

次の基本サンプルコードは連続した値を格納した 2 行 4 列の配列テーブルを作成する方法です。ここではテーブルを追加するグラフ・エリアを準備するために CanvasGraph を使っていきます。このスクリプトの結果は次のように表示されます。

例 12.4.1 最も基本的な 2 行 4 列テーブル

```
(File: table_howto1.php)
<?php // content="text/plain; charset=utf-8"
require_once 'jpgraph/jpgraph.php';
require_once 'jpgraph/jpgraph_canvas.php';
require_once 'jpgraph/jpgraph_table.php';

// Create a canvas graph where the table can be added
$graph = new CanvasGraph(70,50);

// Setup the basic table
$data = array( array(1,2,3,4),array(5,6,7,8));
$table = new GTextTable();
$table->Set($data);

// Add the table to the graph
$graph->Add($table);

// ... and send back the table to the client
$graph->Stroke();

?>
```

1	2	3	4
5	6	7	8

図 12.4.1.1 最も基本的な 2 行 4 列テーブル

上の例では、簡単なショートカットも利用しています。テーブルに入力するデータが 2 次元配列で指定されている場合、テーブル・サイズは与えられた配列によって決定されるので、Init() メソッドを呼び出す必要ありません。

12.4.2 サンプル集

例 12.4.2_1

	w631	w632	w633	w634	w635	w636
Critical (sum)	13	17	15	8	3	9
High (sum)	34	35	26	20	22	16
Low (sum)	41	43	49	45	51	47
Sum:	88	95	90	73	76	72

```
<?php // content="text/plain; charset=utf-8"
require_once 'jpgraph/jpgraph.php';
require_once 'jpgraph/jpgraph_canvas.php';
require_once 'jpgraph/jpgraph_table.php';

// 新規のキャンバスグラフをグラフ・エリアとして設定
$graph = new CanvasGraph(430,150);

// 基本テーブルのデータを設定
$data = array(
    array("", 'w631','w632','w633','w634','w635','w636'),
    array('Critical (sum)',13,17,15,8,3,9),
    array('High (sum)',34,35,26,20,22,16),
    array('Low (sum)',41,43,49,45,51,47),
    array('Sum:',88,95,90,73,76,72)
);
```

```
// 新規テーブルを作成しデータを格納。フォントを設定
$table = new GTextTable();
$table->Set($data);
$table->SetFont(FF_TIMES,FS_NORMAL,11);

// デフォルトのテーブルの配置を設定
$table->SetAlign('right');

// テーブルをグラフ・エリアに追加
$graph->Add($table);

// グラフをブラウザに返す
$graph->Stroke();

?>
```

例 12.4.2_2

	w631	w632	w633	w634	w635	w636
Critical (sum)	13	17	15	8	3	9
High (sum)	34	35	26	20	22	16
Low (sum)	41	43	49	45	51	47
Sum:	88	95	90	73	76	72

```
<?php // content="text/plain; charset=utf-8"
require_once 'jpgraph/jpgraph.php';
require_once 'jpgraph/jpgraph_canvas.php';
require_once 'jpgraph/jpgraph_table.php';

// 新規のキャンバスグラフをグラフ・エリアとして設定
$graph = new CanvasGraph(430,150);

// 基本テーブルのデータを設定
$data = array(
    array("", 'w631','w632','w633','w634','w635','w636'),
    array('Critical (sum)',13,17,15,8,3,9),
```

```
array('High (sum)',34,35,26,20,22,16),
array('Low (sum)',41,43,49,45,51,47),
array('Sum:',88,95,90,73,76,72)
);

// 新規テーブルを作成しデータを格納
$table = new GTextTable();
$table->Set($data);

// フォントを設定
$table->SetFont(FF_TIMES,FS_NORMAL,11);
$table->SetColFont(0,FF_ARIAL,FS_NORMAL,11);
```

```
$table->SetRowFont(0,FF_ARIAL,FS_NORMAL,11);
$table->SetRowFont(4,FF_TIMES,FS_BOLD,14);

//グリッド線を非表示にする
$table->SetGrid(0);

//色の設定
$table->SetRowFillColor(0,'lightgray@0.5');
$table->SetRowFillColor(4,'lightgray@0.5');
$table->SetColFillColor(0,'lightgray@0.5');
$table->SetFillColor(0,0,4,0,'lightgray@0.5');
```

```
//デフォルトの最小列幅を設定
$table->SetMinColWidth(45);

// デフォルトのテーブルの配置を設定
$table->SetAlign('right');

// テーブルをグラフ・エリアに追加
$graph->Add($table);

// グラフをブラウザに返す
$graph->Stroke();

?>
```

例 12.4.2_3

	w631	w632	w633	w634	w635	w636
Critical (sum)	13	17	15	8	3	9
High (sum)	34	35	26	20	22	16
Low (sum)	41	43	49	45	51	47
Sum	88	95	90	73	76	72

```
<?php // content="text/plain; charset=utf-8"
require_once 'jpgraph/jpgraph.php';
require_once 'jpgraph/jpgraph_canvas.php';
require_once 'jpgraph/jpgraph_table.php';

// 新規のキャンバスグラフをグラフ・エリアとして設定
$graph = new CanvasGraph(430,150);

// 基本テーブルのデータを設定
$data = array(
    array("", 'w631', 'w632', 'w633', 'w634', 'w635', 'w636'),
    array('Critical (sum)',13,17,15,8,3,9),
    array('High (sum)',34,35,26,20,22,16),
    array('Low (sum)',41,43,49,45,51,47),
    array('Sum',88,95,90,73,76,72)
);

// 新規基本テーブルを作成しデータを格納する
$table = new GTextTable();
$table->Set($data);
$table->SetFont(FF_ARIAL,FS_NORMAL,11);

// デフォルトの列の幅を設定
$table->SetMinColWidth(40);

// デフォルトのテーブルの配置を設定
$table->SetAlign('right');
```

```
//囲み線を非表示にする
$table->SetBorder(0);

//グリッド線を非表示にする
$table->SetGrid(0);

//行4と行0にフォントを設定
$table->SetRowFont(4,FF_ARIAL,FS_BOLD,11);
$table->SetRowFont(0,FF_ARIAL,FS_BOLD,11);

//色の設定
$table->SetRowFillColor(4,'orange@0.5');
$table->SetFillColor(0,1,0,6,'teal@0.8');

//グリッド線を設定
$table->SetRowGrid(4,1,'black',TGRID_DOUBLE2);
$table->SetColGrid(1,1,'black',TGRID_SINGLE);
$table->SetRowGrid(1,1,'black',TGRID_SINGLE);

// テーブルをグラフ・エリアに追加
$graph->Add($table);

// グラフをブラウザに返す
$graph->Stroke();

?>
```

例 12.4.2_4

	w631	w632	w633	w634	w635	w636
Critical (sum)	13	17	15	8	3	9
High (sum)	34	35	26	20	22	16
Low (sum)	41	43	49	45	51	47
Sum:	88	95	90	73	76	72

```

<?php // content="text/plain; charset=utf-8"
require_once 'jpgraph/jpgraph.php';
require_once 'jpgraph/jpgraph_canvas.php';
require_once 'jpgraph/jpgraph_table.php';

// 新規のキャンバスグラフをグラフ・エリアとして設定
$graph = new CanvasGraph(430,150);

// 基本テーブルのデータを設定
$data = array(
    array("", 'w631','w632','w633','w634','w6
35','w636'),
    array('Critical (sum)',13,17,15,8,3,9),
    array('High (sum)',34,35,26,20,22,16),
    array('Low (sum)',41,43,49,45,51,47),
    array('Sum:',88,95,90,73,76,72),
);

// 新規テーブルを作成しデータを格納。フォントを設
定
$table = new GTextTable();
$table->Set($data);
$table->SetFont(FF_ARIAL,FS_NORMAL,11);

// デフォルトの最小列幅を設定
$table->SetMinColWidth(40);

// デフォルトのテーブルの配置を設定
$table->SetAlign('right');

// グリッド線を非表示にする
$table->setGrid(0);

// 囲み線を非表示にする
$table->SetBorder(2);

// フォントを設定
$table->SetRowFont(4,FF_ARIAL,FS_BOLD,11);
$table->SetRowFont(0,FF_ARIAL,FS_BOLD,11);
$table->SetFont(1,2,1,3,FF_ARIAL,FS_BOLD,11);

// グリッド線の設定
$table->SetRowGrid(4,2,'black',TGRID_SINGLE);
$table->SetColGrid(1,1,'black',TGRID_SINGLE);
$table->SetRowGrid(1,1,'black',TGRID_SINGLE);

// 色の設定
$table->SetFillColor(0,1,0,6,'black');
$table->SetRowColor(0,'white');
$table->SetRowFillColor(4,'lightgray@0.3');
$table->SetFillColor(2,0,2,6,'lightgray@0.6');
$table->SetFillColor(1,2,1,3,'lightred');

// テーブルをグラフ・エリアに追加
$graph->Add($table);

// グラフをブラウザに返す
$graph->Stroke();

?>

```

例 12.4.2_5

GROUP 10		w631	w632	w633	w634	w635	w636
		High (sum)	Critical (sum)				
Sum:	Low (sum)	34	13	17	15	8	3
88	41	43	35	26	20	22	16
95	49	43	49	45	51	47	9
90	73	45	49	51	51	47	72
76	72	47	47	47	47	47	72

```
<?php // content="text/plain; charset=utf-8"
require_once 'jpgraph/jpgraph.php';
require_once 'jpgraph/jpgraph_canvas.php';
require_once 'jpgraph/jpgraph_table.php';

// 新規のキャンバスグラフをグラフ・エリアとして設定
$graph = new CanvasGraph(630,600);

// 基本テーブルのデータを設定
$data = array(
    array('GROUP 10',      'w631','w632','w633','w
634','w635','w636'),
    array('Critical (sum)',13,17,15,8,3,9),
    array('High (sum)',34,35,26,20,22,16),
    array('Low (sum)',41,43,49,45,51,47),
    array('Sum:',88,95,90,73,76,72)
);

// 新規テーブルを作成しデータを格納。配置・フォントを設定
$table = new GTextTable();
$table->Set($data);
$table->SetAlign('right');
$table->SetFont(FF_TIMES,FS_NORMAL,12);
$table->SetCellFont(0,0,FF_ARIAL,FS_BOLD,16);

// テーブルを 90 度回転させる
$table->SetTextOrientation(90);
// $table->SetCellTextOrientation(0,0,0);

// ヘッダの列の背景色を設定
$table->SetColFillColor(0,'lightgray');

// デフォルトの最小の行の高さを設定
$table->SetMinRowHeight(0,150);

// テーブルをグラフ・エリアに追加
$graph->Add($table);

// グラフをブラウザに返す
$graph->Stroke();

?>
```

例 12.4.2_6

	April	May	June	July	August
2005	7	13	17	15	8
2006	7	34	35	26	20
2007	7	41	43	49	45
Sum:	21	88	95	90	73

```

<?php // content="text/plain; charset=utf-8"
require_once 'jpgraph/jpgraph.php';
require_once 'jpgraph/jpgraph_canvas.php';
require_once 'jpgraph/jpgraph_table.php';
require_once 'jpgraph/jpgraph_iconplot.php';
require_once 'jpgraph/jpgraph_flags.php';

// 新規のキャンバスグラフをグラフ・エリアとして設定
$graph = new CanvasGraph(700,300);

// 基本テーブルのデータを設定
$data = array(
    array("", 'April', 'May', 'June', 'July', 'August'),
    array("", 'Triumph', 'Triumph', 'Triumph', 'Triumph', 'Triumph'),
    array('2005', 7, 13, 17, 15, 8),
    array('2006', 7, 34, 35, 26, 20),
    array('2007', 7, 41, 43, 49, 45),
    array('Sum:', 21, 88, 95, 90, 73)
);

// 新規テーブルを作成しデータを格納。フォントを設定
$table = new GTextTable();
$table->Set($data);
$table->SetFont(FF_TIMES,FS_NORMAL,11);

// デフォルトのテーブルの水平・垂直の配置を設定
$table->SetAlign('left','top');

// ヘッダ行の設定
$table->SetRowFont(0,FF_ARIAL,FS_BOLD,12);
$table->SetRowColor(0,'white');
$table->SetRowAlign(0,'center');
$table->SetRowFillColor(0,'darkorange');

// テキストの色の設定
$table->SetRowFont(1,FF_ARIAL,FS_BOLD,12);
$table->SetRowColor(1,'white');

// 合計行のフォーマットを設定
$table->SetRowFont(5,FF_ARIAL,FS_BOLD,12);

// グリッド線を表示する
$table->SetGrid(1);

// 最上行と最左列の背景色を設定
// $table->SetRowFillColor(4,'lightgray@0.5');
// $table->SetColFillColor(0,'lightgray@0.5');
// $table->SetFillColor(0,0,4,0,'lightgray@0.5');

// 列2の最小列幅をコンテンツに関係なく 100 ピクセルに設定
$table->SetMinColWidth(2,100);

// 行1にバイクの画像を追加
$table->SetRowAlign(1,'center','top');
for( $i=1; $i <= 5; ++$i ) {
    $table->SetCellImage(1,$i,"tr$i.jpg");
    $table->SetCellImageConstrain(1,$i,TIMG_HEIGHT,80);
}

// テーブルをグラフ・エリアに追加
$graph->Add($table);

// グラフをブラウザに返す
$graph->Stroke();

?>

```

12.5 マトリックス・グラフ

このモジュールはエンタープライズ版のライブラリにのみ含まれています。

マトリックス・グラフは四角形のマトリックスのコンテンツを視覚化するために使います。マトリックスの各セルの入力値はそれぞれの色でマッピングされ、入力するマトリックスのサイズに適合した四角形のプロットに表示されます。入力用マトリックスに対応する各モジュールのサイズはカスタマイズできます。マトリックスの構成要素であるモジュールには円形と四角形の 2 種類あります。

円形のモジュールで高品質のマトリックスにするためには、ユーザ設定のオプショナルのスーパー・サンプリングを活用して色のついた円の強みを生かしたアンチ・エイリアス効果を達成します。

各入力値に対する色の割当ては選択したカラー・マップによってコントロールされます。ライブラリはまずマトリックス上で最小値と最大値を確立し、最小値はカラーマップ上、一番下の色、最大値は一番上の色に合わせます。その間に存在するすべての値はリニア補間によって、一定間隔でその値に応じた色が割り当てられます。カラー・マップは、事前に設定されたものと、ユーザが自由に設定できるものがいくつか装備されています。

凡例(追加するかどうかは選択可能)は対応しているプロットの色の範囲を表示しています。

Tips: ライブラリはメッシュ補間をサポートしています。メッシュ補間そのものは新たなデータを生成するものではありませんが、マトリックス上の元の入力値の間に「仮の」補間値を作成してスムーズなプロットを作成するのに役立ちます。これは MATLAB™ 関数の 2 次元データ補間に使われる `interp2` に似ています。

マトリックス・グラフの全体的な構造と性能は、風配図などライブラリの他のグラフと同様です。グラフ・エリアには複数のプロット、タイトル、フッタ、テキスト、アイコン、背景画像等を追加することができます。図 12.5 は中程度に複雑な 2 つのマトリックス・プロットをマトリックス・グラフ・エリアに追加した例です。背景画像やアイコン(左上のロゴ)、グラフ・エリアのトリミング、フリー・ポジショニングによるテキスト配置などの機能を使っています。ここではプロット上にアルファ・ブレンディングも若干取り入れ、背景に輝きが少し出るようにしました。これら 2 つのプロットは同じサンプル・データによるものですが、異なるカラー・マップを使っています。

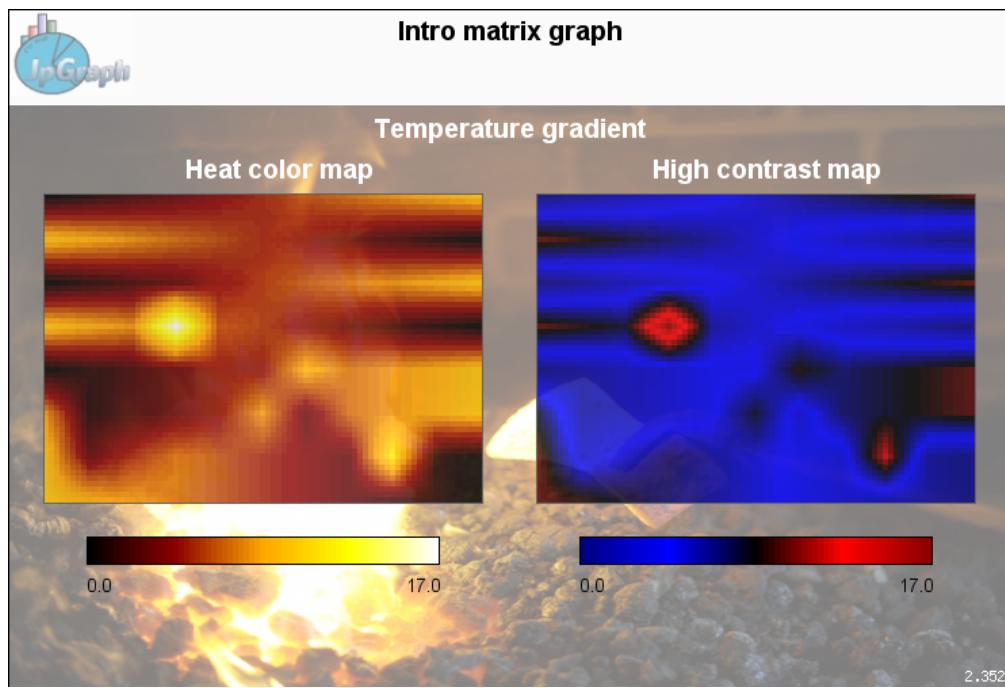


図 12.5 マトリックス・プロットの性能を一部反映した中程度に複雑なグラフ

ライブラリには、プロットに使用する色を自在に調整してビジュアル的にフォーマットする広範囲の機能を装備しています。カラー・マップは、ビルトイン・タイプとプロットのコントラスト調整(スケール・レンジの調整)などマニュアル指定ができるタイプがあります。

12.4.2 マトリックス・プロットの機能

- 手動と自動のどちらでも設定できるスケール・レンジ
- カスタマイズできる凡例の位置とレイアウト
- 22 のビルトイン・カラー・マップを装備」
- カラー・マップを任意で手動指定可能
- カラー・コントラスト調整可能
- 一つのグラフ・エリア上で複数のプロットを位置設定するレイアウト・クラス
- 円形・四角形2つのモジュール・タイプ
- 入力データをメッシュ補間
- マトリックス・プロットのサイズ設定は自在。手動、自動、あるいは両方の組合せ
- マトリックス・プロットのアルファ・ブレンディングによるサポート
- 背景画像、テキスト・オブジェクト、アイコン等すべて他のグラフ同様のサポート

12.4.3 基本的なマトリックス・グラフの作成とフォーマット

マトリックス・プロットのを作成するときは、拡張モジュール「jpgraph_matrix.php」をコア・モジュールである「jpgraph.php」と共にスクリプトに含める必要があります。

それ以外はマトリックス・グラフの作成は、1つ以上のマトリックス・プロットのキャンバスとなるマトリックス・グラフ・エリアを作成するときの、ライブラリの通常の手順に従います。原則的な手順は次の通りです。

1. MatrixGraph クラスのインスタンスとして基本マトリックス・グラフ・エリアを作成する。
(JpGraph のすべての例で、このインスタンスを使う場合、\$graph 変数を使用しています。)
2. `P align="left" class="style1">>2.` 1つ以上のマトリックス・プロットを MatrixPlot クラスのインスタンスとして作成し、カラー・マップや体裁に必要な変更を加え、マトリックス・グラフ・キャンバスに追加する。
3. MatrixGraph::Stroke()メソッドで、グラフをブラウザに送る。他のグラフ同様、このメソッドは、ブラウザ(クライアント)に返すこと、ファイル名を第 1 引数として指定してファイルにグラフを書き込むことも可能。

図 12.5.1 はすべてのパラメタに初期設定値を使用したマトリックス・プロットを表示したものです。

```
(File: matrix_ex0.php)
<?php // content="text/plain; charset=utf-8"
require_once('jpgraph/jpgraph.php');
require_once('jpgraph/jpgraph_matrix.php');

// ランダムなマトリックス
$data = array(
    array(0,1,2,3,4,5,6,7,8,9,10),
    array(10,9,8,7,6,5,4,3,2,1,0),
    array(0,1,2,3,4,5,6,7,8,9,10),
    array(10,9,8,17,6,5,4,3,2,1,0),
    array(0,1,2,3,4,4,9,7,8,9,10),
    array(8,1,2,3,4,8,3,7,8,9,10),
    array(10,3,5,7,6,5,4,3,12,1,0),
    array(10,9,8,7,6,5,4,3,2,1,0),
);
```

```
// 基本マトリックスグラフ・エリアを作成し、タイトルを
設定。
$graph = new MatrixGraph(400,300);
$graph->title->Set('Basic matrix example');
$graph->title->SetFont(FF_ARIAL,FS_BOLD,14);

// 初期設定値のみを使用したマトリックス・プロットを作成。
$mp = new MatrixPlot($data);
$graph->Add($mp);

$graph->Stroke();

?>
```

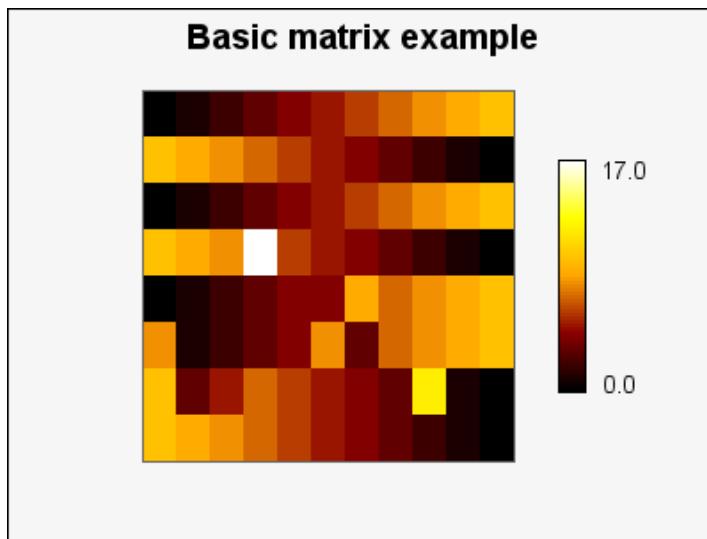


図 12.5.1 初期設定値の基本マトリックス・グラフ

他のタイプのグラフ同様、マトリックス・プロットも単一でも複数でも同じマトリックス・グラフ・エリアに自由に配置できます。位置設定は座標の絶対値でも、グラフ・エリアの幅・高さに対する比率でもどちらでも指定ができます。

Tips: マトリックス・プロットを簡単に位置設定するにはレイアウト・クラスを使います。

Tips: マトリックスのデータがファイルに存在するなら、ユーティリティクラスの `ReadFileData` を使い、次のメソッドでファイル内のデータを取り込むのが便利でしょう。

```
ReadFileData::FromMatrix($aFile,$aSeparator='')
```

これは、ファイルからマトリックスを読み込みます。マトリックスの各行は個別のラインで、各セルは第2引数として指定された文字で区別されます。初期設定によりスペースはセパレータとして認識されます。読み返される値はすべて浮動小数点(倍精度)に変換されます。次の例はこの手法の使い方が非常に簡単であることを示しています。

```
$data = ReadFileData::FromMatrix('matrixdata.txt');
```

タイトルやフッタの追加などのグラフ・フォーマットの標準オプションについては、他のグラフと同じですでの他の章をご参照ください。

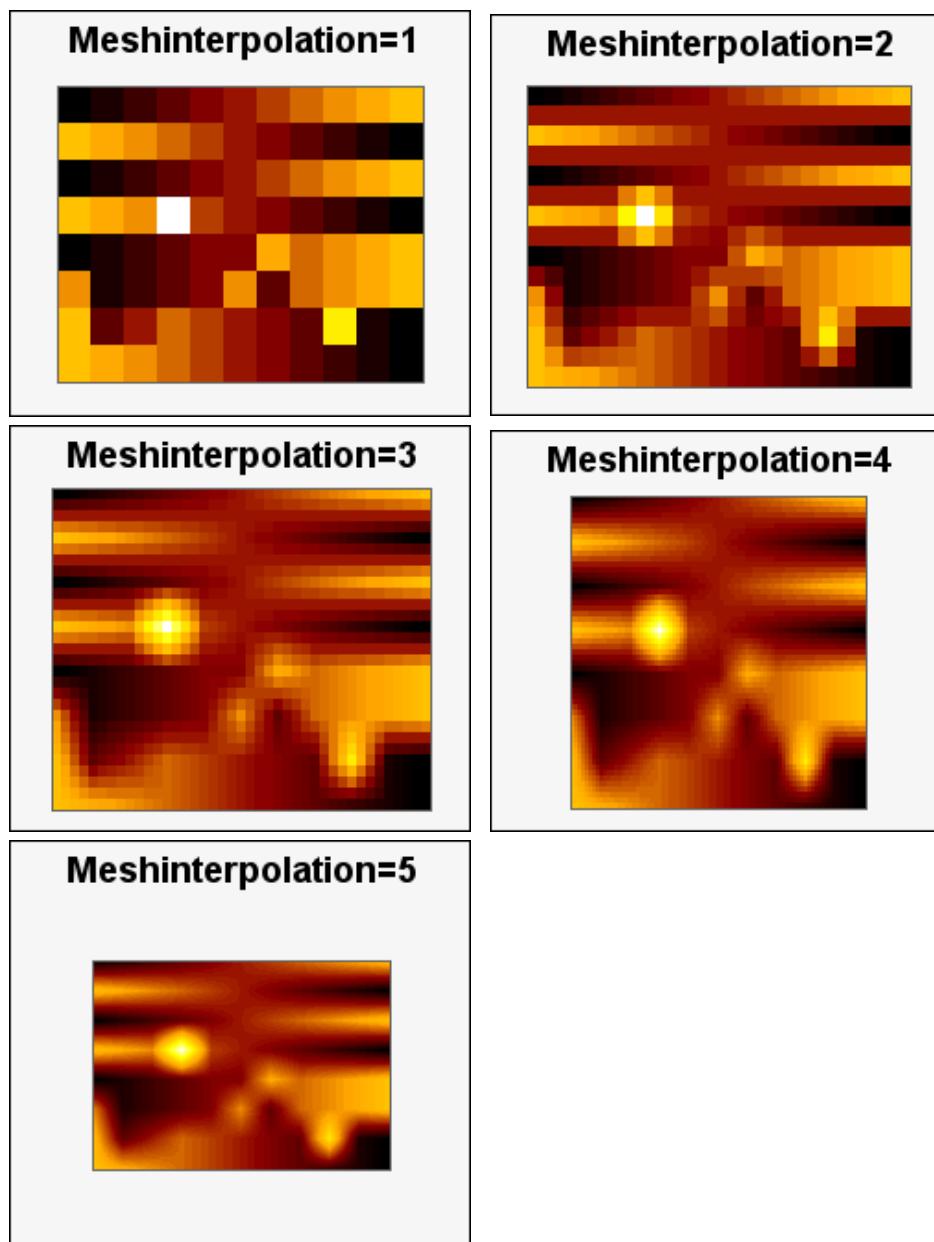
12.4.4 入力データのメッシュ補間

メッシュ補間を利用するとマトリックス・プロットの体裁がスムーズに見えるようにできます。これは元のマトリックスの入力データ間にリニア補間した中間値を生成するためです。

補間ファクターでは、補間再現の頻度を指定する必要があります。通常は2から6の範囲になります。6より大きい値を指定することは可能ですが、補間ファクター上で補間回数が激増することになります。また、この補間はマトリックスにすでに存在する情報以上のものを生成することはな

いことにご留意ください。このようなデータのリニア補間ではマトリックスの基礎データが完全に有効であることを検証することが必要です。

次の図は元データ(補間頻度=1)を1~5回補間した場合の効果を表した例です。ここで選択しているグラフ・サイズでは、5回の補間によってグラフ・エリアの制約によりモジュールが 1×1 ピクセルになってしまったので、これ以上の補間は意味がありません。(8x11の元データを5回補間すると 113×161 のマトリックスになっています。)



プロットのサイズが異なるのは、マトリックスの各セルに整数のピクセル数を格納する必要があるためです。上のグラフでは最大のモジュール・サイズを使いましたが画像内に収まっています。つまり、見栄えが異なるわけです。

この補間を行うには2通りの方法があります。

- 補間ファクターをプロット関数の第2引数として指定してマトリックス・プロットを作成する場合は次のように記述します。

```
$matrixplot = new MatrixPlot($data,4); // 補間4回
```

- 複数のプロットで同じデータを共有する場合には、新規マトリックス・プロット・オブジェクトを作成するたびに毎回設定するより、最初に補間の設定を一度行うほうが効率的です。これはユーティリティ関数を使って次のように記述します。

- doMeshInterpolate(&\$aData, \$aFactor)

宣言からわかるように、これは元のデータが指定された回数補間された新規データに置換されるリファレンス・メソッドです。この方法により、大きいマトリックスを複数コピーして不要なデータを作ることが防げます。

注意: MATLAB™関数に慣れているユーザは interp2()によく似たメッシュ補間があることに気付かれるでしょう。

12.4.5 モジュール・タイプを変更する

初期設定によりマトリックスの個々のセルに割当てられるモジュール・タイプは四角形です。この章の始めに説明したように円形のモジュールも使うことができます。タイプの選択は次のメソッドで行えます。

- MatrixPlot::SetModuleType(\$aType)
\$aType, 0 = 四角形を使用, 1 = 円形を使用

円形のモジュールを使うときはプロットの背景色も別途指定したほうがいいかもしれません。円と円の隙間から背景が見えるからです。背景色は次のメソッドで指定します。

- MatrixPlot::SetBackgroundColor(\$aColor)

円形モジュールの使用例です。

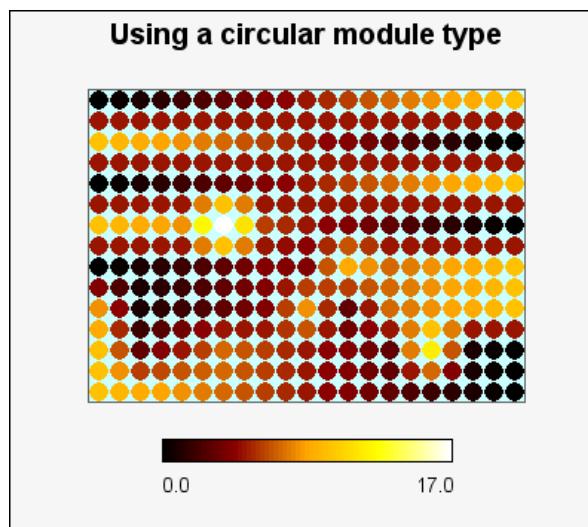


図 12.5.4 円形モジュール・タイプの使用例

エンタープライズ版スタンダードで作成できるグラフ：マトリックス・グラフ

```
<?php // content="text/plain; charset=utf-8"
require_once('../jpgraph.php');
require_once('../jpgraph_matrix.php');

$data = array(
    array(0,1,2,3,4,5,6,7,8,9,10),
    array(10,9,8,7,6,5,4,3,2,1,0),
    array(0,1,2,3,4,5,6,7,8,9,10),
    array(10,9,8,17,6,5,4,3,2,1,0),
    array(0,1,2,3,4,4,9,7,8,9,10),
    array(8,1,2,3,4,8,3,7,8,9,10),
    array(10,3,5,7,6,5,4,3,12,1,0),
    array(10,9,8,7,6,5,4,3,2,1,0),
);
$width=400; $height=350;

$graph = new MatrixGraph($width,$height);
$graph->title->Set('Using a circular module type');
$graph->title->SetFont(FF_ARIAL,FS_BOLD,14);

$mp = new MatrixPlot($data,2);
$mp->SetSize(0.85);
$mp->SetModuleType(1);
$mp->SetBackgroundColor('teal:1.8');
$mp->SetCenterPos(0.5,0.45);
$mp->SetLegendLayout(1);

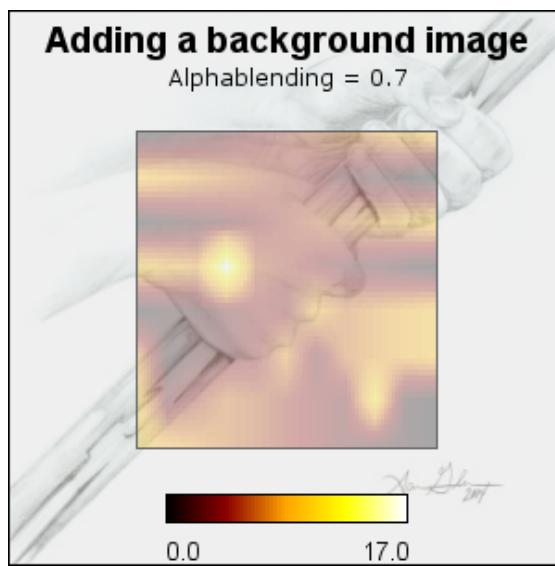
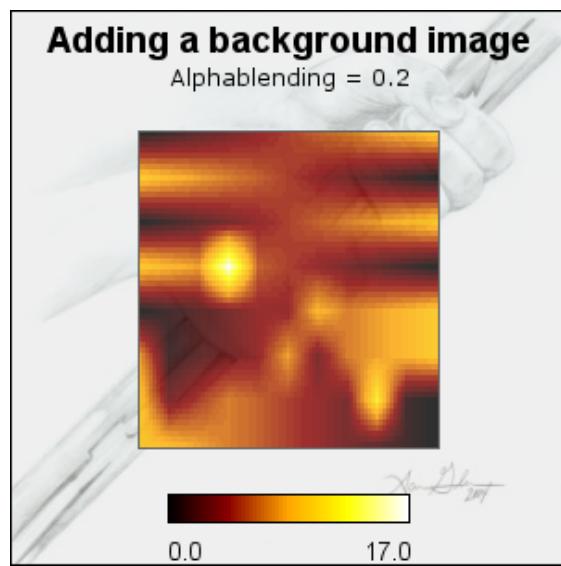
$graph->Add($mp);
$graph->Stroke();

?>
```

12.4.6 プロットのアルファ・ブレンディング

初期設定ではプロットは選択したカラー・マップのソリッド・カラー（べた塗り色）で色づけされます。アルファ値を指定することによって背景の輝きをマトリックス・プロットに反映させることができます。アルファ・ブレンディングは次のメソッドで選択できます。

- MatrixPlot::SetAlpha(\$aAlpha)
\$aAlpha , 0=無透明, 1=透明（背景だけが表示されるため実用的ではない。）



ご覧のとおりプロット・エリアだけが調整されます。凡例は常に透明化されることなく表示されます。

それぞれのソースコードは次のようにになります。

図 12.5.5_1 アルファ・ブレンディング=0.2 のマトリックスのソースコード

```
<?php // content="text/plain; charset=utf-8"
require_once('../jpgraph.php');
require_once('../jpgraph_matrix.php');
require_once('../jpgraph_iconplot.php');

$data = array(
    array(0,1,2,3,4,5,6,7,8,9,10),
    array(10,9,8,7,6,5,4,3,2,1,0),
    array(0,1,2,3,4,5,6,7,8,9,10),
    array(10,9,8,17,6,5,4,3,2,1,0),
    array(0,1,2,3,4,4,9,7,8,9,10),
    array(8,1,2,3,4,8,3,7,8,9,10),
    array(10,3,5,7,6,5,4,3,12,1,0),
    array(10,9,8,7,6,5,4,3,2,1,0),
);

// Do the meshinterpolation once for the data
doMeshInterpolate($data,4);
$r=count($data);$c=count($data[0]);

$width=300; $height=300;
$graph = new MatrixGraph($width,$height);
$graph->title->Set('Adding a background image');
$graph->title->SetFont(FF_ARIAL,FS_BOLD,14);
$graph->subtitle->Set('Alphablending = 0.2');

// Add a stretched background image
$graph-
>SetBackgroundImage('ironrod.jpg',BGIMG_FILLFRAME);
$graph->SetBackgroundImageMix(50);

$mp = new MatrixPlot($data,1);
$mp->SetSize(0.65);
$mp->SetCenterPos(0.5,0.5);
$mp->SetLegendLayout(1);
$mp->SetAlpha(0.2);

$graph->Add($mp);
$graph->Stroke();

?>
```

図 12.5.5_2 アルファ・ブレンディング=0.7 のマトリックスのソースコード

```
<?php // content="text/plain; charset=utf-8"
require_once('../jpgraph.php');
require_once('../jpgraph_matrix.php');
require_once('../jpgraph_iconplot.php');

$data = array(
    array(0,1,2,3,4,5,6,7,8,9,10),
    array(10,9,8,7,6,5,4,3,2,1,0),
    array(0,1,2,3,4,5,6,7,8,9,10),
    array(10,9,8,17,6,5,4,3,2,1,0),
    array(0,1,2,3,4,4,9,7,8,9,10),
    array(8,1,2,3,4,8,3,7,8,9,10),
    array(10,3,5,7,6,5,4,3,12,1,0),
    array(10,9,8,7,6,5,4,3,2,1,0),
);

// Do the meshinterpolation once for the data
doMeshInterpolate($data,4);
$r=count($data);$c=count($data[0]);

$width=300; $height=300;
$graph = new MatrixGraph($width,$height);
$graph->title->Set('Adding a background image');
$graph->title->SetFont(FF_ARIAL,FS_BOLD,14);
$graph->subtitle->Set('Alphablending = 0.7');

// Add a stretched background image
$graph-
>SetBackgroundImage('ironrod.jpg',BGIMG_FILLFRAME);
$graph->SetBackgroundImageMix(50);

$mp = new MatrixPlot($data,1);
$mp->SetSize(0.65);
$mp->SetCenterPos(0.5,0.5);
$mp->SetLegendLayout(1);
$mp->SetAlpha(0.7);

$graph->Add($mp);
$graph->Stroke();

?>
```

JpGraph の設定を変更する

定義、デフォルト値	解説
"TTF_DIR", "/usr/local/fon_ts.ttf"	JpGraph で使用する TTF フォントを格納するディレクトリを指定します。"/" が終わりに必須です。格納するフォントは JpGraph が読み込めるような命名規則にしたがっている必要があります。
"CSIMCACHE_DIR", "csimcache"	CSIM を使用したグラフのキャッシュを保存するディレクトリです。"/" が終わりに必須です。キャッシュを使用する設定の CSIM グラフがここにキャッシュされます。ディレクトリは PHP が見ることができるようなファイルシステム名でなければなりません。また、http バージョンは htdocs ディレクトリと関連する HTTP サーバーが見ることができるものを見除いて同じディレクトリであるべきです。相対パスが指定された場合、イメージ スクリプトが実行された場所からの相対パスとなります。注意：デフォルトの設定では、イメージ スクリプトが実行されたディレクトリにサブディレクトリを作り、すべてのファイルをそこに保存します。もちろん、このディレクトリも PHP のプロセスによる書き込みが可能である必要があります。
"CSIMCACHE_HTTP_DIR", "csimcache"	HTTP 形式の CSIM を使用したグラフのキャッシュを保存するディレクトリです。"/" が終わりに必須です。キャッシュを使用する設定の CSIM グラフがここにキャッシュされます。ディレクトリは PHP が見ることができるようなファイルシステム名でなければなりません。また、http バージョンは htdocs ディレクトリと関連する HTTP サーバーが見ることができるものを見除いて同じディレクトリであるべきです。相対パスが指定された場合、イメージ スクリプトが実行された場所からの相対パスとなります。注意：デフォルトの設定では、イメージ スクリプトが実行されたディレクトリにサブディレクトリを作り、すべてのファイルをそこに保存します。もちろん、このディレクトリも PHP のプロセスによる書き込みが可能である必要があります。
"CHINESE_TTF_FONT", "bkai00mp.ttf"	FF_CHINESE aka FF_BIG5 で使用される TTF ファイルのファイル名です。これは、フォント ファミリーが FF_CHINESE、または FF_BIG5 のどちらかで指定される場合に使用されます。
"LANGUAGE_CYRILLIC", false	キリル言語をサポートします。
"CYRILLIC_FROM_WINDOWS", false	この設定を true に設定した場合、入力テキストは windows 1251 であると想定されて変換されます。false に設定した場合は koi8-r と仮定されます。
'MINCHO_TTF_FONT', 'ipam.ttf'	FF_MINCHO で使用される日本語の TrueType フォント名を設定します。
'PMINCHO_TTF_FONT', 'ipamp.ttf'	FF_PMINCHO で使用される日本語の TrueType フォント名を設定します。
'GOTHIC_TTF_FONT', 'ipag.ttf'	FF_GOTHIC で使用される日本語の TrueType フォント名を設定します。
'PGOTHIC_TTF_FONT', 'ipagp.ttf'	FF_PGOthic で使用される日本語の TrueType フォント名を設定します。

"INSTALL_PHP_ERR_HA NDLER", false	ライブラリがデフォルトのエラーハンドラにグラフィカルなエラーメッセージを生成させるかどうかを設定する。画像が期待されるページでエラーが起きた際、赤い×印が表示される代わりにエラー メッセージを画像で表示するようになるので、開発中はとても便利な設定です。
"CATCH_PHPERRMSG", true	ライブラリが global php_errmsg 文字列を調査し、いくつかのエラーをグラフィカルに表示するようにします。たとえば、ヘッダー ファイルが見つからないためグラフが作成されず、赤い×印だけが表示されてしまう場合にこれはとても便利です。稼動中のサイトではこれは false に設定しておいてください。
'USE_TRUECOLOR',true	truecolor を使用するかどうかを設定します。注意 1.GD 2.0.2 以上の場のみ使用することができます。注意 2.Win32 の GD 2.0.1 + PHP 4.0.6 の環境で truecolor を使用しようとするとクラッシュします。まだ GD2.x が安定しているとは(特に Win32 では)考えられないので、Truecolor サポートは、アルファ版だと考えてください。注意 3.背景画像の使用は GD2 を有効にしなければなりません。注意4:もし有効にしていたら、truetype フォントはとても見た目が悪くなるでしょう。⇒それらのバグが GD 2.0.1 で修正されるまで、同じ画像による背景画像と truetype フォントを同時に表示させることはできません。
"USE_CACHE",false	キャッシュファイルを使用するかどうかを設定します。これを false に設定すると、キャッシュディレクトリにファイルが作成されないようになります。READ_CACHE を false に設定しても、キャッシュファイルは作成されています。USE_CACHE=false を設定することで、キャッシュファイルは生成されないようにになります。
"READ_CACHE",true	画像の生成前にキャッシュファイルを探すかどうかを設定します。これを false に設定すると、画像を再生成した際にキャッシュの読み込みをせずに画像を作成するようになります。キャッシュの読み込みをしないように設定していても、画像が生成されるたびにキャッシュファイルは更新されることに注意してください。"USE_CACHE" も設定するようにしてください。
"DEFAULT_GFORMAT", "auto"	これを "auto" に設定すると、画像の形式は png gif jpg の中から最適なものが選択されます。(サポートされる形式は、あなたの PHP の設定に依存します。)
"USE_IMAGE_ERROR_H ANDLER",true	エラーハンドラが画像ベースかテキストベースかを設定します。画像ベースの場合、スクリプトは常に画像形式でエラーを返すのでとても作業がしやすくなります。
"USE_APPROX_COLORS ,true	もしカラーパレットがいっぱいだったら、似ている色は近似値を使用してまとめるようにします。もし画像背景やグラデーションを使う予定なら、これを使用するのは良い考えかもしれません。もしそうでなければ、カラーパレットが使い尽くされたとエラーが出るでしょう。近似値を使用する際の欠点は、指定した通りの色が表示されないかもしれないということです。注意1:色数の制限がないので、これは、true color イメージではなくパレットイメージを適用させるだけです。
"ERR_DEPRECATED",fal se	反対する関数の使用とパラメーターは、致命的なエラーを与えるべきでしょうか?(コードが今後の証拠であるかどうかチェックするために、役に立ちます。)
"BRAND_TIMING",false	各画像を作るのにかかる時間を、それぞれ作成された画像の左下に表示させます。これはグラフを作る際の性能の測定に役に立ちます。
"BRAND_TIME_FORMA T","Generated in: %01.3fs"	"BRAND_TIMING" で表示される、画像の作成時間の表示フォーマットを設定します。

以下の定義は、多くの場合変更する必要はありません。

定義、デフォルト値	解説
"CACHE_FILE_GROUP", "wwwadmin"	キャッシュされたファイルがどのグループに属するかを設定します。("" に設定すると、デフォルトグループの "PHP-user" が与えられます) Apache ユーザーは指定されたグループのメンバーでなくてはなりません。そうでなければ指定されたグループを Apache が設定することが不可能になります。
"CACHE_FILE_MOD", 664	キャッシュされたファイルのパーミッションを設定します。("" を設定すると、 "PHP-user" のデフォルトのパーミッションが与えられます。)
"USE_BRESENHAM", false	Bresenham 円のアルゴリズムを使用するか、組み込まれている Arc() 関数を使用するかを決定します。Bresenham はより綺麗な円を描くことができますが、CPU に高い負荷をかけ、GD の Arc() 関数よりも処理が遅くなります。速度を重視するために、デフォルトでは false に設定されています。
"_CSIM_SPECIALFILE", "_csim_special_"	GetHTMLCSIM() 関数に内部的に使用される Graph::Stroke() の呼び出しにおいて、イメージマップを計算だけを行うということを示す特別なファイル名です。
"_CSIM_DISPLAY", "_jp_g_csimd"	イメージマップを使用する際、完全な CSIM HTML ページではなく、ただ画像の生成だけを行わせるということをスクリプトに知らせるための HTTP GET 引数です。