SVGコンポーネント

概要

- 4D SVGコンポーネント
- 開発ツール
- シンタックス詳細

4D SVGコンポーネント

SVG (Scalable Vector Graphics) とは、XMLに基づいた、2次元のベクターグラフィックファイルです。4Dには、SVGファイルを表示するのに使用可能な統合されたレンダリングエンジンが含まれています。

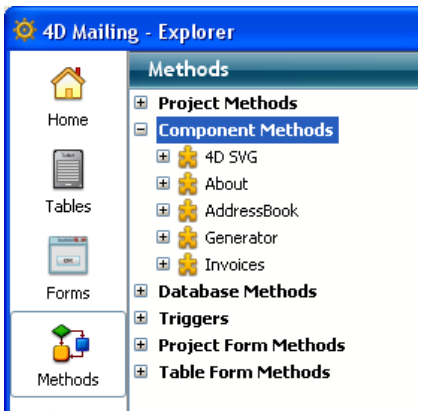
SVGピクチャーを操作するのに使用されるXMLランゲージは、特に豊富で広範囲に及びます。SVGの導入を容易にするために、4Dでは一般的なグラフィックオブジェクトを作成・操作するための数々のコマンドを備えたSVGコンポーネントを用意しています。このライブラリーは、難解なものというよりは、多くの4Dデベロッパの要求に応えるためのものです。また、4D XMLコマンドを使用することで、さらに複雑な処理も行う事ができます。

インストールと実装

4D SVG コンポーネントは、4D v11 SQL リリース3(バージョン11.3)以降にインストールすることができます。ホストデータベースはUnicodeモードで実行されている必要があります (ASCII 互換モードではこのコンポーネントを使用することができません)。

他の4Dコンポーネント同様、4D SVGコンポーネントはコンポーネントフォルダ(4D SVG.4dbase)をComponentフォルダへとコピーすることによってインストールできます。データベースのComponentフォルダは、ス トラクチャーファイルと同階層に置かれている必要があります。コンポーネントはスタートアップ時に読み込まれるため、全ての要素をコピーし終わる前にデー タベースを開いてはいけません。

コンポーネントが正常にインストールされていれば、データベースのメソッドタブの中の、"コンポーネントメソッド"セクション内に"4D SVG"という要素が表示されているはずです。



この要素を展開すると、全てのコンポーネントコマンドを閲覧する事ができます。これらのコマンドは、通常の4Dランゲージや、標準のプラグインコマンドと同様に4Dメソッドエディター内で使用することができます。

4D SVG コンポーネントを使用することによって、コマンドの選択とSVGコードレンダリングのための追加のウィンドウが表示されるようになるという利点があります。この点についての詳細は**開発ツール**セクションを参照して下さい。

SVG エフェクトと Direct2D (Windows)

4D v13以降、Windows ではデフォルトで Direct2D グラフィックレンダリングエンジンが使用されています。ハードウェアとソフトウェアの設定によっては、このエンジンを使用することによって影などの一部のSVGエフェクトのレンダリングが変化してしまうことがあります。このような場合には、**SET DATABASE PARAMETER** コマンドを使用して Direct2D を無効化することができます。

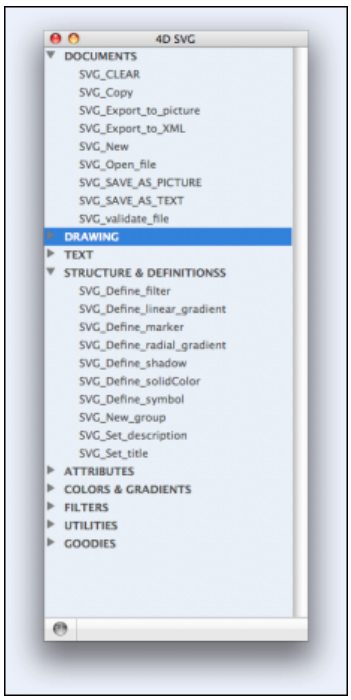
開発ツール

4D SVG コンポーネントには、コードの入力とSVGグラフィックのプレビューを容易にするためのツールセットが用意されています:

- シンタックスパレット
- カラーパレット
- SVG ビューアー

シンタックスパレット

シンタックスパレットは、4D SVG コンポーネントコマンドをテーマごとにグループ分けして表示します。



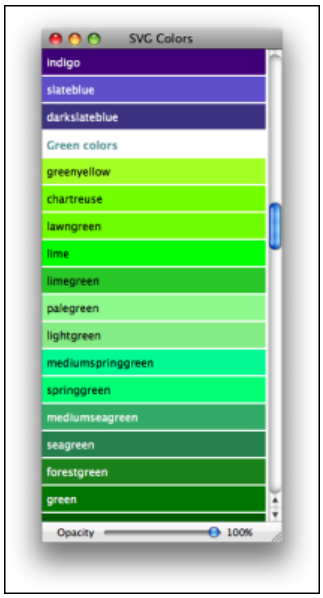
ドラッグ&ドロップによって、パレットからコンポーネントコマンドをメソッドエディター内に挿入することができます。コマンドはその引数とともにペーストされます。任意の引数についてはその頭に下線がついた状態でペーストされます。

シンタックスパレットを表示するには:

- **SVGTool_Display_syntax** メソッドを実行する
- 4D Pop を使用している場合、4D Pop コンポーネントパレット内の**SVG** ボタンをクリックし、**SVG Component syntax** コマンドを選択します(以下を参照して下さい)

カラーパレット

カラーパレットはSVGスタンダードで指定されているそれぞれのカラーの名前とサンプルを表示するとともに、不透明度を調整するためのスライダーも表示します:

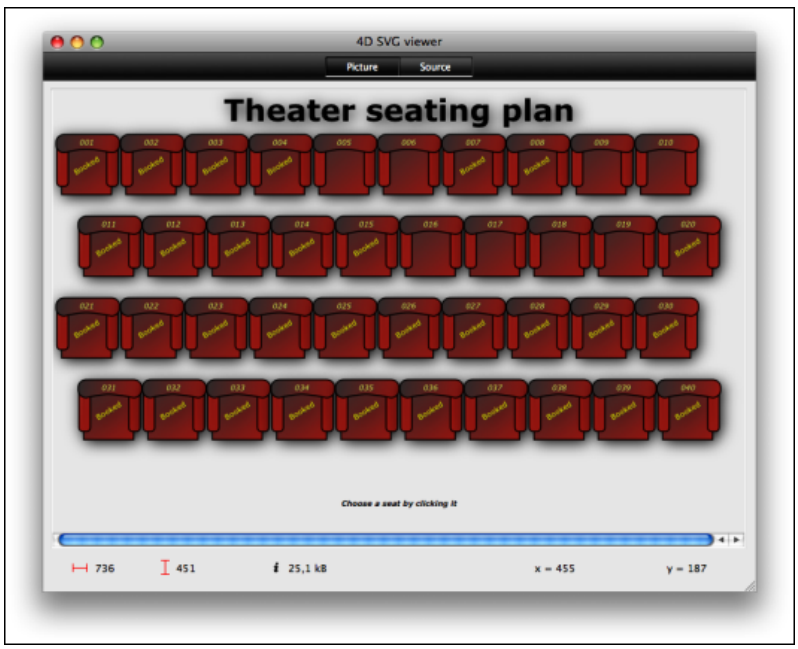


このパレットからドラッグ&ドロップによって4Dメソッドエディター内にカラー参照を挿入することができます。カラーは文字列として挿入され、不透明度の指定があった場合にはそれを文字列内に含みます(例えば lavenda のカラーで不透明度が30%の場合には"lavender:30"となります)。カラー参照についての詳細な情報については セクションを参照して下さい。
カラーは4Dフォームエディターにもドラッグ&ドロップが可能です。ドロップすると、フォーム内にそのカラーの正方形の静的なSVGピクチャーを作成します。

カラーパレットを表示するためには、**SVGTool_Display_colors** メソッドを実行して下さい。

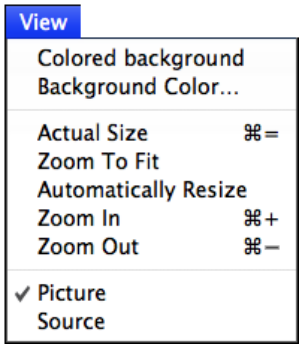
SVG Viewer

4D SVG には、開発フェーズにおいてきわめて便利なSVG Viewerが用意されています。



Viewer には、**ピクチャー**ボタンと**ソース**ボタン、または**表示**メニューからアクセス可能な、二つのページがあります。

- **ピクチャー**: このページでは、SVGピクチャーファイルを開いたり(**ファイル**メニュー内)、ドラッグ&ドロップしたりすることのできる、表示エリアが用意されています。ここでは*SVGTool_SHOW_IN_VIEWER* コマンドを使用して有効なSVG 参照を表示することもできます。
- **ソース**: このページでは、ピクチャーに関連付けられたXMLコードを表示します。コードを選択・コピーすることはできますが、編集はここではできません。
  ウィンドウが前面にあるときには、**表示**メニューを使用して複数の表示オプションを変更したり、ピクチャーファイルをディスクに保存したりすることができます。



注: "ピクチャー"ページでは、標準のコンテキストメニューを使用することができます。

Viewerウィンドウは、以下の方法のいずれかによって表示することができます:

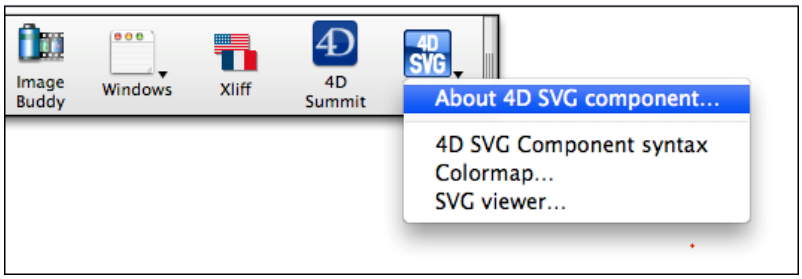
- **SVGTool_Display_viewer** メソッドを実行する。この場合、ウィンドウは空の状態で表示されます。
- *SVGTool_SHOW_IN_VIEWER* メソッドに、有効なSVG参照を渡して、ピクチャー参照のプレビューを見る(詳細はコマンドのページを参照して下さい)。
- 4D Pop コンポーネントを使用している場合、SVGボタンをクリックしてその中のSVG Viewerコマンドを選択する(以下を参照して下さい)。

4D Popへの統合

4D Popは、デベロッパーの利便性のためにコンポーネントをまとめたもので、4D開発環境に統合可能なツールバーの中にグループ分けされています。

4D Pop と 4D SVGを同時に使用すると、新しいボタンが4D Popツールバーに追加されます:

このボタンを使用すると、4D SVGアシスタントツールへとダイレクトにアクセスすることができます:



注: 4DPop は4Dフルインストーラーの追加ツールの中に含まれています。

シンタックス詳細

SVG_Ref

4D SVGコンポーネントコマンドのほとんどは、SVGストラクチャーを **SVG_Ref** 型の参照として操作します。

SVG_Ref とは、16文字の文字列型の4D式で、メモリーに読み込まれたSVGストラクチャーを固有に識別します。このSVGストラクチャーには*SVG_Copy*、 *SVG_New*、 *SVG_Open_picture*、 または *SVG_Open_file* コマンドなどを使用して読み込まれたSVGドキュメントに加え、プログラムによって管理されているあらゆるSVGストラクチャー(オブジェクト、フィルター、パス、等)も含まれます。

SVG_Ref はXML参照です。SVG_Ref 参照は4D XML DOM コマンドに対して *elementRef* 引数として使用することができます。

SVG_Ref は必要がなくなったら、*SVG_CLEAR* コマンドにSVG_Ref を渡して使用し、メモリーを解放するのを忘れないようにしてください。

任意の引数

特に記載がない限り、任意の数字型の引数は渡された値が-1のときには無視され、テキスト型の引数は空の文字列が渡されたときには無視されます。

座標

特に記載がない限り、カレントのユーザー座標システムにおいて、位置 (*x, y*) とサイズ (*width*, *height*, *radius*) 引数が必要になります。

カラー&グラデーション

- SVG カラー
- SVG_Color_from_index
- SVG_Color_grey
- SVG_Color_RGB_from_CMYK
- SVG_Color_RGB_from_HLS
- SVG_Color_RGB_from_long
- SVG_FADE_TO_GREY_SCALE
- SVG_Filter_ColorMatrix
- SVG_GET_COLORS_ARRAY
- SVG_GET_DEFAULT_BRUSHES
- SVG_Get_named_color_value
- SVG_SET_BRIGHTNESS
- SVG_SET_DEFAULT_BRUSHES
- SVG_SET_HUE
- SVG_SET_SATURATION

SVG カラー

カラーの定義について

SVGはCSS2仕様で定義されているカラーの代理シンタックスを認識する事ができます。4D SVGコンポーネントのコマンドはこれらのシンタックスを全てサポートしています。

カラーは、以下のどちらかの形式によって表現することができます:

- RGB フォーマット

フォーマット	例
#rgb	#f00
#rrggbb	#ff0000
rgb(r,g,b)	rgb(255,0,0)
	rgb(100%, 0%, 0%)

- "Color" キーワードフォーマット
  SVG はたくさんのカラー名(例えば"red"など)のキーワード名を受け付けることができます。

キーワードの一覧と、それに対応するRGBの値は、 にまとめてあります。この一覧を表示して、4D SVGカラーパレット経由で直接カラー値を表示・挿入させることもできます。この点についてのより詳細な情報に関しては、 の章を参照して下さい。

不透明度

コンポーネントコマンドのカラー式において、不透明度を指定することができます。"color:opacity"というシンタックスを使用して指定します。このときopacityには0(無色)から100(完全に不透過のカラー)の値を渡す事ができます。つまり"red:50"という値は透過度50%の赤色、と解釈されます。

グラデーション

グラデーションとは、ヴェクターを用いた先進的なカラーの遷移です。これらのグラデーションは*SVG_Define_linear_gradient* コマンドと*SVG_Define_radial_gradient* コマンドを使用して設定することができます。設定がなされた後は、そのグラデーションは"url(#GradientName)"シンタックスを用いて使用する事ができます。

同様に、*SVG_Define_solidColor* コマンドを用いて透過度もカスタムのカラーを設定することができます。

SVG_Color_from_index

SVG_Color_from_index (index) -> 戻り値

引数	型		説明
index	倍長整数	➡	カラー番号
戻り値	テキストフィールド	⮌	indexで指定されたカラー

説明

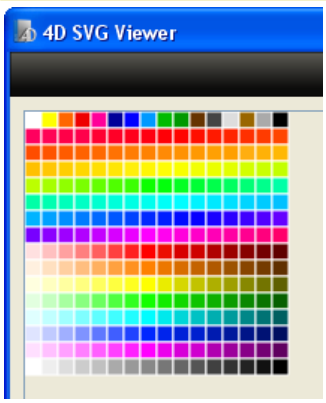
SVG_Color_from_indexは*index*引数で指定した4Dカラーに対応するSVGカラーを返します。

index 引数には4Dカラーパレット中の番号を渡します。4Dカラーは1~256の番号が付けられています。このポイントについては4Dの**OBJECT SET COLOR**コマンドの説明を参照してください。

例題

この例題では4DカラーパレットをSVGで作成します:

```
$Dom_svg:=SVG_New
$Lon_line:=0
For($Lon_ii;0;15;1)
   $Lon_column:=0
   For($Lon_i;1;16;1)
      $Txt_color:=SVG_Color_from_index(($Lon_ii*16)+$Lon_i)
      $Dom_rect:=SVG_New_rect($Dom_svg;$Lon_column;
      $Lon_line;11;11;0;0;"white";$Txt_color)
      $Lon_column:=$Lon_column+11
   End for
   $Lon_line:=$Lon_line+11
End for
SVGTool_SHOW_IN_VIEWER($Dom_svg)
```



SVG_Color_grey

SVG_Color_grey (percentage) -> 戻り値

引数	型		説明
percentage	整数	⇒	灰色の濃度
戻り値	文字	⇐	カラーを表す文字列

説明

SVG_Color_grey コマンドは、*percentage* 引数で指定した濃度を持つ灰色を表す文字列を返します。文字列は"RGB(red, green, blue)"形式で返され、3つの値は同じ値になります。このシンタックスはSVGレンダリングエンジンによって認識可能なものです。

例題

```
$txtColor:=SVG_Color_grey(60)
 `$txtColor は "rgb2102,102,102)"
```

SVG_Color_RGB_from_CMYK

SVG_Color_RGB_from_CMYK (cyan ; magenta ; yellow ; black {; format}) -> 戻り値

引数	型		説明
cyan	倍長整数	⇒	シアン値
magenta	倍長整数	⇒	マゼンタ値
yellow	倍長整数	⇒	イエロー値
black	倍長整数	⇒	ブラック値
format	倍長整数	⇒	カラーフォーマット
戻り値	テキスト	⇐	カラー文字列

説明

SVG_Color_RGB_from_CMYKコマンドは4つのカラー引数*cyan*, *magenta*, *yellow* そして *black*に対応するカラーを表す文字列を返します。返される文字列はデフォルトで、SVG描画エンジンが認識するシンタックス"RGB(r,g,b)"形式です。

cyan, *magenta*, *yellow* そして *black* は倍長整数で0 ~ 100%の値を指定します。

オプションの *format* 引数を使用して返されるカラー文字列のフォーマットを指定できます。使用可能なフォーマットは以下の通りです:

値	フォーマット
1 (デフォルト)	rgb(r,g,b)
2	#rgb
3	#rrggbb
4	rgb(r%, g%, b%)

SVG_Color_RGB_from_HLS

SVG_Color_RGB_from_HLS (hue ; luminosity ; saturation {; format}) -> 戻り値

引数	型		説明
hue	倍長整数	⇒	色相値
luminosity	倍長整数	⇒	明度値
saturation	倍長整数	⇒	彩度値
format	倍長整数	⇒	カラーフォーマット
戻り値	テキスト	⇐	カラー文字列

説明

SVG_Color_RGB_from_HLSコマンドは*hue*, *luminosity* そして *saturation*引数に対応する、カラーを表す文字列を返します。返される文字列はデフォルトで、SVG描画エンジンが認識するシンタックス"RGB(r,g,b)"形式です。

hue は倍長整数で0 ~ 360°の値を指定します。

luminosity と *saturation* 倍長整数で0 ~ 100%の値を指定します。

オプションの *format* 引数を使用して返されるカラー文字列のフォーマットを指定できます。使用可能なフォーマットは以下の通りです:

値	フォーマット
1 (デフォルト)	rgb(r,g,b)
2	#rgb
3	#rrggbb
4	rgb(r%, g%, b%)

SVG_Color_RGB_from_long

SVG_Color_RGB_from_long (color {; format}) -> 戻り値

引数	型		説明
color	倍長整数	➡	カラー値
format	整数	➡	カラーのフォーマット
戻り値	文字	⮌	カラーを表す文字列

説明

SVG_Color_RGB_from_long コマンドは、*color* 引数に渡された色を表す文字列として返します。文字列はSVGレンダリングエンジンによって解釈可能な"RGB(red, green, blue)"形式で返されます。

color 引数は4バイトの倍長整数で、その形式(0x00RRGGBB)は以下のところに詳細があります(バイトは右から左へと0から3の番号が振られます):

バイト	詳細
2	カラーの赤成分(0..255)
1	カラーの緑成分(0..255)
0	カラーの青成分(0..255)

任意の*format* 引数を使用すると、返されるカラー文字列のフォーマットを指定することができます。使用できる値は以下の通りです:

値	形式
1 (デフォルト)	rgb(r,g,b)
2	#rgb
3	#rrggbb
4	rgb(r%, g%, b%)

例題

```
$txtColor:=SVG_Color_RGB_from_long($color)
 `$colorが16744448(オレンジ)のとき、$txtColor は "rgb (255,128,0)"
```

SVG_FADE_TO_GREY_SCALE

SVG_FADE_TO_GREY_SCALE (svgObject {; value})

引数	型		説明
svgObject	SVG_Ref	⇒	SVGオブジェクトの参照
value	実数	⇒	グレー値

説明

SVG_FADE_TO_GREY_SCALE コマンドは、*svgObject* 引数で参照しているSVG画像のグレースケールを変換するフィルターを適用します。

任意の*value* 引数には、適用したいグレースケール値を渡す事ができます。この引数を渡さなかった場合、明るさの視覚認知に基づいて変換が行われます(赤30%、緑59%、青11%)。

SVG_Filter_ColorMatrix

SVG_Filter_ColorMatrix (svgObject {; in ; result} {; type {; values}}) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクト参照
in	テキスト	➡	原始フィルターに対しての入力を指定
result	テキスト	➡	フィルターの出力結果の参照
type	テキスト	➡	行列演算の型を指定
values	テキスト	➡	変換行列に適用する数値
戻り値	SVG_Ref	↩	新しいカラー値のSVGオブジェクトに対する参照

説明

SVG_Filter_ColorMatrix コマンドは、 *svgObject* 引数に渡したソース画像の各ピクセルに行列変換を行い、新しいカラー値を持った結果を生成します。

in 引数には、その前の'result' の値と同じ文字列を渡すか、以下の6つのキーワードのどれかを渡す事ができます:

- *SourceGraphic*: フィルターを参照するターゲット要素(画像、図形、グループ等)を指定します。このキーワードはフィルター要素へのオリジナルの入力だったグラフィック要素を表します。
- *SourceAlpha*: *SourceGraphic* の下にあるキャンバスを指定します。このキーワードはフィルター要素へのオリジナルの入力だったグラフィック要素を表します。
- *BackgroundImage*: *SourceGraphic* の下にあるキャンバスを指定します。このキーワードは、フィルター要素が呼び出された時点の、フィルターリージョンの下のキャンバスのスナップショットを表します。
- *BackgroundAlpha*: *SourceGraphic*. Same の下にあるキャンバスのアルファチャンネルを指定します。アルファチャンネルが使用されるという点以外は *BackgroundImage* と同じです。
- *FillPaint*: ターゲット要素のfillプロパティで塗りつぶされたフィルターリージョンと等しいサイズをもつ疑似グラフィックを指定します。このキーワードは、フィルター効果のためのターゲット要素の'fill'プロパティの値を表します。
- *StrokePaint*: ターゲット要素のfillプロパティで塗りつぶされたフィルターリージョンと等しいサイズをもつ疑似グラフィックを指定します。このキーワードは、フィルター効果のためのターゲット要素の'stroke'プロパティの値を表します。

何の値も渡さず、またこれが最初の原始フィルターであった場合は、*SourceGraphic* が入力として使用されます。何の値も渡さず、これが二つ目以降の原始フィルターであった場合は、このフィルターは前の原始フィルターの結果を入力として使用します。

result 引数には、フィルターの出力結果の参照を渡します。この出力は、コマンド内の同じ 'filter '要素内においてその後に使用する *in* 引数で参照する事ができます。何の値も渡さなかった場合、この出力は、次の原始フィルターが *in* 引数に何の値も渡されなかった場合にそのフィルターの入力としてのみ再使用することができます。

type 引数には、以下のキーワードのどれかを渡して行列変換の型を指定する事ができます:

- *saturate*: RGBカラーの彩度を、*values* 引数に渡された0から1の実数値を使用して調整します。
- *hueRotate*: 全てのRGBカラーチャンネルの色相行列を、*values* 引数にて指定された角度(度単位)だけ回転させます。
- *luminanceToAlpha*: 赤、緑、青のチャンネルを輝度値へと変換します。RGBチャンネルは黒(0,0,0)に設定されます。
- *matrix*: *values* 引数に渡された値の一覧を使用してカラーを設定します。既存のカラーとアルファチャンネルの組み合わせで出力でのそれぞれのチャンネルの値を指定します。

type 引数を渡さなかった場合、デフォルトで、値に *matrix* が指定されたのと同じ効果になります。

values 引数には、*type* 引数に渡したキーワードに基づいた数値を渡します。:

- "*matrix*" キーワードの場合: 20の行列値のリストを渡して下さい。値は空白もしくはカンマで区切られます。
- "*saturate*" キーワードの場合: 0から1の間の単一の実数値を渡します。仕様で許可されている値は0-1の間ですが、多数のブラウザではオーバーさちゅレーションのために1を超える値を受け付けています。
- "*hueRotate*" キーワードの場合: (回転の角度を表す)単一の実数値を渡します。
- "*luminanceToAlpha*" キーワードの場合: 数値は渡しません。この型では *values* 引数は使用されず、アルファチャンネルは破棄され、入力の輝度に等しい値で置き換えられます。

values 引数を渡さなかった場合、デフォルトの挙動は *type* 引数に渡したキーワードによって異なります:

- "*matrix*" キーワードの場合: デフォルトで単位行列の値が使用されます。

- "*saturate*" キーワードの場合: デフォルトで値は1になります(変化なし)。
- "*hueRotate*" キーワードの場合: デフォルトでは値は0になります(変化なし)。
- "*luminanceToAlpha*" キーワードの場合: デフォルトで、この値は使用されません。

注: Windows 環境下では、このコマンドを使用するためにはその前にDirect2Dを無効化しておく必要があります(**SET DATABASE PARAMETER**コマンド内の詳細の Direct2D disabled 定数を参照して下さい)。

例題


```
C_TEXT($Dom_filter;$Dom_node;$Dom_rect;$Dom_svg;$Txt_matrix)

SVG_SET_OPTIONS(SVG_Get_options?+5)

$Dom_svg:=SVG_New

$Dom_filter:=SVG_Define_filter($Dom_svg;"Matrix")
$Txt_matrix:=\
".33 .33 .33 0 0 "\
+".33 .33 .33 0 0 "\
+".33 .33 .33 0 0 "\
+".33 .33 .33 0 0"


$Dom_node:=SVG_Filter_ColorMatrix($Dom_filter;"SourceGraphic";"";"matrix";$Txt_matrix)

$Dom_filter:=SVG_Define_filter($Dom_svg;"Saturate")
$Dom_node:=SVG_Filter_ColorMatrix($Dom_filter;"SourceGraphic";"";"saturate";"1.5")
 // この値を渡すための別のシンタックス
 //$Dom_node:=SVG_Filter_ColorMatrix
($Dom_filter;"SourceGraphic";"";"saturate";String(1,5;"&xml"))

$Dom_filter:=SVG_Define_filter($Dom_svg;"HueRotate90")
$Dom_node:=SVG_Filter_ColorMatrix($Dom_filter;"SourceGraphic";"";"hueRotate";"90")

$Dom_filter:=SVG_Define_filter($Dom_svg;"LuminanceToAlpha")
$Dom_node:=SVG_Filter_ColorMatrix($Dom_filter;"SourceGraphic";"";"luminanceToAlpha")

$Dom_rect:=SVG_New_rect($Dom_svg;2;0;797;100;0;0;"none";"coral")

$Dom_rect:=SVG_New_rect($Dom_svg;2;100;797;100;0;0;"none";"coral")
SVG_SET_FILTER($Dom_rect;"Matrix")

$Dom_rect:=SVG_New_rect($Dom_svg;2;200;797;100;0;0;"none";"coral")
SVG_SET_FILTER($Dom_rect;"Saturate")

$Dom_rect:=SVG_New_rect($Dom_svg;2;300;797;100;0;0;"none";"coral")
SVG_SET_FILTER($Dom_rect;"HueRotate90")

$Dom_rect:=SVG_New_rect($Dom_svg;2;400;797;100;0;0;"none";"coral")
SVG_SET_FILTER($Dom_rect;"LuminanceToAlpha")

SVG_New_text($Dom_svg;"No filter";110;10;"Verdana";60;Bold;-1;"black")
SVG_New_text($Dom_svg;"Matrix";110;110;"Verdana";60;Bold;-1;"black")
```

```
SVG_New_text($Dom_svg;"Saturate";110;210;"Verdana";60;Bold;-1;"black")
SVG_New_text($Dom_svg;"HueRotate";110;310;"Verdana";60;Bold;-1;"black")
SVG_New_text($Dom_svg;"Luminance";110;410;"Verdana";60;Bold;-1;"black")

 //結果を見る
SVGTool_SHOW_IN_VIEWER($Dom_svg)

 //SVG_SAVE_AS_TEXT($Dom_svg;System folder(Desktop)+"export.svg")

 //最後にメモリをクリアすることもお忘れなく
SVG_CLEAR($Dom_svg)
```

SVG_GET_COLORS_ARRAY

SVG_GET_COLORS_ARRAY (colorNamesArrayPointer)

引数	型		説明
colorNamesArrayPointer	ポインター	⇒	カラー名を受け取る配列へのポインター

説明

SVG_GET_COLORS_ARRAYコマンドはSVGによって認識されるカラー名を、 *colorNamesArrayPointer* 引数が指し示す配列に返します。

参照: http://www.w3.org/TR/SVG/types.html#ColorKeywords

SVG_GET_DEFAULT_BRUSHES

SVG_GET_DEFAULT_BRUSHES (line {; background})

引数	型		説明
line	ポインター	➡	文字変数
background	ポインター	➡	文字変数

説明

SVG_GET_DEFAULT_BRUSHES コマンドは、*line* 引数で指定した変数の中に、描画ラインのカレントのデフォルトカラーを返します。

任意の*background* 引数が渡された場合、背景で使用されるカレントのデフォルトカラーがこの引数で指定した変数に格納されます。

何の変更も行われていない場合、これらのカラーはそれぞれ、黒と白です。

例題

SVG_SET_DEFAULT_BRUSHES コマンドを参照して下さい。

SVG_Get_named_color_value

SVG_Get_named_color_value (colorName {; rgbComponent}) -> 戻り値

引数	型		説明
colorName	テキスト	➡	SVGカラー名
rgbComponent	テキスト	➡	カラーコンポーネントを指定する"R"、"G" または "B"
戻り値	倍長整数	⮌	カラーの値を返す

説明

SVG_Get_named_color_value コマンドは、*colorName* 引数で指定したSVGカラーの値を返します。

任意の*rgbComponent* 引数には、"R" (red)、"G" (green) または "B" (blue)を渡すことによって値を取得したい特定のカラーコンポーネントを指定する事ができます。この引数を渡さない場合、コマンドは(完全な)長いカラー値を返します。

SVG_SET_BRIGHTNESS

SVG_SET_BRIGHTNESS (svgObject ; brightness {; brightness2 ; brightness3})

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクト参照
brightness	実数	➡	オブジェクト全体、または赤コンポーネントの明度(0と1の間の値で暗く、1を超える値で明るく)
brightness2	実数	➡	緑コンポーネントの明度の値
brightness3	実数	➡	青コンポーネントの明度の値

説明

SVG_SET_BRIGHTNESS コマンドは、*svgObject* 引数に参照が渡されたSVG画像またはコンテナの明度を設定します。

brightness 引数には、明度を暗くするためには0から1の間の値を渡し、明度を明るくするためには1を超える値を渡します。単一の *brightness* 引数を渡した場合は、明度の要素はオブジェクト全体に適用されます。

また、二つの(任意の)引数 *brightness2* と *brightness3* を渡す事により、それぞれの明度の値が個別のカラーコンポーネントへと適用されます。つまり、*brightness* は"R" (赤)パートへと適用され、 *brightness2* は "G" (緑)パートへと適用され、そして *brightness3* は "B" (青)ポートへと適用されます。

SVG_SET_DEFAULT_BRUSHES

SVG_SET_DEFAULT_BRUSHES (line {; background})

引数	型		説明
line	文字	➡	カラー
background	文字	➡	カラー

説明

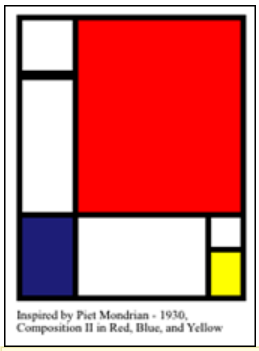
SVG_SET_DEFAULT_BRUSHES コマンドを使用すると、コンポーネントによって使用されるデフォルトのカラーを設定することができます。

line 引数には線において使用されることになる新しいカラーを渡します。任意の*background* 引数には背景の描画に使用される新しいカラーを渡します。

どちらの引数に対しても、空の文字列を渡す事によって、コンポーネントのデフォルトの値をリセットすることができます。つまり、線のカラーには黒が、背景のカラーには白が適用されます。

例題

モンドリアン風の画像を作成する場合を考えます




```
$svg:=SVG_New
 `デフォルトのカラーを設定
SVG_SET_DEFAULT_BRUSHES("black";"white")
 `線の太さを4-pointに設定
SVG_SET_STROKE_WIDTH($svg;4)
$g:=SVG_New_group($svg)
SVG_New_rect($g;2;2;40;40)
SVG_New_rect($g;2;45;40;100)
SVG_SET_FILL_BRUSH(SVG_New_rect($g;2;144;40;60);"midnightblue")
SVG_SET_FILL_BRUSH(SVG_New_rect($g;42;2;120;142);"red")
SVG_New_rect($g;42;144;95;60)
SVG_New_rect($g;137;144;25;25)
SVG_SET_FILL_BRUSH(SVG_New_rect($g;137;169;25;35);"yellow")
SVG_SET_TRANSFORM_TRANSLATE($g;10;10)
 `注釈
SVG_New_text($svg;"Inspired by Piet Mondrian - 1930,\rComposition II in Red, Blue, and
Yellow";10;220;"";9)
```

SVG_SET_HUE

SVG_SET_HUE (svgObject ; hue)

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクト参照
hue	倍長整数	➡	色相

説明

SVG_SET_HUE メソッドは、*svgObject* 引数で指定したSVGオブジェクトに色相値を設定します。*svgObject* はSVGコンテナー (svg, group, symbol, pattern, marker等) または画像でなければなりません。そうでない場合、エラーが生成されます。

hue 引数には0 ~ 360の値を渡すことができます。

SVG_SET_SATURATION

SVG_SET_SATURATION (svgObject ; saturation)

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクト参照
saturation	倍長整数	➡	彩度値

説明

SVG_SET_SATURATION メソッドは、*svgObject* 引数で指定したSVGオブジェクトに彩度値を設定します。*svgObject* はSVGコンテナー (svg, group, symbol, pattern, marker等) または画像でなければなりません。そうでない場合、エラーが生成されます。

saturation 引数には0 ~ 100の値を渡すことができます。

ストラクチャー & 定義

- SVG_Define_clip_path
- SVG_Define_filter
- SVG_Define_linear_gradient
- SVG_Define_marker
- SVG_Define_pattern
- SVG_Define_radial_gradient
- SVG_Define_shadow
- SVG_Define_solidColor
- SVG_Define_style
- SVG_DEFINE_STYLE_WITH_ARRAYS
- SVG_Define_symbol
- SVG_DELETE_OBJECT
- SVG_Get_default_encoding
- SVG_New_group
- SVG_SET_DEFAULT_ENCODING
- SVG_Set_description
- SVG_SET_PATTERN_CONTENT_UNITS
- SVG_SET_PATTERN_UNITS
- SVG_Set_title

SVG_Define_clip_path

SVG_Define_clip_path (parentSVGObject ; clipPathID) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	⇒	親要素の参照
clipPathID	テキスト	⇒	クリップパスの名前
戻り値	SVG_Ref	⇐	クリップパスの参照

説明

SVG_Define_clip_pathを使用して、*parentSVGObject* で指定したSVGコンテナー中で新しいクリップパスを定義し、その参照を返します。*parentSVGObject* がSVGドキュメントでなく、あるいはSVGドキュメントに属していない場合、エラーが生成されます。

clipPathID 引数はクリップパスの名前です。この名前はクリップパスとオブジェクトを関連付けるために使用さっれます。ドキュメント中に既に同じ名前の要素が存在する場合、エラーが生成されます。

参照: http://www.w3.org/TR/2001/REC-SVG-20010904/masking.html#EstablishingANewClippingPath

例題

SVG_SET_CLIP_PATHコマンドの例題を参照してください。

SVG_Define_filter

SVG_Define_filter (parentSVGObject ; id {; frameX ; frameY {; frameWidth ; frameHeight {; frameUnit ; filterUnit}}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
id	文字	➡	シンボル名
frameX	倍長整数	➡	X軸の座標
frameY	倍長整数	➡	Y軸の座標
frameWidth	倍長整数	➡	ターゲットの四角の幅
frameHeight	倍長整数	➡	ターゲットの四角の高さ
frameUnit	文字	➡	フレームの座標系
filterUnit	文字	➡	値のフィルター系
戻り値	SVG_Ref	↩	フィルターの参照

説明

SVG_Define_filter コマンドは*parentSVGObject* 引数で参照しているSVGコンテナ内に新しいフィルターを設定し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

フィルターはターゲット要素へと適用されるグラフィックオペレーションの継承です。フィルター要素自身は直接レンダリングされることはありません。フィルターは*SVG_SET_FILTER* コマンドによってオブジェクトへと適用されます。

id 引数はマーカーの名前を指定します。この名前はフィルターをオブジェクトと関連付けるのに使用されます。その名前が既存の要素で使用されている場合、その要素は上書きされます。

任意の*frameX*、*frameY*、*frameWidth* そして *frameHeight* 引数はドキュメント内にてこのフィルターが適用される長方形のリージョンを設定します。

任意の*frameUnit* 引数はその4つの引数に対しての座標系を設定します。使用可能な値は"*userSpaceOnUse*" または "*objectBoundingBox*" (デフォルト値)です。

任意の*filterUnit* 引数は長さとフィルター定義プロパティ用の座標系を設定します。使用可能な値は"*userSpaceOnUse*" (デフォルト値) または "*objectBoundingBox*"です。

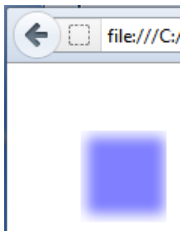
例題

ここでは、以下のようなオペレーションを実行したい場合を考えます:

- 50%の青の背景を持つ長方形を作成します
- 4%のぼかしフィルターを作成しこの長方形に適用します
- 結果をディスク上のSVGファイルに保存します

```
$Dom_SVG:=SVG_New

 //50%の青の背景を持つ長方形を作成
$Dom_rect:=SVG_New_rect($Dom_SVG;50;50;50;50;0;0;"blue:50";"blue:50")

 //4%のぼかしフィルターを作成
$Dom_filter:=SVG_Define_filter($Dom_SVG;"blur")
SVG_Filter_Blur($Dom_filter;4)
SVG_Filter_Offset($Dom_filter;4)

 //フィルターを長方形に適用
SVG_SET_FILTER($Dom_rect;"blur")

 //結果をSVGファイルに保存
SVG_SAVE_AS_TEXT($Dom_SVG;System folder(Desktop)+"test.svg")

SVG_CLEAR($Dom_SVG)
```

結果:



SVG_Define_linear_gradient

SVG_Define_linear_gradient (parentSVGObject ; id ; startColor ; endColor {; rotation {; spreadMethod {; x1 ; y1 ; x2 ; y2}{; startColorOffset ; endColorOffset}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	⇒	親要素の参照
id	文字	⇒	グラデーション名
startColor	文字	⇒	始めのカラー
endColor	文字	⇒	終わりのカラー
rotation	整数	⇒	グラデーションベクトルの回転
spreadMethod	テキスト	⇒	グラデーションのスプレッドメソッド(pad、reflect、repeat)
x1	実数	⇒	グラデーションベクターのx1座標(省略時0)
y1	実数	⇒	グラデーションベクターのy1座標(省略時1)
x2	実数	⇒	グラデーションベクターのx2座標(省略時0)
y2	実数	⇒	グラデーションベクターのy2座標(省略時1)
startColorOffset	実数	⇒	始めのカラーのオフセットのパーセンテージ値
endColorOffset	実数	⇒	終わりのカラーのオフセットのパーセンテージ値
戻り値	文字	⇐	グラデーションの参照

説明

SVG_Define_linear_gradient コマンドは*parentSVGObject* 引数で指定されたSVGコンテナ内にリニアなグラデーションを設定し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

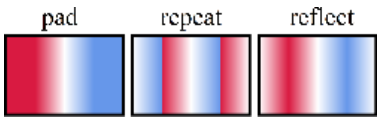
グラデーションは、一つのカラーからもう一つのカラーへのベクターに沿った無段階な変化で構成されています。指定をすると、グラデーションは与えられたグラフィック要素上に呼び出され、そのグラデーションで要素を塗るか縁どるかを指定することもできます。

id 引数はグラデーション名を指定します。同じ名前の要素が既にある場合、それは上書きされます。このグラデーションを呼び出す名前は、"url(#ID)"のシンタックスを使用してカラーの指定をする際にグラデーションを呼び出す名前です。

startColor と *endColor* 引数はグラデーションの始めと終わりのカラーを指定するために使います。

任意の*rotation* 引数はグラデーションのベクターの方向と位置を設定します(例題を参照して下さい)。
任意の*spreadMethod* 引数は、グラデーションが*parentSVGObject*の淵から始まる、もしくは終わる場合の塗りについて定義します。この引数には以下の文字列のどれか一つを値として渡す事ができます:

- "pad": グラデーションの終わりのカラーを使用して残りのエリアを塗りつぶします。
- "reflect": グラデーションパターンを、オブジェクトが埋まるまで始まり→終わり、終わり→始まり、始まり→終わり、、、と反射し続けます。
- "repeat": グラデーションパターンを、オブジェクトが埋まるまで始まり→終わり、始まり→終わり、、、と繰り返し続けます。



この引数が省略された場合、"pad"の値のエフェクトが使用されます。

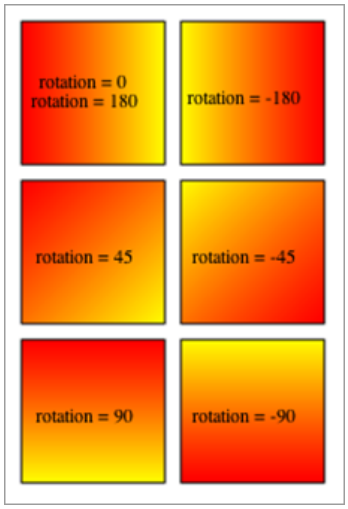
任意の*x1*、*y1*、*x2*、*y2* 引数はグラデーションのベクターを定義します。このベクターはレンダリングエンジンが使用する開始点と終着点を指定します。これらの引数には、パーセンテージを小数であらわした値(0...1)を渡します。

v14以降、任意の*startColorOffset* と *endColorOffset* 引数を渡す事によってそれぞれ始めのカラーと終わりのカラーのオフセットのパーセンテージ値を定義できるようになりました。パーセンテージを指定するためには、1未満の実数値か、0から100までのパーセンテージを表す値を渡す事ができます。例えば、"0.1"または"10"はどちらも10%として解釈されます。

負の値を渡した場合、*startColorOffset* 引数においては0%として、*endColorOffset* 引数においては100%として認識されます。100を超える値を渡した場合、それは100%として解釈されます。

例題 1

6つの四角を描画し、それぞれに回転と方向が異なるリニアなグラデーションを適用する場合を考えます:




```
$svg:=SVG_New

SVG_Define_linear_gradient($svg;"demoGradient_1";"red";"yellow")
SVG_New_rect($svg;10;10;90;90;0;0;"black";"url(#demoGradient_1)")
SVG_New_text($svg;"rotation = 0\rrotation = 180";50;40;"";-1;-1;Align center)

SVG_Define_linear_gradient($svg;"demoGradient_2";"red";"yellow";-180)
SVG_New_rect($svg;110;10;90;90;0;0;"black";"url(#demoGradient_2)")
SVG_New_text($svg;"rotation = -180";150;50;"";-1;-1;Align center)

SVG_Define_linear_gradient($svg;"demoGradient_3";"red";"yellow";45)
SVG_New_rect($svg;10;110;90;90;0;0;"black";"url(#demoGradient_3)")
SVG_New_text($svg;"rotation = 45";50;150;"";-1;-1;Align center)

SVG_Define_linear_gradient($svg;"demoGradient_4";"red";"yellow";-45)
SVG_New_rect($svg;110;110;90;90;0;0;"black";"url(#demoGradient_4)")
SVG_New_text($svg;"rotation = -45";150;150;"";-1;-1;Align center)

SVG_Define_linear_gradient($svg;"demoGradient_5";"red";"yellow";90)
SVG_New_rect($svg;10;210;90;90;0;0;"black";"url(#demoGradient_5)")
SVG_New_text($svg;"rotation = 90";50;250;"";-1;-1;Align center)

SVG_Define_linear_gradient($svg;"demoGradient_6";"red";"yellow";-90)
SVG_New_rect($svg;110;210;90;90;0;0;"black";"url(#demoGradient_6)")
SVG_New_text($svg;"rotation = -90";150;250;"";-1;-1;Align center)

 //ドキュメントを保存
SVG_SAVE_AS_TEXT($svg;"test.svg")
 //メモリを解放
SVG_CLEAR($svg)
```

例題 2

startColorOffset と *endColorOffset* 引数(v14からの新機能)を使用する例を考えます:

```
$Dom_node:=SVG_Define_linear_gradient($Dom_svg;"clicked";"black:50";"black:20";-90;"reflect";0;80)
```

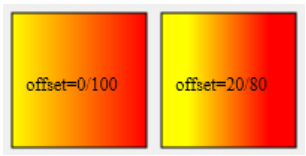
上記のコードは、以下の様に定義をします:

```
<linearGradient id="clicked spreadMethod="reflect x1="0" x2="0" y1="1" y2="0">     <stop
offset="0%" stop-color="black" stop-opacity="0.5"/>     <stop offset="80%" stop-color="black"
stop-opacity="0.2"/> </linearGradient>
```

例題 3

この例では*startColorOffset* と *endColorOffset* 引数のエフェクトについて説明します:

```
$svg:=SVG_New

SVG_Define_linear_gradient($svg;"demoGradient_1";"red";"yellow";-180;"reflect")
SVG_New_rect($svg;10;10;90;90;0;0;"black";"url(#demoGradient_1)")
SVG_New_text($svg;"offset=0/100";50;50;"";-1;-1;Align center)

SVG_Define_linear_gradient($svg;"demoGradient_2";"red";"yellow";-180;"reflect";20;80)
SVG_New_rect($svg;110;10;90;90;0;0;"black";"url(#demoGradient_2)")
SVG_New_text($svg;"offset=20/80";150;50;"";-1;-1;Align center)
 //ドキュメントを保存
SVG_SAVE_AS_TEXT($svg;"test2.svg")
 //メモリを解放
SVG_CLEAR($svg)
```



SVG_Define_marker

SVG_Define_marker (parentSVGObject ; id {; x ; y {; width ; height {; orientation}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
id	文字	➡	シンボル名
x	倍長整数	➡	参照点のX軸の座標
y	倍長整数	➡	参照点のY軸の座標
width	倍長整数	➡	マーカーの幅
height	倍長整数	➡	マーカーの高さ
orientation	倍長整数	➡	マーカーの向き
戻り値	SVG_Ref	↩	マーカーの参照

説明

SVG_Define_marker コマンドは*parentSVGObject* 引数によって指定されたSVGコンテナ内にマーカーを作成し、その参照を返します。*parentSVGObject* 引数がSVGドキュメント出なかった場合、エラーが生成されます。

マーカーオブジェクトは、矢印を描画したり、複数のマーカー(例えば曲線の点など)を描画するのに使用されます。マーカーは*SVG_SET_MARKER* コマンドを使用すると特定のSVG要素へと割り当てられます。

id 引数にはマーカー名を指定します。マーカー名はオブジェクトとマーカーを割り当てるのに使用されます。同名の要素が既に存在する場合、それは上書きされます。

任意の*x* と *y* 引数は、マーカーの位置とぴったり一致する参照点の座標を指定します。

任意の*width* と *height* 引数はレンダリングされた長方形の幅と高さを指定します。

任意の*orientation* 引数を使用すると、マーカーの向きを調節することができます。0から360の間の値を渡す事で、マーカーのX軸と、ユーザースペースの間の角度を指定します。この引数が省略された場合、または0から360の間に含まれない値を渡した場合、その位置はレンダリングエンジンによってオブジェクトに応じて自動的に計算されます。

例題

SVG_SET_MARKER コマンドの例題を参照して下さい。

SVG_Define_pattern

SVG_Define_pattern (parentSVGObject ; patternID {; width {; height {; x {; y {; unit {; viewBox}}}}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
patternID	テキスト	➡	パターンの名前
width	倍長整数	➡	パターンの幅
height	倍長整数	➡	パターンの高さ
x	倍長整数	➡	パターンのX座標
y	倍長整数	➡	パターンのY座標
unit	テキスト	➡	幅、高さ、座標の単位
viewBox	テキスト	➡	ビューボックスの四角
戻り値	SVG_Ref	⬅	パターンの参照

説明

SVG_Define_patternコマンドは、*parentSVGObject* で指定したSVGコンテナー中に新しいカスタムパターンを設定し、その参照を返します。*parentSVGObject* がSVGドキュメントではなく、またSVGドキュメントに属していない場合、エラーが生成されます。

patternID を使用してパターンの名前を指定します。この名前はパターンとオブジェクトを関連付けるために使用されます。同じ名前の要素が既に存在する場合、エラーが生成されます。

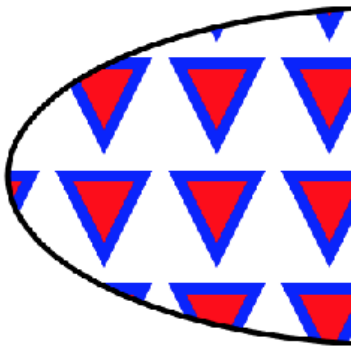
オプションの *width*, *height*, *x*, *y*, *unit* そして *viewBox* 引数を使用してパターンの四角 (パターンタイルの配置方法と間隔) を定義します。

カラー式が必要なとき、値として "url(#id)" 文字列を渡すことで、パターンは塗りつぶしや線のペイントとして割り当てられます。

参照: http://www.w3.org/TR/SVG/pservers.html#Patterns

例題 1

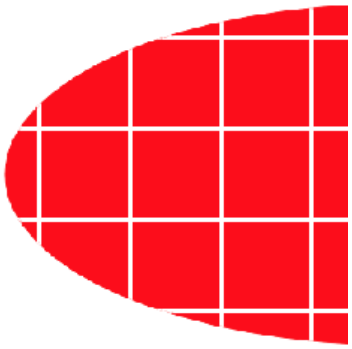
楕円を塗りつぶすためにパターンを設定する:



```
 // パターンの定義
$Dom_pattern:=SVG_Define_pattern($Dom_SVG;"MyPattern";100;100;0;0;"";"0 0 10 10")
$Dom_path:=SVG_New_path($Dom_pattern;0;0)

SVG_PATH_MOVE_TO($Dom_path;0;0)
SVG_PATH_LINE_TO($Dom_path;7;0)
SVG_PATH_LINE_TO($Dom_path;3,5;7)
SVG_PATH_CLOSE($Dom_path)
SVG_SET_FILL_BRUSH($Dom_path;"red")
SVG_SET_STROKE_BRUSH($Dom_path;"blue")

 // 楕円を描画し、パターンで塗りつぶす
$Dom_ellipse:=SVG_New_ellipse($Dom_SVG;400;200;350;150;"black";"url(#MyPattern)";5)
```

例題 2

パターンを設定し、楕円の塗りつぶしと外枠に使用する:



```
 // パターンの定義
$Dom_pattern:=SVG_Define_pattern($Dom_SVG;"MyPattern ";80;80;0;0;"";"0 0 20 20")
$Dom_rect:=SVG_New_rect($Dom_pattern;0;0;20;20;0;0;"white";"red")

 // 楕円の描画
$Dom_ellipse:=SVG_New_ellipse($Dom_SVG;400;200;350;150)

 // パターンを塗りつぶしや外枠に使用
SVG_SET_FILL_BRUSH($Dom_ellipse;"url(#MyPattern)")
SVG_SET_STROKE_BRUSH($Dom_ellipse;"url(#MyPattern)")
```

SVG_Define_radial_gradient

SVG_Define_radial_gradient (parentSVGObject ; id ; startColor ; endColor {; cx ; cy ; r {; fx ; fy}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
id	文字	➡	グラデーション名
startColor	文字	➡	始めのカラー
endColor	文字	➡	終わりのカラー
cx	整数	➡	終わりのカラーの中心のX軸の座標
cy	整数	➡	終わりのカラーの中心のY軸の座標
r	整数	➡	終わりのカラーの半径
fx	整数	➡	始めのカラーの中心のX軸の座標
fy	整数	➡	始めのカラーの中心のY軸の座標
戻り値	文字	⮌	グラデーションの参照

説明

SVG_Define_radial_gradient コマンドは、*parentSVGObject* 引数で指定したSVGコンテナ内に新しい放射型グラデーションを設定し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではなかった場合、エラーが生成されます。
グラデーションは、一つのカラーからもう一つのカラーへのベクターに沿った無段階な変化で構成されています。指定をすると、グラデーションは与えられたグラフィック要素上に呼び出され、そのグラデーションで要素を塗るか縁どるかを指定することもできます。

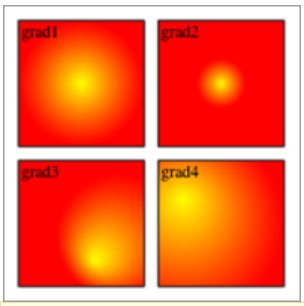
id 引数はグラデーション名を指定します。同じ名前の要素が既にある場合、それは上書きされます。このグラデーションを呼び出す名前は、"url(#ID)"のシンタックスを使用してカラーの指定をする際にグラデーションを呼び出す名前です。

startColor と *endColor* 引数はグラデーションの始めと終わりのカラーを指定するために使います。

任意の*cx*、*cy* と *r* 引数は、グラデーションの終わりのカラーの外部境界線の円を、パーセンテージ値で指定します。これらの引数に渡す値は0から100の間の値である必要があります。

任意の*fx* と *fy* 引数はグラデーションの焦点をパーセンテージ値で指定します。*startColor* 引数で指定した始めのカラーは[*fx*、*fy*]の点からスタートします。これらの値は0から100の間でなければなりません。これらの引数が省略された場合、この点は[*cx*、*cy*]と同じになります。

例題




```
$svg:=SVG_New

SVG_Define_radial_gradient($svg;"grad1";"yellow";"red")
SVG_New_rect($svg;10;10;90;90;0;0;"black";"url(#grad1)")
SVG_New_text($svg;"grad1";12;10)

SVG_Define_radial_gradient($svg;"grad2";"yellow";"red";50;50;20;50;50)
SVG_New_rect($svg;110;10;90;90;0;0;"black";"url(#grad2)")
SVG_New_text($svg;"grad2";112;10)

SVG_Define_radial_gradient($svg;"grad3";"yellow";"red";80;60;50;60;80)
SVG_New_rect($svg;10;110;90;90;0;0;"black";"url(#grad3)")
SVG_New_text($svg;"grad3";12;110)
```

```
SVG_Define_radial_gradient($svg;"grad4";"yellow";"red";20;50;80;20;30)
SVG_New_rect($svg;110;110;90;90;0;0;"black";"url(#grad4)")
SVG_New_text($svg;"grad4";112;110)

 `ドキュメントを保存
SVG_SAVE_AS_TEXT($svg;"test.svg")
 `メモリを解放
SVG_CLEAR($svg)
```

SVG_Define_shadow

SVG_Define_shadow (parentSVGObject ; id {; deviation {; offsetX {; offsetY}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
id	文字	➡	フィルター名
deviation	倍長整数	➡	影の分散の値
offsetX	倍長整数	➡	X軸のオフセット
offsetY	倍長整数	➡	Y軸のオフセット
戻り値	SVG_Ref	⮌	フィルターの参照

説明

SVG_Define_shadow コマンドは*parentSVGObject* 引数で指定されたSVGコンテナ内に新しい影フィルターを設定し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントでない場合、エラーが生成されます。

このフィルターは*SVG_SET_FILTER* コマンドを用いることによって、影を必要としているオブジェクトに対して適用されます。

引数には、フィルターの名前を指定します。この名前は、フィルターをオブジェクト関連付けるために使用される名前です。同名の要素が既に存在していた場合、それは上書きされます。

任意の*deviation* 引数は影の分散の度合いを指定します。デフォルトの値は4です。

任意の*offsetX* と *offsetY* 引数はそれぞれ、オブジェクトに対する影の水平方向と垂直方向のオフセットを指定します。デフォルトの値は4です。

例題

オブジェクトの下に影を付けるのに使用できるフィルターの宣言を考えます:


```
$svg:=SVG_New

$text:=SVG_New_text($svg;"SVG";52;76-45;"Verdana";45)
SVG_SET_FONT_COLOR($text;"red")
 `フィルターを設定
SVG_Define_shadow($svg;"myShadow")
 `テキストに対してそれを適用
SVG_SET_FILTER($text;"myShadow")
```

SVG_Define_solidColor

SVG_Define_solidColor (parentSVGObject ; id ; color {; opacity}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
id	文字	➡	カラー名
color	文字	➡	カラー式
opacity	倍長整数	➡	不透明度
戻り値	SVG_Ref	⮌	カラーの参照

説明

SVG_Define_solidColor コマンドは*parentSVGObject* 引数によって指定されたSVGコンテナ内に新しいカスタムのカラーを設定し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではなかった場合、エラーが生成されます。

id 引数はカラー名を指定します。この名前は、オブジェクトとカラーを関連付けるのに使用されます。同名の要素が既に存在していた場合、それは上書きされます。

color 引数はSVGによって認識されるカラー式です(詳細はを参照して下さい)。

任意の*opacity* 引数を使用すると、このカラーに対して不透明度を(0から100で)指定することができます。この引数が省略された場合、不透明度は100%になります。この方法によって設定されたカラーは、"url(#ID)"形式でカラー値を渡す際に、fillまたはstrokeと関連付けることができます。

例題

```
 `blue を50%で設定
SVG_Define_solidColor($svg;"MyColor";"blue";50)
...
SVG_New_rect($svg;0;0;20;20;0;0;"url(#MyColor)";"url(#MyColor)")
...
$line:=SVG_New_line(10;10;100;100)
SVG_SET_STROKE_BRUSH($line;"url(#MyColor)")
```

SVG_Define_style

SVG_Define_style (parentSVGObject ; style {; type {; media}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	⇒	親要素の参照
style	テキスト	⇒	スタイル定義、または使用するファイルのパス名
type	テキスト	⇒	コンテンツのタイプ
media	テキスト	⇒	メディアディスクリプター
戻り値	SVG_Ref	⇐	スタイルの参照

説明

SVG_Define_styleは、*parentSVGObject* で指定したSVGコンテナー内に新しいスタイルシートを設定し、その参照を返します。*parentSVGObject* がSVGドキュメントではなく、またSVGドキュメントに属していない場合、エラーが生成されます。

style 引数を使用してSVGコンテンツ内に直接スタイルシートを埋め込みます:

- *style* 引数に有効なCSSファイルへのパス名が渡された場合、スタイル定義は外部スタイルシートを参照することで行われます。パスが"#"または"file:"で始まっている場合、そのパスはデータベースの"Resources"フォルダーをルートとする相対パスであることを示します。
- *style* 引数にはURLである"http://… "タイプを渡すこともできます。この場合スタイルは外部リソースを参照します。

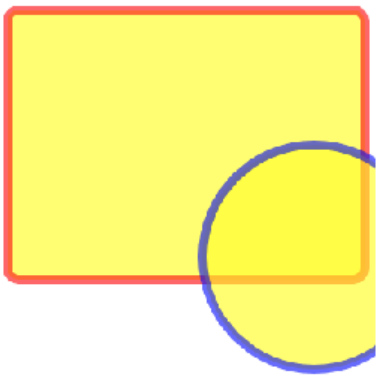
オプションの *type* 引数はスタイルシートのランゲージを指定するために使用します。デフォルト値は"text/css"です。

オプションの *media* 引数はスタイル情報を適用するメディアを指定するために使用します。この引数を省略すると、デフォルト値として"all"が使用されます。この引数に渡す値はCSS2で規定されたメディアタイプリストに含まれるものでなければなりません。そうでない場合エラーが生成されます。

参照: http://www.w3.org/TR/SVG/styling.html#StyleElement

例題 1

埋め込みスタイルとオーバーレイを設定:

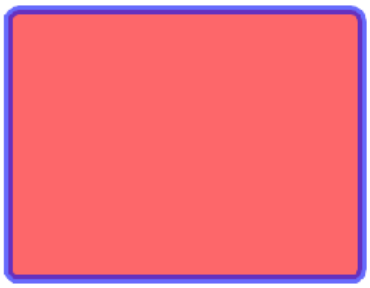


```
  //スタイルの定義
 $Txt_style:=".colored {fill: yellow; fill-opacity: 0.6; stroke: red;stroke-width:8; stroke-
opacity: 0.6}"
 SVG_Define_style($Dom_SVG;$Txt_style)

  //グループを作成し、デフォルトスタイルを割り当てる
 $Dom_g:=SVG_New_group($Dom_SVG)
 SVG_SET_CLASS($Dom_g;"colored")

  //四角を描画する
 $Dom_rect:=SVG_New_rect($Dom_g;25;30;320;240;10;10;"";"")

  //円を描画し、カスタム枠線色を使用してオーバーレイを設定する
```

```
$Dom_circle:=SVG_New_circle($Dom_g;300;250;100;"";"")
SVG_SET_STROKE_BRUSH($Dom_circle;"blue")
```

例題 2

"Resources/dev/mystyle.css"ファイルを参照する:



```
 //スタイル定義
SVG_Define_style($Dom_svg;"#dev/mystyle.css")

 //グループを作成しデフォルトスタイルを割り当てる
$Dom_g:=SVG_New_group($Dom_SVG)
SVG_SET_CLASS($Dom_g;"colored")

 //四角を描画
$Dom_rect:=SVG_New_rect($Dom_g;25;30;320;240;10;10;"";"")
```

mystyle.css ファイル:

```
.colored {fill: red; fill-opacity: 0.6; stroke: blue; stroke-width:8; stroke-opacity: 0.6}
```

SVG_DEFINE_STYLE_WITH_ARRAYS

SVG_DEFINE_STYLE_WITH_ARRAYS (svgObject ; namesArrayPointer ; valuesArrayPointer {; className {; type {; media {; title}}}})

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクト参照
namesArrayPointer	ポインター	➡	スタイル名配列へのポインター
valuesArrayPointer	ポインター	➡	スタイル値配列へのポインター
className	テキスト	➡	CSSクラス名
type	テキスト	➡	スタイルシートランゲージ
media	テキスト	➡	メディア記述子
title	テキスト	➡	スタイル名

説明

SVG_DEFINE_STYLE_WITH_ARRAYSメソッドは、*svgObject* 引数で指定したSVGオブジェクト用に (配列を使用して) スタイルを定義します。

- *svgObject* 引数がルート要素である場合、スタイルは"defs"セクションに含まれる"style"要素として定義されます (内部スタイルシート)。この場合 *className* 引数は必須です (渡されない場合エラーが生成されます)。この後**SVG_SET_CLASS** に名前を指定することで、SVGオブジェクトに*className*スタイルシートを割り当てることができます (例題1参照)。
- *svgObject* 引数がルート要素でないSVG要素を参照している場合、スタイルはこの要素のスタイル属性として設定されます (インラインスタイル) (例題2参照)。

オプションの *type* 引数を使用してスタイルシートランゲージを指定できます。省略した場合のデフォルトは "text/css" です。

オプションの *media* 引数を使用してスタイル情報を適用するメディアを指定できます。この引数を省略すると、デフォルトで"all"が使用されます。指定した値がCSS2で認識可能なメディアタイプに含まれない場合、エラーが生成されます。

オプションの *title* 引数は "title" タイプ属性を追加します。

例題 1

内部スタイル定義例:

```
ARRAY TEXT($arrnames;0)
ARRAY TEXT($arrvalues;0)
APPEND TO ARRAY($arrnames;"fill")
APPEND TO ARRAY($arrvalues;"black")
APPEND TO ARRAY($arrnames;"font-family")
APPEND TO ARRAY($arrvalues;"'Lucida Grande' Verdana")
APPEND TO ARRAY($arrnames;"font-size")
APPEND TO ARRAY($arrvalues;"20px")
APPEND TO ARRAY($arrnames;"text-align")
APPEND TO ARRAY($arrvalues;"center")

$svg:=SVG_New
SVG_DEFINE_STYLE_WITH_ARRAYS($svg;->$arrnames;->$arrvalues;"title")
$object:=SVG_New_textArea($svg;"Hello World!";10;10;200;310)
SVG_SET_CLASS($object;"title")
```

このメソッドは以下のコードを生成します:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?> <svg xmlns="http://www.w3.org/2000/svg">
    <defs id="4D">         <style type="text/css">.title{fill:red;font-family:'Lucida Grande'
Verdana;font-size:20px;text-align:center;}</style>     </defs>     <textArea class="title"
height="310" width="200" x="10" y="10">Hello World!</textArea> </svg>
```

例題 2

インラインスタイル定義の例題:

```
ARRAY TEXT($arrnames;0)
ARRAY TEXT($arrvalues;0)
APPEND TO ARRAY($arrnames;"fill")
APPEND TO ARRAY($arrvalues;"black")
APPEND TO ARRAY($arrnames;"font-family")
APPEND TO ARRAY($arrvalues;"'Lucida Grande' Verdana")
APPEND TO ARRAY($arrnames;"font-size")
APPEND TO ARRAY($arrvalues;"20px")
APPEND TO ARRAY($arrnames;"text-align")
APPEND TO ARRAY($arrvalues;"center")

$svg:=SVG_New
$object:=SVG_New_textArea($svg;"Hello World!";10;10;200;310)
SVG_DEFINE_STYLE_WITH_ARRAYS($object;->$arrnames;->$arrvalues)
```

このメソッドは以下のコードを生成します:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?> <svg xmlns="http://www.w3.org/2000/svg">
    <textArea height="310" style="fill:red;font-family:'Lucida Grande' Verdana;font-
size:20px;text-align:center;" width="200" x="10" y="10">Hello World!</textArea> </svg>
```

SVG_Define_symbol

SVG_Define_symbol (parentSVGObject ; id {; x {; y {; width {; height {; mode}}}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
id	文字	➡	シンボル名
x	倍長整数	➡	ビューボックスのX軸の位置
y	倍長整数	➡	ビューボックスのY軸の位置
width	倍長整数	➡	ビューボックスの幅
height	倍長整数	➡	ビューボックスの高さ
mode	文字	➡	ビューボックスへの調整
戻り値	SVG_Ref	↩	シンボルの参照

説明

SVG_Define_symbol コマンドは*parentSVGObject* 引数で指定されたSVGコンテナ内にシンボルを作成し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではなかった場合、エラーが生成されます。

シンボルオブジェクトは、グラフィックオブジェクトを指定するのに使用します(グラフィックオブジェクトは*SVG_Use* コマンドを使用するなどしてインスタンス化できます)。

id 引数はシンボル名を指定します。

任意の*x*、*y*、*width* と *height* 引数はビューボックスの長方形('viewBox' 属性)を指定します。

任意の*mode* 引数はグラフィックをビューボックスにちょうど納めたい場合に、どのように納めるべきかを指定します。この点についてのより詳細な情報に関しては、*SVG_New* コマンドの詳細を参照して下さい。

例題

SVG_Use コマンドの詳細を参照して下さい。

SVG_DELETE_OBJECT

SVG_DELETE_OBJECT (svgObject)

引数	型		説明
svgObject	SVG_Ref	⇨	SVG要素の参照

説明

SVG_DELETE_OBJECTはSVGドキュメントから*svgObject* で指定したSVGオブジェクトを削除します。*svgObject* が有効な参照でない場合、エラーが生成されます。

SVG_Get_default_encoding

SVG_Get_default_encoding -> 戻り値

引数	型		説明
戻り値	テキスト		文字セット

説明

SVG_Get_default_encodingコマンドは新しいドキュメントを作成する際に使用される文字エンコーディングを返します。

SVG_New_group

SVG_New_group (parentSVGObject {; id {; url {; target}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	⇒	親要素の参照
id	文字	⇒	グループ名
url	文字	⇒	外部リンク
target	文字	⇒	リンクのターゲット
戻り値	SVG_Ref	⇐	グループの参照

説明

SVG_New_group コマンドは*parentSVGObject* 引数で指定されたSVGコンテナ内にグループを作成し、その参照を返します。*parentSVGObject* 引数が有効なSVGグループまたはドキュメント出なかった場合には、エラーが生成されます。

作成したグループ('g' 要素)を使用するとリンクされた複数のグラフィック要素をグループ化することができ、作成したグループのプロパティを引き継ぎます。

任意の*id* 引数を使用すると、グループに名前を付けることができます。命名されたグループはアニメーションや再利用可能オブジェクトなど、いくつかの目的において必要となります。

任意の*url* 引数を使用すると、外部リンクをグループに割り当てることができます。外部リンクを割り当てると、グループはクリック可能になります(HTMLの'a' 要素に近いです)。

任意の*target* 引数はリンクがクリックされた際にドキュメントが開くターゲット名を指定します。渡せる値はHTML仕様に準じていて、'new' を渡すと新しいウィンドウでリンクを開き、'none' を渡すとこの属性を処理しないのと同じになります。

例題 1

全て同色の、線のグループを描画する場合を考えます:

```
$SVG:=SVG_New
$group:=SVG_New_group($SVG)
 `グループ要素にカラーを割り当てます
SVG_SET_STROKE_BRUSH($group;"firebrick")
$newobject:=SVG_New_line($group;100;300;300;100;"";5)
$newobject:=SVG_New_line($group;300;300;500;100;"";10)
$newobject:=SVG_New_line($group;500;300;700;100;"";15)
$newobject:=SVG_New_line($group;700;300;900;100;"";20)
$newobject:=SVG_New_line($group;900;300;1100;100;"";25)
```

例題 2

クリック可能なテキストを描画する場合を考えます:


```
$SVG:=SVG_New
$group:=SVG_New_group($SVG;"w3Link";"http://www.w3.org";"new")
$newobject:=SVG_New_text($group;"www.w3.org";10;10;"arial";12;Underline;Align left;"blue")
```

SVG_SET_DEFAULT_ENCODING

SVG_SET_DEFAULT_ENCODING {(encoding)}

引数	型		説明
encoding	文字	⇨	文字セット

説明

SVG_SET_DEFAULT_ENCODINGコマンドを使用して、新しいドキュメントが作成されるときに使用される文字エンコーディングを設定します。

encoding 引数が省略された場合、コマンドはデフォルト文字セットである"UTF-8"を設定します。

SVG_Set_description

SVG_Set_description (parentSVGObject ; description) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	⇒	親要素の参照
description	文字	⇒	コメントのテキスト
戻り値	SVG_Ref	⇐	詳細の参照

説明

SVG_Set_description コマンドは*parentSVGObject* 引数で指定されたSVG要素に対してテキストを設定し、その参照を返します。*parentSVGObject* 引数がSVG要素ではない場合、エラーが生成されます。

詳細は主に、SVGコードにコメントや説明文を挿入するのに使用されます。

例題

```
$SVG:=SVG_New
$g:=SVG_group($SVG)
SVG_Set_title($g;"地域ごとの会社の売り上げ")
SVG_Set_description($g;"地域ごとの会社の売り上げの棒グラフ")
```

SVG_SET_PATTERN_CONTENT_UNITS

SVG_SET_PATTERN_CONTENT_UNITS (patternObject ; sysCoord)

引数	型		説明
patternObject	SVG_Ref	➡	更新するパターンの参照
sysCoord	テキスト	➡	使用する座標システム

説明

SVG_SET_PATTERN_CONTENT_UNITSは*patternObject*で指定されたパターンの座標システムを設定するために使用します。*patternObject* がパターンでない場合、エラーが生成されます。

sysCoord 引数には使用するシステムを渡します。それは"userSpaceOnUse"または"objectBoundingBox"のいずれかでなければならず、それ以外の場合エラーが生成されます。

参照: http://www.w3.org/TR/SVG/pservers.html#Patterns

SVG_SET_PATTERN_UNITS

SVG_SET_PATTERN_UNITS (patternObject ; sysCoord)

引数	型		説明
patternObject	SVG_Ref	➡	変更するパターンの参照
sysCoord	テキスト	➡	使用する座標システム

説明

SVG_SET_PATTERN_UNITSコマンドを使用して、*patternObject*で指定したパターンの*x*, *y*, *width* そして *height*属性で使用されるシステム座標を設定できます。*patternObject* がパターンでない場合、エラーが生成されます。

sysCoord 引数で使用するシステムの名前を指定します。"userSpaceOnUse"または"objectBoundingBox"のいずれかを指定します。そうでない場合エラーが生成されます。

参照: http://www.w3.org/TR/SVG/pservers.html#Patterns

SVG_Set_title

SVG_Set_title (parentSVGObject ; title) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	⇒	親要素の参照
title	文字	⇒	タイトルのテキスト
戻り値	SVG_Ref	⇐	タイトルの参照

説明

SVG_Set_title コマンドは*parentSVGObject* 引数で指定されたSVG要素に対しタイトルを指定し、その参照を返します。*parentSVGObject* 引数がSVG要素ではない場合、エラーが生成されます。

タイトルとは、レンダリングされたピクチャーには含まれないものの、複雑なドキュメント構成する際に有用なテキストデータです。一部のSVGレンダリングエンジンは、マウスがそのオブジェクト上にマウスオーバーした際に表示するヘルプTipsとしてこの要素のテキストを使用します。

例題

```
$SVG:=SVG_New
$rec:=SVG_New_rect($SVG;20;20;650;650;0;0;"gray";"lemonchiffon")
SVG_Set_title($rec;"Background rectangle")
$Symbol:=SVG_Define_symbol($SVG;"MySymbol";0;0;110;110;"true")
SVG_Set_title($Symbol;" Set a symbol composed of 2 squares and 2 circles ")
...
```

テキスト

- SVG_APPEND_TEXT_TO_TEXTAREA
- SVG_Get_text
- SVG_New_text
- SVG_New_textArea
- SVG_New_tspan
- SVG_New_vertical_text
- SVG_SET_FONT_COLOR
- SVG_SET_FONT_FAMILY
- SVG_SET_FONT_SIZE
- SVG_SET_FONT_STYLE
- SVG_SET_TEXT_ANCHOR
- SVG_SET_TEXT_KERNING
- SVG_SET_TEXT_LETTER_SPACING
- SVG_SET_TEXT_RENDERING
- SVG_SET_TEXT_WRITING_MODE
- SVG_SET_TEXTAREA_TEXT

SVG_APPEND_TEXT_TO_TEXTAREA

SVG_APPEND_TEXT_TO_TEXTAREA (svgObject ; addedText)

引数	型		説明
svgObject	SVG_Ref	➡	テキスト要素の参照
addedText	テキスト	➡	追加するテキスト

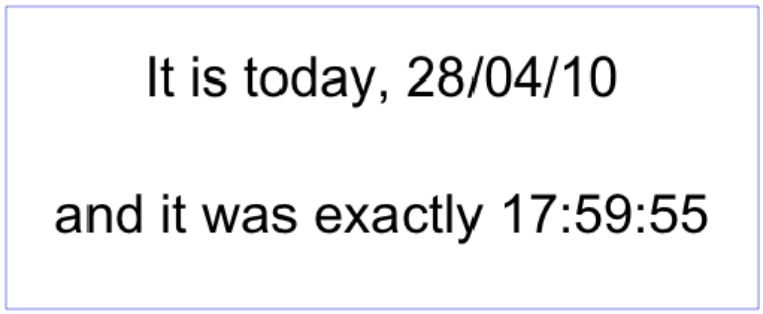
説明

SVG_APPEND_TEXT_TO_TEXTAREA コマンドを使用して、*svgObject*で指定したテキストオブジェクトにテキストを追加できます。*svgObject*が"textArea"オブジェクトでない場合、エラーが生成されます。

改行文字は自動で"<tbreak/>"要素に置き換えられます。

例題

以下のテキストを追加します:




```
 // 'rect' 要素を使用して枠線を表示
$Dom_rect:=SVG_New_rect($Dom_SVG;10;10;500;200;0;0;"blue:50";"none")

 // テキストエリアを作成
$Dom_text:=SVG_New_textArea($Dom_SVG;"It is today, ";10;30;500;200;"'Arial'";36;0;3)

 // 日付と改行 x 2を追加
SVG_APPEND_TEXT_TO_TEXTAREA($Dom_text;String(Current date)+"\r\r")

 // 最後に時刻を追加
SVG_APPEND_TEXT_TO_TEXTAREA($Dom_text;"and it was exactly "+String(Current time))
```

SVG_Get_text

SVG_Get_text (svgObject) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	➡	テキスト要素の参照
戻り値	テキスト	⮌	テキストコンテンツ

説明

SVG_Get_textは*svgObject*で指定した要素のテキストコンテンツを返します。*svgObject*がテキストオブジェクト ('text'、'textArea'、'tspan') の参照でない場合、エラーが生成されます。

textAreaオブジェクトの場合、<tbreak/>要素はCRに置き換えられます。

SVG_New_text

SVG_New_text (parentSVGObject ; text {; x {; y {; font | styleDef {; size {; style {; alignment {; color {; rotation {; lineSpacing {; stretching}}}}}}}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
text	テキスト	➡	挿入するテキスト
x	実数	➡	X軸の座標
y	実数	➡	Y軸の座標
font | styleDef	テキスト	➡	フォント名またはスタイル定義
size	倍長整数	➡	文字サイズのポイント数
style	倍長整数	➡	文字スタイル
alignment	倍長整数	➡	行揃え
color	文字	➡	テキストカラー
rotation	実数	➡	テキストの回転角度
lineSpacing	実数	➡	行間隔のポイント数
stretching	実数	➡	水平ストレッチファクター
戻り値	SVG_Ref	⮌	SVGテキストオブジェクト参照

説明

SVG_New_text コマンドは*parentSVGObject* 引数で指定されたSVGコンテナ内にテキストを挿入し、その参照を返します。*parentSVGObject* 引数がSVGドキュメント出ない場合には、エラーが生成されます。

注: 4D v15以降、**SVG_New_text** コマンドは**スタイル付きテキスト** をサポートするようになりました(例題5を参照して下さい)。

任意の*x* と *y* 引数を使用するとテキストの最初の文字の上端のX軸とY軸の座標を指定することができます。この点は、行揃えの値に応じて扱われ方が異なってきます。左揃えの場合にはこの点は文字列の左端として扱われ、右揃えの場合にはこの点は文字列の右端として扱われます。中央揃えの場合にはこの点が中央として扱われます。

SVG_New_text コマンドは二つの異なる文字設定のシンタックスを受け付けます:

- *font*、*size*、*style* と *alignment* 引数に対して様々な値を渡す事ができます:*font* と *size* 引数を使用するとフォントとサイズをポイント単位で指定できます。これらの引数が省略された場合、テキストはで**Times New Roman 12 pts** で描画されます。
  任意の*style* 引数は使用される文字スタイルに関する情報を渡します。*style* 引数には、以下の値のどれか一つ、または値同士の組み合わせを渡す必要があります(または、4D定数の**Font Styles** テーマから、それぞれのスタイルに対応する値を使用することもできます):

  0 = 標準
  1 = 太字
  2 = 斜字
  4 = 下線付き
  8 = 打消し線付き

  任意の*alignment* 引数を使用すると描画されたテキストに対して適用する行揃えのタイプを指定する事ができます。渡す事ができる値は以下の通りです:

  2 = 左揃え
  3 = 中央揃え
  4 = 右揃え

  任意の*color* 引数はフォントカラーを受け付けます(カラーに関する詳細については、**カラー&グラデーション** の章を参照して下さい)。
  任意の*rotation* 引数を使用するとテキストに適用する回転を指定することができます。
  任意の*lineSpacing* 引数を使用するとテキストが複数行にまたがった際の行間隔を指定する事ができます。デフォルトの値は1です。

任意の*stretching* 引数を使用するとテキストの水平方向の拡大(値>1)または縮小(0から1の間の値)をしていることができます。

- または、(*font* 引数の代わりに)*styleDef* 引数にスタイル定義を渡し、それに続く引数を省略することもできます。例えば、以下の様に指定する事ができます。:

```
SVG_New_textArea($Dom_svg;"Hello World !";x;y;vWidth;vHeight;style_definition)
```

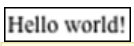
... ここでは*style_definition* 引数には完全なスタイル定義が含まれています。例えば、"{font-size:48px;fill:red;}"を渡した場合、この定義はスタイル属性としてフォームに追加されます:

```
style="font-size:48px;fill:red;"
```

この場合、これ以外の引数は全て無視されます。

例題 1

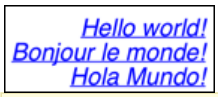
デフォルトのテキストプロパティを使用したシンプルなテキスト:




```
$SVG:=SVG_New
$textID:=SVG_New_text($SVG;"Hello world!")
```

例題 2

青い、斜字で下線付きの右揃えのテキスト:




```
$SVG:=SVG_New
$text:="Hello world!\rBonjour le monde!\rHola Mundo!"
$size:=48
$font:="helvetica"
$textID:=SVG_New_text($SVG;$text;400;10;$font;$size;Italic+Underline;Align right;"blue")
```

例題 3

垂直なテキスト:


```
$SVG:=SVG_New
$textID:=SVG_New_text($SVG;$text;-250;0;"";48;-1;-1;"red";-90)
```

例題 4

縮小、または拡大されたテキスト:


```
$SVG:=SVG_New
$textID:=SVG_New_text($SVG;"Hello world (condensed)";0;0;"";-1;-1;-1;"blue";0;1;0,8)
$textID:=SVG_New_text($SVG;"Hello world (normal)";0;24)
$textID:=SVG_New_text($SVG;"Hello world (stretched)";0;48;"";-1;-1;-1;"red";0;1;2)
```

例題 5

マルチスタイルテキストの表示:

```
C_TEXT($Dom_svg;$Dom_text;$Txt_buffer)
 //マルチスタイルテキストを定義
$Txt_buffer:="<SPAN STYLE=\"font-size:18pt\">Hello </SPAN>"+\
"<SPAN STYLE=\"font-size:24pt;font-weight:bold;color:#D81E05\">World</SPAN>"+\
"<SPAN STYLE=\"font-size:36pt\">!</SPAN><BR/>"+\
"<SPAN STYLE=\"font-size:19pt;font-style:italic\">It's </SPAN>"+\
"<SPAN STYLE=\"font-size:24pt\">Monday</SPAN>"
$Dom_svg:=SVG_New

 //タイトル
SVG_SET_FONT_COLOR(SVG_New_text($Dom_svg;"_______ svg_Newtext _______";10;30);"blue")
 //テキスト
$Dom_text:=SVG_New_text($Dom_svg;$Txt_buffer;50;50)

SVGTool_SHOW_IN_VIEWER($Dom_svg)
SVG_CLEAR($Dom_svg)
```

SVG_New_textArea

SVG_New_textArea (parentSVGObject ; text {; x {; y {; textWidth {; textHeight {; font | styleDef {; size {; style {; alignment}}}}}}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
text	テキスト	➡	挿入するテキスト
x	倍長整数	➡	X軸の座標
y	倍長整数	➡	Y軸の座標
textWidth	倍長整数	➡	テキストエリアの幅
textHeight	倍長整数	➡	テキストエリアの高さ
font | styleDef	テキスト	➡	フォント名またはスタイル定義
size	整数	➡	ポイント単位での文字サイズ
style	整数	➡	文字のスタイル
alignment	整数	➡	行揃え
戻り値	SVG_Ref	⬅	SVGテキストオブジェクトの参照

説明

SVG_New_textArea コマンドは*parentSVGObject* 引数で指定されたSVGコンテナ内にテキストエリアを挿入し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントじゃない場合、エラーが生成されます。

"textArea"要素はSVG Tiny 1.2で推奨され4D v11.3以降でv11 SQLに実装された要素です(詳細はhttp://www.w3.org/TR/SVGMobile12/text.html#TextAreaElement参照して下さい)。この要素は"text"要素とは異なり、テキストが指定された幅を越えた際には自動的に改行を管理するテキストエリアを実装します。

注:

- "textArea" 要素内では、<tbreak/> 要素が改行を置き換えます。
- 4D v15以降、**SVG_New_textArea** コマンドは**スタイル付きテキスト** をサポートするようになりました(例題2を参照して下さい)。

任意の *x* と *y* 引数を使用するとエリアの左上端の位置のX軸とY軸の位置を指定することができます。

任意の*textWidth* と *textHeight* 引数はユーザー座標系ないでのエリアのサイズを指定します。これらの引数のどちらか一つが渡されない場合、テキストエリアは自動的にそのコンテンツに合わせたサイズとなります。

SVG_New_textArea コマンドは二つの異なる文字設定のシンタックスを受け付けます:

- *font*、*size*、*style* と *alignment* 引数に対して様々な値を渡す事ができます:*font* と *size* 引数を使用するとフォントとサイズをポイント単位で指定できます。これらの引数が省略された場合、テキストはで**Times New Roman 12 pts** で描画されます。
  任意の*style* 引数は使用される文字スタイルに関する情報を渡します。*style* 引数には、以下の値のどれか一つ、または値同士の組み合わせを渡す必要があります(または、4D定数の**Font Styles** テーマから、それぞれのスタイルに対応する値を使用することもできます):

  0 = 標準
  1 = 太字
  2 = 斜字
  4 = 下線付き
  8 = 打消し線付き

  任意の*alignment* 引数を使用すると描画されたテキストに対して適用する行揃えのタイプを指定する事ができます。渡す事ができる値は以下の通りです:

  2 = 左揃え
  3 = 中央揃え
  4 = 右揃え

任意の*color* 引数はフォントカラーを受け付けます(カラーに関する詳細については、**カラー&グラデーション** の章を参照して下さい)。
任意の*rotation* 引数を使用するとテキストに適用する回転を指定することができます。
任意の*lineSpacing* 引数を使用するとテキストが複数行にまたがった際の行間隔を指定する事ができます。デフォルトの値は1です。
任意の*stretching* 引数を使用するとテキストの水平方向の拡大(値>1)または縮小(0から1の間の値)をしていすることができます。

- または、(*font* 引数の代わりに)*styleDef* 引数にスタイル定義を渡し、それに続く引数を省略することもできます。例えば、以下の様に指定する事ができます。:

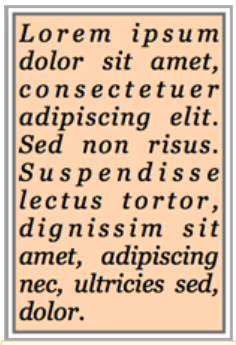
```
SVG_New_textArea($Dom_svg;"Hello World !";x;y;vWidth;vHeight;style_definition)
```

... ここでは*style_definition* 引数には完全なスタイル定義が含まれています。例えば、"{font-size:48px;fill:red;}"を渡した場合、この定義はスタイル属性としてフォームに追加されます:

```
style="font-size:48px;fill:red;"
```

この場合、これ以外の引数は全て無視されます。

例題 1




```
$svg:=SVG_New
 `境界線の四角を設定する
$rec:=SVG_New_rect($svg;5;5;210;320;0;0;"#777";"peachpuff";3)
 `テキストを用意
$txt:="Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor."
$txtArea:=SVG_New_textArea($svg;$txt;10;10;200;310;"Georgia";25;Italic;5)
 `ドキュメントを保存
SVG_SAVE_AS_TEXT($svg;"test.svg")
```

例題 2

マルチスタイルテキストの表示:

```
C_TEXT($Dom_svg;$Dom_text;$Txt_buffer)
 //マルチスタイルテキストの定義
$Txt_buffer:="<SPAN STYLE=\"font-size:18pt\">Hello </SPAN>"+\
"<SPAN STYLE=\"font-size:24pt;font-weight:bold;color:#D81E05\">World</SPAN>"+\
"<SPAN STYLE=\"font-size:36pt\">!</SPAN><BR/>"+\
"<SPAN STYLE=\"font-size:19pt;font-style:italic\">It's </SPAN>"+\
"<SPAN STYLE=\"font-size:24pt\">Monday</SPAN>"
$Dom_svg:=SVG_New

 //タイトル
SVG_SET_FONT_COLOR(SVG_New_text($Dom_svg;"_______ SVG_New_textArea ______";10;30;"";-1);"blue")
```

```
 //テキストエリア
$Dom_text:=SVG_New_textArea($Dom_svg;$Txt_buffer;50;50)

SVGTool_SHOW_IN_VIEWER($Dom_svg)
SVG_CLEAR($Dom_svg)
```

SVG_New_tspan

SVG_New_tspan (parentSVGObject ; text {; x {; y {; font | styleDef {; size {; style {; alignment {; color}}}}}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
text	テキスト	➡	挿入するテキスト
x	倍長整数	➡	X軸の座標
y	倍長整数	➡	Y軸の座標
font | styleDef	テキスト	➡	フォント名またはスタイル定義
size	整数	➡	ポイント単位での文字サイズ
style	整数	➡	文字のスタイル
alignment	整数	➡	行揃え
color	文字	➡	テキストカラー
戻り値	SVG_Ref	↩	SVGテキストオブジェクトの参照

説明

SVG_New_tspan コマンドは*parentSVGObject* 引数で指定した'text'、'tspan' または 'textArea' 要素内に新しい要素を作成します。もし*parentSVGObject* 引数で参照した要素が'text'、'tspan' または 'textArea' 要素ではない場合、エラーが生成されます。

その他の任意の引数に関しては、**SVG_New_text** コマンドに詳細が記載されています。どれかの任意の引数が省略された場合、それらの値は親の要素から引き継がれます。

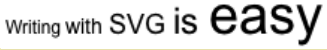
例題 1

テキストにおいて、親要素のプロパティを引き継ぐ段落を作成することが可能です。


```
$SVG:=SVG_New
 `Arialフォントで青色で左揃えの新規テキストを作成
$textID:=SVG_New_text($SVG;"";0;0;"arial";-1;-1;Align left;"blue")
 `インデントを付けて、文字のサイズとスタイルを変えて行く
$textID:=SVG_New_tspan($textID;"TITLE 1";10;10;"";24;Bold+Underline)
$textID:=SVG_New_tspan($textID;"Title 2";20;42;"";12;Bold)
$textID:=SVG_New_tspan($textID;"Title 3";30;60;"";10;Bold+Italic)
$textID:=SVG_New_tspan($textID;"Title 4";40;78;"";8;Italic)
```

例題 2

同じ"text"要素内で、プロパティを変えて行く場合を考えます。ここでテキストのサイズを変えて行きます。




```
$textID:=SVG_New_text($SVG;"Writing ";10;10;"arial";12)
SVG_SET_FONT_SIZE(SVG_New_tspan($textID;"with ");14)
SVG_SET_FONT_SIZE(SVG_New_tspan($textID;"SVG ");18)
SVG_SET_FONT_SIZE(SVG_New_tspan($textID;"is ");24)
SVG_SET_FONT_SIZE(SVG_New_tspan($textID;"easy ");36)
```

SVG_New_vertical_text

SVG_New_vertical_text (parentSVGObject ; text {; x {; y {; font {; size {; style {; alignment {; color {; rotation}}}}}}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
text	テキスト	➡	挿入するテキスト
x	倍長整数	➡	X軸の座標
y	倍長整数	➡	Y軸の座標
font	文字	➡	フォント名
size	整数	➡	ポイント単位での文字サイズ
style	整数	➡	文字のスタイル
alignment	整数	➡	行揃え
color	文字	➡	テキストカラー
rotation	倍長整数	➡	テキストの回転の角度
戻り値	SVG_Ref	⬅	SVGテキストオブジェクトの参照

説明

SVG_New_vertical_text コマンドは、*parentSVGObject* 引数で指定したSVGコンテナ内に垂直なテキストを挿入し、その参照を返します。*parentSVGObject* 引数で参照しているのがSVGドキュメントではない場合、エラーが生成されます。

任意の*x* と *y* 引数を使用すると、テキストの1文字目の左下端の位置を指定する事ができます。

任意の*font* と *size* 引数を使用すると、テキストに使用するフォントとサイズ(ポイント単位)を指定できます。これらの引数が渡されなかった場合、テキストはTimes New Roman 12 ptで描画されます。

任意の *style* 引数は使用する文字スタイルを指定します。この引数には、以下のスタイルの値のどれか一つ、またはそれらの組み合わせを渡す必要があります:

0 = 標準
1 = 太字
2 = 斜字
4 = 下線付き
8 = 打消し線付き

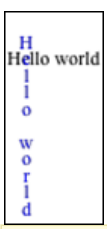
任意の*alignment* 引数を使用すると描画されたテキストに対して適用する行揃えのタイプを指定する事ができます。渡す事ができる値は以下の通りです:

2 = 左揃え
3 = 中央揃え
4 = 右揃え

任意の *color* 引数はフォントカラーを受け付けます(カラーに関する詳細については、"**カラー&グラデーション**" の章を参照して下さい)。

任意の*rotation* 引数を使用するとテキストに適用する回転を指定することができます。

例題




```
$SVG:=SVG_New
$textID:=SVG_New_text($SVG;"Hello world";10;12)
```

```
$textID:=SVG_New_vertical_text($SVG;"Hello world";22;3;"";-1;-1;Center;"blue")
```

SVG_SET_FONT_COLOR

SVG_SET_FONT_COLOR (svgObject ; color)

引数	型		説明
svgObject	SVG_Ref	⇒	SVG要素の参照
color	文字	⇒	テキストカラー

説明

SVG_SET_FONT_COLOR コマンドを使用すると、*svgObject* 参照で参照しているSVGオブジェクトのフォントカラーを指定することができます。*svgObject* が有効な参照ではない場合、エラーが生成されます。

color 引数には使用するカラー名を渡します(カラーに関する詳細については、”**カラー&グラデーション**” の章を参照して下さい)。

SVG_SET_FONT_FAMILY

SVG_SET_FONT_FAMILY (svgObject ; font {; font2 ; ... ; fontN})

引数	型		説明
svgObject	SVG_Ref	➡	SVG要素の参照
font	文字	➡	フォント名

説明

SVG_SET_FONT_FAMILY コマンドを使用すると、*svgObject* 参照で参照しているSVGオブジェクトのフォントを指定する事ができます。*svgObject* 参照が有効な要素を参照していない場合、エラーが生成されます。

font 引数には使用するフォント名を渡します。複数の名前を渡した場合、コマンドは自動的にフォントファミリー名、または一般的なファミリー名を作成します。

例題

複数のフォント名を渡した場合:

```
SVG_SET_FONT_FAMILY(svgObject;"Lucida grande";"Sans-serif")
 // この場合、" 'Lucida grande' 'Sans-serif'"というリストを作成します
```

SVG_SET_FONT_SIZE

SVG_SET_FONT_SIZ E (svgObject ; size)

引数	型		説明
svgObject	SVG_Ref	⇒	SVG要素の参照
size	整数	⇒	ポイント単位での文字サイズ

説明

SVG_SET_FONT_SIZE コマンドを使用すると、*svgObject* 参照で参照されているSVGオブジェクトのフォントのサイズを*size* 引数で指定します。*svgObject* 引数が有効な要素ではない場合、エラーが生成されます。

size 引数には、文字サイズをポイント数で渡します。

SVG_SET_FONT_STYLE

SVG_SET_FONT_STYLE (svgObject ; style)

引数	型		説明
svgObject	SVG_Ref	➡	SVG要素の参照
style	整数	➡	Style of characters

説明

SVG_SET_FONT_STYLE コマンドを使用すると、*svgObject* 参照で指定しているSVGオブジェクトのテキストスタイルを指定することができます。*svgObject* が有効な要素への参照ではない場合、エラーが生成されます。

style 引数には、以下の値の一つまたは複数の組み合わせを渡す必要があります:

0 = 標準
1 = 太字
2 = 斜字
4 = 下線付き
8 = 打消し線付き

SVG_SET_TEXT_ANCHOR

SVG_SET_TEXT_ANCHOR (svgObject ; alignment)

引数	型		説明
svgObject	SVG_Ref	⇒	SVG要素の参照
alignment	整数	⇒	行揃え

説明

SVG_SET_TEXT_ANCHOR コマンドを使用すると、*svgObject* 引数で参照している行揃えを変更することができます。*svgObject* 引数が有効な要素を参照していない場合、エラーが生成されます。
alignment 引数には、以下の値のどれか一つを渡す必要があります:

1 = デフォルトの行揃え(左)
2 = 左揃え
3 = 中央揃え
4 = 右揃え
5 = 両端揃え(textArea オブジェクトのみ)

SVG_SET_TEXT_KERNING

SVG_SET_TEXT_KERNING (svgObject ; kerning {; unit})

引数	型		説明
svgObject	SVG_Ref	➡	テキスト要素の参照
kerning	実数	➡	カーニング幅
unit	テキスト	➡	値の単位

説明

SVG_SET_TEXT_KERNINGコマンドを使用して、*svgObject*で指定したテキストオブジェクトのカーニングを変更できます。*svgObject*がSVGテキストオブジェクトでない場合、エラーが生成されます。

オプションの*unit*引数を使用してカーニング値の単位を指定できます。デフォルトの単位は"%"です。

kerning に -1 を渡すと'auto'が設定されます。

注: Windowsでは実装が左から右、上から下のテキスト、および'text'と'tspan'に限定されます (右から左に書くテキストでは無効です)。Mac OSではサポートに制限はありません。

参照: http://www.w3.org/TR/SVG/text.html#KerningProperty

例題

様々なカーニングの例題です:

Hello world !
Hello world !
Hello world !
Hello world !
Hello world !
Hello world !
Hello world !
Hello world !

```
 //Reference
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;40;"";36)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;80;"";36)
SVG_SET_TEXT_KERNING($Dom_text;0,5)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;120;"";36)
SVG_SET_TEXT_KERNING($Dom_text;1)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;160;"";36)
SVG_SET_TEXT_KERNING($Dom_text;1,5)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;200;"";36)
SVG_SET_TEXT_KERNING($Dom_text;2)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;240;"";36)
SVG_SET_TEXT_KERNING($Dom_text;1,5)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;280;"";36)
SVG_SET_TEXT_KERNING($Dom_text;1)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;320;"";36)
SVG_SET_TEXT_KERNING($Dom_text;0,5)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;360;"";36)
SVG_SET_TEXT_KERNING($Dom_text;0)
```

SVG_SET_TEXT_LETTER_SPACING

SVG_SET_TEXT_LETTER_SPACING (svgObject ; spacing {; unit})

引数	型		説明
svgObject	SVG_Ref	➡	テキスト要素の参照
spacing	実数	➡	文字間隔
unit	テキスト	➡	値の単位

説明

SVG_SET_TEXT_LETTER_SPACINGを使用して、*svgObject* で指定したテキストオブジェクトの文字間隔を変更できます (カーニングプロパティとは異なります)。*svgObject*がSVGテキストオブジェクトでない場合、エラーが生成されます。

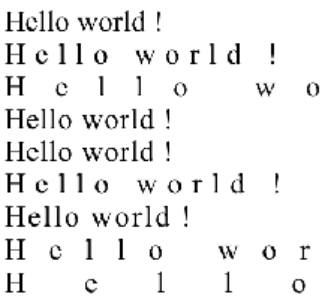
オプションの*unit*引数を使用して文字間隔の値の単位を指定できます。デフォルトは"%"です。

spacing に -1 を指定すると'normal'が設定されます。

参照: http://www.w3.org/TR/SVG/text.html#LetterSpacingProperty

例題

様々な文字間隔の例題です:




```
 //Reference
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;40;"";36)

$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;80;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1)
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;120;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"em")
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;160;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"px")
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;200;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"pt")
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;240;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"pc")
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;280;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"mm")
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;320;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"cm")
$Dom_text:=SVG_New_text($Dom_SVG;"Hello world !";20;360;"";36)
SVG_SET_TEXT_LETTER_SPACING($Dom_text;1;"in")
```

SVG_SET_TEXT_RENDERING

SVG_SET_TEXT_RENDERING (svgObject ; rendering)

引数	型		説明
svgObject	SVG_Ref	➡	テキスト要素の参照
rendering	テキスト	➡	レンダリングの値

説明

SVG_SET_TEXT_RENDERINGコマンドを使用して、*svgObject*で指定したテキストオブジェクトのテキストレンダリングに関わるトレードオフを指定できます。*svgObject*がSVGテキストオブジェクトでない場合、エラーが生成されます。

rendering 引数には以下のいずれかの値を渡すことができます:
"auto"、"optimizeSpeed"、"optimizeLegibility"、"geometricPrecision"または"inherit"。これ以外の場合、エラーが生成されます。

参照: http://www.w3.org/TR/2001/REC-SVG-20010904/painting.html#TextRenderingProperty

SVG_SET_TEXT_WRITING_MODE

SVG_SET_TEXT_WRITING_MODE (svgObject ; writingMode)

引数	型		説明
svgObject	SVG_Ref	➡	テキスト要素の参照
writingMode	テキスト	➡	文字書き方向

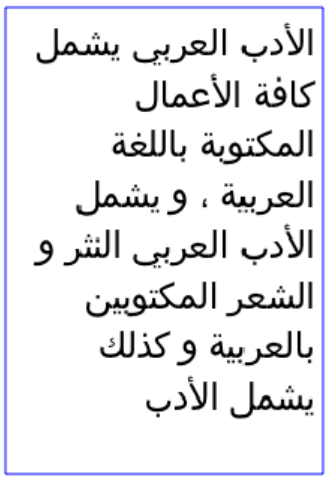
説明

SVG_SET_TEXT_WRITING_MODEコマンドを使用して、*svgObject*で指定したテキストオブジェクトの書き方向を左から右、右から左、上から下、下から上に設定できます。*svgObject*がSVGテキストオブジェクトでない場合、エラーが生成されます。

*writingMode*には以下のいずれかの値を指定できます: "lr-tb"、"rl-tb"、"tb-rl"、"lr"、"rl"、 "tb" または "inherit"。これら以外の場合、エラーが生成されます。

例題

右から左の例:




```
 //フレーム
SVG_New_rect($Dom_SVG;5;5;210;310;0;0;"blue";"none")

 //テキスト
$Dom_text:=SVG_New_textArea($Dom_SVG;$Txt_sample;10;10;200;300;"Baghdad 'Arial Unicode MS'";25)
SVG_SET_TEXT_WRITING_MODE($Dom_text;"rl")
```

SVG_SET_TEXTAREA_TEXT

SVG_SET_TEXTAREA_TEXT (svgObject ; theText)

引数	型		説明
svgObject	SVG_Ref	➡	テキスト要素の参照
theText	テキスト	➡	設定するテキスト

説明

SVG_SET_TEXTAREA_TEXTを使用して、*svgObject*で指定したテキストオブジェクトの内容テキストを設定、あるいは置換できます。*svgObject*が"textArea"オブジェクトでない場合、エラーが生成されます。

改行文字は自動で"<tbreak/>"要素に置換されます。

ドキュメント

- SVG_CLEAR
- SVG_Copy
- SVG_Export_to_picture
- SVG_Export_to_XML
- SVG_New
- SVG_Open_file
- SVG_Open_picture
- SVG_SAVE_AS_PICTURE
- SVG_SAVE_AS_TEXT
- SVG_SET_DOCUMENT_VARIABLE
- SVG_Validate_file

SVG_CLEAR

SVG_CLEAR {(svgObject)}

引数	型		説明
svgObject	SVG_Ref	⇒	SVG オブジェクト参照

説明

SVG_CLEAR コマンドは、*svgObject* 引数で指定されたSVGオブジェクトによって占有されていたメモリーを解放します。

svgObject 引数にはSVGルートオブジェクト(*SVG_New*、*SVG_Copy* または *SVG_Open_file* コマンドによって作成されたもの)や、他のどんな有効なSVGオブジェクトでも可能です。

svgObject 引数が渡されなかった場合、コマンドは*SVG_New*、*SVG_Copy* または *SVG_Open_file* コマンドによって作成された全てのSVGオブジェクトを解放します。このシンタックスは開発フェーズにおいて有効です。例えば、SVG参照が作成されたもの、エラーによってメソッドが完了しなかったためにメモリが開放されていなかった場合、などです。最終開発においては、使用されていないSVG参照は全 てを*SVG_CLEAR* を使用してメモリを開放すべきです。

SVG_Copy

SVG_Copy (svgObject) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	➡	コピーするSVGオブジェクトの参照
戻り値	SVG_Ref	⮌	新しいSVGオブジェクトの参照

説明

SVG_Copy コマンドは、*svgObject* 引数によって参照されているドキュメントのコピーであるSVGドキュメントを作成します。

コマンドは、メモリー内のドキュメントのバーチャルなストラクチャーの参照を含んだ16文字の文字列(*SVG_Ref*)を返します。この参照はコンポーネントの他のコマンドにおいて使用される必要があります。

重要: この参照が必要なくなった場合、この参照に*SVG_CLEAR* コマンドを使用してメモリーを解放するのを忘れないようにしてください。

例題

```
svgRef:=SVG_New
...
svgRef Copy:=SVG_Copy(svgRef)
```

SVG_Export_to_picture

SVG_Export_to_picture (svgObject {; exportType}) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクト参照
exportType	倍長整数	➡	0=データソースを保存しない　1(デフォルト)=データソースをコピーする　2=データソースを所有する
戻り値	ピクチャー	⮌	SVGエンジンによってレンダリングされたピクチャー

説明

SVG_Export_to_picture コマンドは、*svgObject* 引数によって参照されているSVGストラクチャーによって描画されるピクチャーを返します。

任意の*exportType* 引数を使用すると、XMLデータソースがコマンドによってどのように管理されるかを指定します。この引数についての詳細は、4Dの **SVG EXPORT TO PICTURE** コマンドの詳細を参照して下さい。この引数を省略した場合、デフォルトの値は1、Copy XML Data Sourceになります。

例題

```
svgRef:=SVG_New(500;200;Test component)
...
MyPicture:=SVG_Export_to_picture(svgRef;0)

SVG_CLEAR(svgRef)
```

SVG_Export_to_XML

SVG_Export_to_XML (svgObject) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	⇒	SVGオブジェクト参照
戻り値	テキスト	⇐	SVGドキュメントのXMLテキスト

説明

SVG_Export_to_XML コマンドは、*svgObject* 引数で参照されたSVGストラクチャーのXMLテキストの詳細を返します。

例題

```
svgRef:=SVG_New(500;200;Test component)
...
MyText:=SVG_Export_to_XML(svgRef)

SVG_CLEAR(svgRef)
```

SVG_New

SVG_New {(width ; height {; title {; description {; rectangle {; display}}}})} -> 戻り値

引数	型		説明
width	倍長整数	➡	ドキュメントの幅
height	倍長整数	➡	ドキュメントの高さ
title	文字	➡	ドキュメントのタイトル
description	文字	➡	詳細
rectangle	ブール	➡	ビューボックスを設定
display	整数	➡	ピクチャー表示のフォーマット
戻り値	SVG_Ref	⬅	SVGオブジェクト参照

説明

SVG_New コマンドは、新しいSVGドキュメントを作成し、その参照番号を返します。

任意の*width* と *height* 引数を使用すると、SVGドキュメントのサイズを制限することができます。これらの2つの引数はユーザーポイント単位であることが期待されます。それ以外の単位を指定したい場合には、*SVG_SET_DIMENSIONS* コマンドを使用する必要があります。

任意の*title* と*description* 引数を使用すると、コンテンツに関する情報を渡す事ができます。

任意の*rectangle* 引数にTrueを渡した場合、ビューボックス('viewBox' 属性)は自動的に作成されたドキュメント同じサイズに設定されます。

注: *SVG_SET_VIEWBOX* コマンドを使用すると、グラフィックビューボックスの座標を変更してピクチャーとの重なりをより細かく調整することができます。

任意の*display* 引数を使用すると、グラフィックをドキュメントのサイズに一致させるかどうかを指定する事ができます。以下の4Dピクチャー表示定数のどちらか一つを引数として渡す事ができます: Scaled to fit prop centered または Scaled to Fit

コマンドは、メモリ内のドキュメントのバーチャルなストラクチャーへの参照を表す16文字の文字列(*SVG_Ref*)を返します。この参照は、コンポーネントの他のコマンドで使用される必要があります。

重要: 必要がなくなった場合には、*SVG_CLEAR* コマンドを呼び出し、この参照を使用して占有していたメモリを開放することを忘れないで下さい。

例題

```
svgRef:=SVG_New
svgRef:=SVG_New(500;200)
svgRef:=SVG_New(900;700;"SVG component test";"This is an example";True;Scaled to fit)
```

SVG_Open_file

SVG_Open_file (path) -> 戻り値

引数	型		説明
path	文字	⇒	開きたいSVGドキュメントへのパス名
戻り値	SVG_Ref	⇐	ドキュメント参照

説明

SVG_Open_file コマンドは、*path* 引数によって指定された場所にあるドキュメントの解析(とDTDによる評価)を行い、そのドキュメントに対するSVG参照(16文字の文字列)を返します。

重要: 必要がなくなったときには、*SVG_CLEAR* コマンドを呼び出し、その参照を使用してこのドキュメントが占有していたメモリを忘れずに解放して下さい。

SVG_Open_picture

SVG_Open_picture (picture) -> 戻り値

引数	型		説明
picture	ピクチャー	➡	4Dピクチャーフィールドまたは変数
戻り値	SVG_Ref	⮌	SVGドキュメントの参照

説明

SVG_Open_picture コマンドは、SVGピクチャーを解析し、そのピクチャーのSVG参照を返します。*picture* 引数で指定したSVGにSVGピクチャーが格納されていない場合、コマンドは空の文字列を返します。

重要: 必要がなくなったときには、*SVG_CLEAR* コマンドを呼び出し、その参照を使用してこのドキュメントが占有していたメモリを忘れずに解放して下さい。

例題

```
READ PICTURE FILE("";$picture)
If(OK=1)
   $ref:=SVG_Open_picture($picture)
   ...
   SVG_CLEAR($ref)
End if
```

SVG_SAVE_AS_PICTURE

SVG_SAVE_AS_PICTURE (svgObject ; document {; codec})

引数	型		説明
svgObject	SVG_Ref	⇒	SVGオブジェクト参照
document	文字	⇒	ドキュメント名またはドキュメントへの完全なパス名
codec	文字	⇒	ピクチャーのコーデックID

説明

SVG_SAVE_AS_PICTURE コマンドは*svgObject* 引数によって指定されたSVGオブジェクトの中身を*document* 引数によって指定されたピクチャーファイルへと書き込みます。*svgObject* 引数で指定したオブジェクトがSVGドキュメントでなかった場合、エラーが生成されます。

document 引数には、ファイルへの完全なパス名か、ファイル名のみを渡す事ができます。後者の場合にはデータベースのストラクチャーファイルの隣にそのファイルが新しく作成されます。*document* 引数に空の文字列を渡した場合、標準のファイルを保存ダイアログボックスが開き、作成するファイルのファイル名、場所、そしてフォーマットを指定することができます。

任意の*codec* 引数を使用するとピクチャーを保存する際に使用するフォーマットを指定することができます。この引数が省略された場合、ピクチャーはpngフォーマットで保存されます。

SVG_SAVE_AS_PICTURE コマンドは**SVG_SET_DOCUMENT_VARIABLE**コマンドによって指定された変数の値を(あれば)変更します。

例題

```
svgRef:=SVG_New(500;200;Sales statistics)
...
SVG_SAVE_AS_PICTURE(svgRef;test.png) `保存
SVG_SAVE_AS_PICTURE(svgRef;test.gif;".gif")
SVG_CLEAR(svgRef)
```

SVG_SAVE_AS_TEXT

SVG_SAVE_AS_TEXT (svgObject {; document})

引数	型		説明
svgObject	SVG_Ref	⇒	SVGオブジェクト参照
document	文字	⇒	ドキュメント名またはドキュメントへの完全なパス名

説明

SVG_SAVE_AS_TEXT コマンドは、*svgObject* 引数によって指定されたSVGオブジェクトの中身を*document* 引数によって指定されたディスクファイルへと書き込みます。*svgObject* 引数で指定したオブジェクトがSVGドキュメントでなかった場合、エラーが生成されます。

document 引数には、ファイルへの完全なパス名か、ファイル名のみを渡す事ができます。後者の場合にはデータベースのストラクチャーファイルの隣にそのファイルが新しく作成されます。*document* 引数に空の文字列を渡した場合、標準のファイルを保存ダイアログボックスが開き、作成するファイルのファイル名、場所、そしてフォーマットを指定することができます。

SVG_SAVE_AS_TEXT コマンドは **SVG_SET_DOCUMENT_VARIABLE** コマンドによって指定された変数の値を(あれば)変更します。

例題

```
svgRef:=SVG_New(500;200;"Sales statistics")
...
SVG_SAVE_AS_TEXT(svgRef;"test.svg") //ドキュメントはストラクチャーの隣に保存されます
SVG_CLEAR(svgRef)
```

SVG_SET_DOCUMENT_VARIABLE

SVG_SET_DOCUMENT_VARIABLE (pointer)

引数	型		説明
pointer	ポインター	➡	設定する変数へのポインター

説明

SVG_SET_DOCUMENT_VARIABLE メソッドは、**SVG_SAVE_AS_PICTURE**や**SVG_SAVE_AS_TEXT**の呼び出しで更新される、ホストデータベースの変数へのポインターを設定します。セッション中で一回だけこのメソッドを呼び出します。

pointer 引数には変更を反映させたい変数へのポインターを渡します (通常 **Document** システム変数)。

pointer 引数にnilポインターを渡すことでリンクを削除できます。

SVG_Validate_file

SVG_Validate_file (path) -> 戻り値

引数	型		説明
path	文字	⇒	評価するSVGドキュメントへのパス名
戻り値	ブール	⇐	ドキュメントがDTDに対応していればTrue

説明

SVG_Validate_file コマンドは、*path* 引数によって指定されたディスク上のドキュメントを、DTD(1.0)で評価します。ドキュメントがDTD文書構造に従っている場合にはコマンドはTrueを返し、そうでない場合にはFalseを返します。

フィルター

- SVG フィルター
- SVG_Filter_Blend
- SVG_Filter_Blur
- SVG_Filter_Offset

SVG フィルター

"**フィルター**"テーマのコマンドを使用すると、SVG要素に適用可能なフィルターエフェクトを設定することができます。フィルターエフェクトはグラフィックオペレーションの継承を、 ソースグラフィックに適用されることで成り立っており、その結果として変更されたグラフィックが生成されます。

フィルターエフェクトの結果は、ターゲットとなるデバイス上の、オリジナルのソースグラフィックの場所にレンダリングされます。

フィルターは"**ストラクチャー & 定義**"テーマ内の *SVG_Define_filter* コマンドを使用することによって設定可能であり、"**属性**"テーマ内の *SVG_SET_FILTER* コマンドを使用することによって適用されます。"**フィルター**"テーマのコマンドを使用すると、フィルタリングオペレーション(または"フィルタープリミティブ")を作成することができます。

SVG_Filter_Blend

SVG_Filter_Blend (filterRef ; picture ; backgroundPict {; mode {; name}}) -> 戻り値

引数	型		説明
filterRef	SVG_Ref	➡	フィルターの参照
picture	文字	➡	ピクチャーソース
backgroundPict	文字	➡	背景のピクチャーのソース
mode	文字	➡	ミックスモード
name	文字	➡	フィルタープリミティブのターゲット
戻り値	SVG_Ref	⬅	プリミティブの参照

説明

SVG_Filter_Blend コマンドは*filterRef* 引数で指定したフィルターに対してブレンドフィルターを設定し、その参照を返します。*filterRef* がフィルターへの参照でない場合、エラーが生成されます。

このフィルターは二つのソースから成り立っています。*backgroundPict* と *picture* です。これに加え、イメージングソフトウェアによって現在使用されているミキシングモードによって決定します。

任意の*mode* 引数を使用するとブレンドに使用されるピクセルのコンビネーションモード(仕様を参照して下さい)を設定できます。使用できる値は"normal" (デフォルト値)、"multiply"、"screen"、"darken" または "lighten"のいづれかです。

任意の*name* 引数には、このフィルタープリミティブの結果に割り当てられた名前を(あれば)渡します。

注: Windowsでは、このコマンドを使用する前にDirect2Dを無効化しておく必要があります(詳細は**SET DATABASE PARAMETER** コマンドのDirect2D disabled 定数の説明を参照して下さい)。

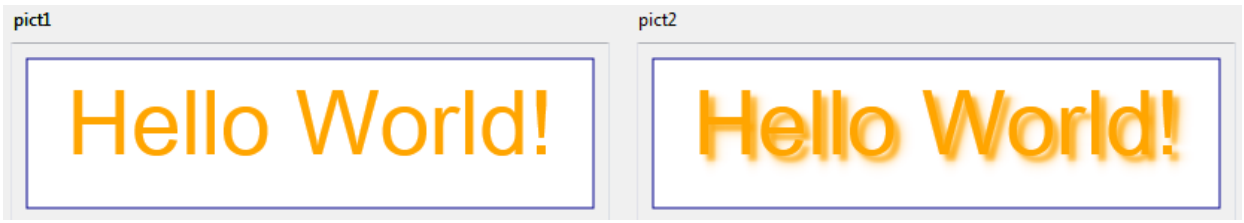
例題

フォーム内に、二つの同一のSVGピクチャーを表示し、"blend"フィルターを作成して右側のピクチャーへと割り当てます。このフィルターは"offset" と "blur"フィルターのコンビネーションです:

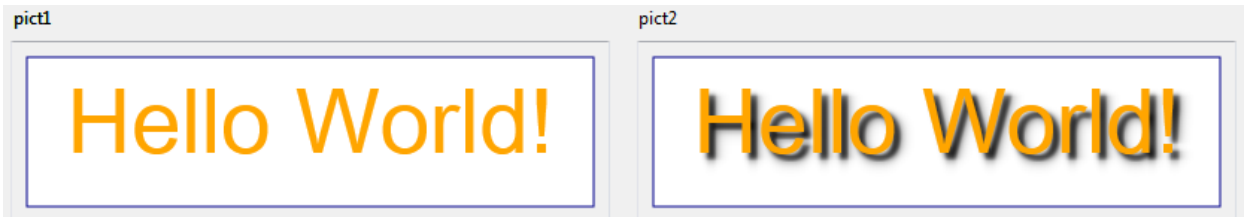
```
 $root:=SVG_New(400;400;"filters test") //最初の(左側の)ピクチャーの定義
 $rect:=SVG_New_rect($root;10;10;380;100;0;0;"darkblue";"white";1)
 SVG_SET_FILL_BRUSH($root;"orange")
 $textAreaRef:=SVG_New_textArea($root;"Hello World!";10;10;380;100;"arial";60;Normal;Align center)
 <>pict1:=SVG_Export_to_picture($root) //最初のピクチャーを表示

 $root2:=SVG_New(400;400;"filters test") //同一の(右側の)ピクチャーの定義

  //フィルターを作成
 $filter:=SVG_Define_filter($root2;"MyShadow")
 $vGraph:=True //グラフィックレイヤーに対して適用 - αレイヤーに対して適用する場合にはFalseを渡す
 If($vGraph)
    $ref1:=SVG_Filter_Blur($filter;2;"sourceGraphic";"blurResult") //"blurResult" はオフセットフィルターの"input" として使用されます。
 Else
    $ref1:=SVG_Filter_Blur($filter;2;"sourceAlpha";"blurResult") //"blurResult" はオフセットフィルターの"input" として使用されます。
 End if
  //オフセットフィルターを追加
 $ref2:=SVG_Filter_Offset($filter;5;5;"blurResult";"alphaBlurOffset")
  //ブレンドフィルターを追加
 $ref3:=SVG_Filter_Blend($filter;"sourceGraphic";"alphaBlurOffset";"normal";"finalFilter";)

 $rect2:=SVG_New_rect($root2;10;10;380;100;0;0;"darkblue";"white";1)
 SVG_SET_FILL_BRUSH($root2;"orange")
 $textAreaRef2:=SVG_New_textArea($root2;"Hello World!";10;10;380;100;"arial";60;Normal;Align center)
```

```
SVG_SET_FILTER($textAreaRef2;"MyShadow") //最終フィルターを適用
<>pict2:=SVG_Export_to_picture($root2) //二つ目のピクチャーを表示
```

結果 (blue input filter = sourceGraphic) :



結果 (blue input filter = sourceAlpha) :



SVG_Filter_Blur

SVG_Filter_Blur (filterRef ; deviation {; input {; name}}) -> 戻り値

引数	型		説明
filterRef	SVG_Ref	➡	フィルターの参照
deviation	実数	➡	ぼかし操作の標準偏差
input	文字	➡	フィルタープリミティブのソース
name	文字	➡	フィルタープリミティブのターゲット
戻り値	SVG_Ref	⬅	プリミティブの参照

説明

SVG_Filter_Blur コマンドは*filterRef* 引数で指定したフィルターに対してガウスぼかしを設定し、その参照を返します。*filterRef* 引数がフィルター参照でない場合、エラーが生成されます。

deviation 引数を使用すると、ぼかし操作に対して標準偏差を設定することができます。渡した数字が整数だった場合、同じ偏差がX軸とY軸に対して適用されます。渡し た数字に小数部分が含まれる場合、整数部分がX軸に対して適用される偏差を表し、小数部分がY軸に対して適用される偏差を表します。

任意の*input* 引数はフィルタープリミティブのグラフィックソースを指定します。以下の値を渡す事ができます:

- "sourceGraphic" を渡した場合、フィルターソースはグラフィックであることを意味します(デフォルト)。
- "sourceAlpha" を渡した場合、フィルターソースはアルファチャンネルであることを意味します。

任意の *name* 引数には、このフィルタープリミティブの結果に割り当てられた名前を(あれば)渡します。

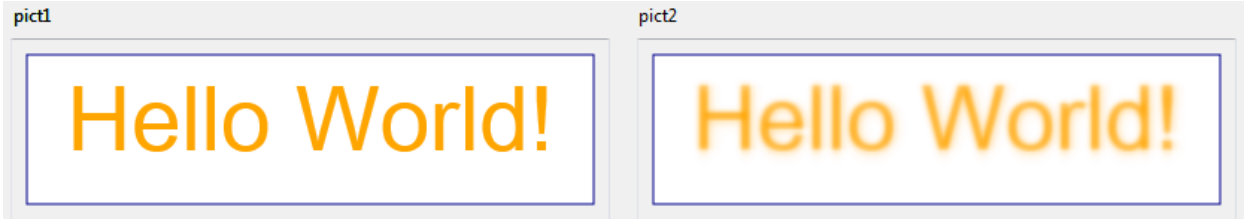
注: Windowsでは、このコマンドを使用する前にDirect2Dを無効化しておく必要があります(詳細は**SET DATABASE PARAMETER** コマンドのDirect2D disabled 定数の説明を参照して下さい)。

例題

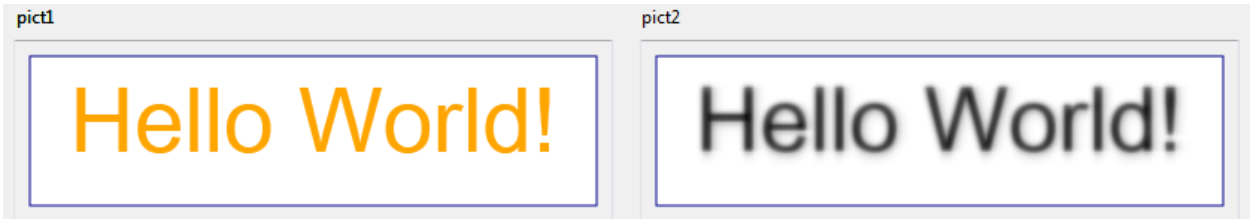
フォーム内において、二つの同一のSVGピクチャーを表示し、"blur"フィルターを作成して右の方のピクチャーへと割り当てる場合を考えます:

```
$root:=SVG_New(400;400;"filters test") //最初の(左側の)ピクチャーの定義
$rect:=SVG_New_rect($root;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH($root;"orange")
$textAreaRef:=SVG_New_textArea($root;"Hello World!";10;10;380;100;"arial";60;Normal;Align center)
<>pict1:=SVG_Export_to_picture($root) //最初のピクチャーを表示

$root2:=SVG_New(400;400;"filters test") //同一の(右側の)ピクチャーの定義

 //フィルターを作成
$filter1:=SVG_Define_filter($root2;"blur")
 // フィルター定義
$vGraph:=True //グラフィックレイヤーに適用 – アルファレイヤーに適用する場合にはFalseを渡します
If($vGraph)
   SVG_Filter_Blur($filter1;Deviation{Deviation};"sourceGraphic")
Else
   SVG_Filter_Blur($filter1;Deviation{Deviation};"sourceAlpha")
End if

$rect2:=SVG_New_rect($root2;10;10;380;100;0;0;"darkblue";"white";1) //同一の(右側の)ピクチャーの定義
SVG_SET_FILL_BRUSH($root2;"orange")
$textAreaRef2:=SVG_New_textArea($root2;"Hello World!";10;10;380;100;"arial";60;Normal;Align
center)

SVG_SET_FILTER($textAreaRef2;"blur") //フィルターを適用
<>pict2:=SVG_Export_to_picture($root2) //二つ目のピクチャーを表示
```

結果 (input = sourceGraphic):



結果 (input = sourceAlpha):



SVG_Filter_Offset

SVG_Filter_Offset (filterRef ; dx {; dy {; input {; name}}}) -> 戻り値

引数	型		説明
filterRef	SVG_Ref	➡	フィルターの参照
dx	倍長整数	➡	X軸のオフセット
dy	倍長整数	➡	Y軸のオフセット
input	文字	➡	フィルタープリミティブのソース
name	文字	➡	フィルタープリミティブのターゲット
戻り値	SVG_Ref	↩	プリミティブの参照

説明

SVG_Filter_Offset コマンドは、*filterRef* 引数で指定したフィルターのオフセットを指定し、その参照を返します。*filterRef* がフィルターの参照でない場合、エラーが生成されます。

dx 引数には水平方向のオフセットの値を渡します。

任意の*dy* 引数には垂直方向のオフセットの値を渡します。

任意の*input* 引数にはフィルタープリミティブのグラフィックソースを指定します。以下の値を渡す事ができます:

- "sourceGraphic" を渡した場合、フィルターソースはグラフィックであることを意味します(デフォルト)。
- "sourceAlpha" を渡した場合、フィルターソースはアルファチャンネルであることを意味します。

任意の*name* 引数には、このフィルタープリミティブの結果に割り当てられた名前を(あれば)渡します。

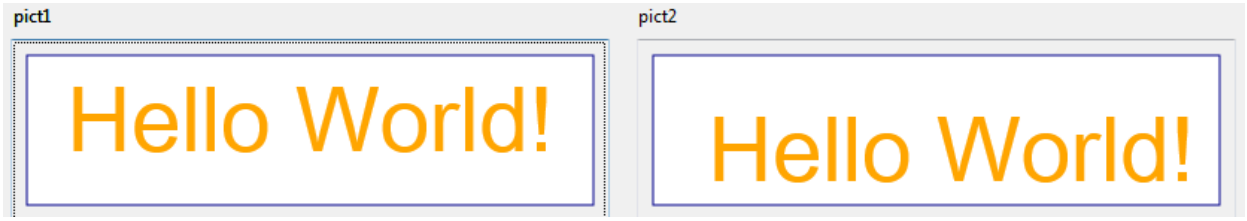
注: Windowsでは、このコマンドを使用する前にDirect2Dを無効化しておく必要があります(詳細は**SET DATABASE PARAMETER** コマンドのDirect2D disabled 定数の説明を参照して下さい。)

例題

フォーム内に、二つの同一のSVGピクチャーを表示し、"offset"フィルターを作成して右の方のピクチャーへと割り当てる場合を考えます:

```
$root:=SVG_New(400;400;"filters test") //最初の(左側の)ピクチャーの定義
$rect:=SVG_New_rect($root;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH($root;"orange")
$textAreaRef:=SVG_New_textArea($root;"Hello World!";10;10;380;100;"arial";60;Normal;Align center)
<>pict1:=SVG_Export_to_picture($root) //最初のピクチャーを表示

$root2:=SVG_New(400;400;"filters test") //同一の(右側の)ピクチャーを定義
$rect2:=SVG_New_rect($root2;10;10;380;100;0;0;"darkblue";"white";1)
SVG_SET_FILL_BRUSH($root2;"orange")
$textAreaRef2:=SVG_New_textArea($root2;"Hello World!";10;10;380;100;"arial";60;Normal;Align
center)

$filter:=SVG_Define_filter($root2;"Offset") //フィルターを作成
SVG_Filter_Offset($filter;10;20)
SVG_SET_FILTER($textAreaRef2;"Offset") //フィルターを適用

<>pict2:=SVG_Export_to_picture($root2) //二つ目のピクチャーを表示
```

結果:


pict1
Hello World!
pict2
Hello World!

ユーティリティ

- SVG_ABOUT
- SVG_Count_elements
- SVG_ELEMENTS_TO_ARRAYS
- SVG_Estimate_weight
- SVG_Find_ID
- SVG_Get_options
- SVG_Get_root_reference
- SVG_Get_version
- SVG_Is_reference_valid
- SVG_Post_comment
- SVG_Read_element_type
- SVG_Read_last_error
- SVG_References_array
- SVG_ROTATION_CENTERED
- SVG_SCALING_CENTERED
- SVG_Set_error_handler
- SVG_SET_OPTIONS
- SVGTool_SET_VIEWER_CALLBACK
- SVGTool_SHOW_IN_VIEWER

SVG_ABOUT

SVG_ABOUT
このコマンドは引数を必要としません

説明

SVG_ABOUTコマンドはコンポーネントのバージョン番号を確認できるダイアログを表示します:

SVG_Count_elements

SVG_Count_elements (svgObject) -> Function result

引数	型		説明
svgObject	SVG_Ref	➡	SVG参照
Function result	倍長整数	⮌	オブジェクト数

説明

SVG_Count_elements コマンドは、*svgObject* 引数内に渡されたオブジェクト内に存在するグラフィックオブジェクトの数を返します。グループは一つのオブジェクトとしてカウントされます。グループ内のグラフィックオブジェクト数を調べたい場合には、そのグループの参照をコマンドに渡します。*svgObject* が有効でない場合には、エラーが生成されます。

SVG_ELEMENTS_TO_ARRAYS

SVG_ELEMENTS_TO_ARRAYS (svgObject ; refsArrayPointer {; typesArrayPointer {; namesArrayPointer}})

引数	型		説明
svgObject	文字	➡	SVG参照
refsArrayPointer	ポインター	➡	オブジェクト参照の文字列配列
typesArrayPointer	ポインター	➡	オブジェクトの型の文字列配列
namesArrayPointer	ポインター	➡	オブジェクトIDの文字列配列

説明

SVG_ELEMENTS_TO_ARRAYS コマンドは、*svgObject* 引数に渡されたSVG参照の第1階層にあるグラフィックオブジェクトの参照を*refsArrayPointer* 引数で指定した配列に入ります。

任意の*typesArrayPointer* ポインターを渡した場合、その配列にはオブジェクトの型が入ります。

任意の*namesArrayPointer* ポインターを渡した場合、その配列にはオブジェクトIDが入ります。

グループは一つのオブジェクトとしてカウントされます。グループ内のグラフィックオブジェクトの情報を取得したい場合、その参照をコマンドへと渡します。

svgObject 引数が無効だった場合、またはその属性が存在しない場合、エラーが生成されます。

SVG_Estimate_weight

SVG_Estimate_weight (svgObject) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	➡	SVGドキュメントの参照
戻り値	実数	⮌	SVGドキュメントのサイズ(バイト単位)

説明

SVG_Estimate_weight コマンドは*svgObject* 引数に参照を渡したSVGツリーのサイズ(バイト単位)を返します。*svgObject* が有効な参照ではない場合、エラーが生成されます。

SVG_Find_ID

SVG_Find_ID (svgObject ; name) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクトの参照
name	文字	➡	SVG要素のID
戻り値	SVG_Ref	↩	要素の参照

説明

SVG_Find_ID コマンドは、*svgObject* 内にルート要素を渡したSVGストラクチャー内に所属する要素について、*name* 引数にIDを渡した要素の参照を返します。
要素が見つからない場合、エラーが生成されます。

SVG_Get_options

SVG_Get_options -> 戻り値

引数	型		説明
戻り値	倍長整数		オプション

説明

SVG_Get_options コマンドは、それぞれのbitがコンポーネントのオプションを表す、32-bitの配列を示す倍長整数を返します。4D bitの演算子を用いることによって、オプションの状態のチェック(??)、有効化(?+)、無効化(?-)ができます。

現在利用可能なオプションは以下の通りです:

ビット	オプション	デフォルト値
1	要素作成時に自動的にIDを割り当て	0 (無効化)
2	可能なオブジェクトは全て自動的に閉じる	0 (無効化)
3	オブジェクトを背景付きで作成	1 (有効化)
4	パスで絶対座標を使用	1 (有効化)
5	より読みやすいコードを作成	0 (無効化)
6	エラー発生時にビープ音を鳴らす	1 (有効化)
7	4Dエラーを表示しない	0 (無効化)
8	透明なピクチャー	1 (有効化)
9	三角法の原点を使用	0 (無効化)
10	自動的にArial を置き換える	1 (有効化)
11	新しいキャンバスにおいてshape-rendering='crispEdges' をデフォルトとして設定	0 (無効化)
12	引数をチェックする	1 (有効化)
13	余分な空白を削除しない	0 (無効化)
14	オブジェクトの中心を軸にして回転	0 (無効化)

- 要素作成時に自動的に*ID*を割り当て
  このオプションが有効化されていると、コンポーネントが新しい要素を作成したときに、作成されたオブジェクトに対して'id'属性をシステム的に追加して入力します(この属性が指定されていない場合に限る)。

- 自動的にオブジェクトを閉じる
  このオプションが有効化されていると、*SVG_New_arc* と *SVG_New_polyline_by_arrays* コマンドで作成されたオブジェクトは自動的に閉じられます。

- オブジェクトを背景付きで作成
  このオプションが有効化されているとき、閉じられたオブジェクトは背景色つきで作成されます。そうでない場合には、背景は透明になります。

- パスで絶対座標を使用
  SVG_PATH_MOVE_TO、*SVG_PATH_LINE_TO*、*SVG_PATH_CURVE* と *SVG_PATH_ARC* コマンドを使用してパスを描画するとき、このオプションが有効化されていると渡されたパスは絶対座標として解釈されます。そうでない場合には、座標は相対座標として解釈されます。

- より読みやすいコードを作成
  このオプションを使用すると、インデントされ、スペースが空いた、それでいて広すぎないコードが作成されます。このオプションは、特にデバッグフェーズにおいて有効です。

- エラー発生時にビープ音を鳴らす
  エラーが発生し、ホストデータベース側で*SVG_Set_error_handler* コマンドを使用したエラーハンドリングメソッドが実装されていないとき、このオプションが有効化されているとビープ音が発生します。

- *4D*エラーを表示しない
  このオプション(デフォルトで有効化)は、コンポーネント独自のエラーハンドリングメソッドを実装することによって4Dエラーの表示をブロックします。場合によってはこの内部管理を使用せず、そういった4Dのメッセージの表示を許可したいこともあるかもしれません。これは例えば、デバッグの途中などでは有効でしょう。

- 透明なピクチャー
  デフォルトでは、*SVG_New* コマンドを使用して作成されたSVGピクチャーは透明です。このオプションを無効化することによって、ピクチャーの背景は白になります。

- 三角法の原点を使用
  デフォルトで、SVGは原点を、上の角度(0時方向)に原点を持ってきます。このオプションを使用すると、通常の三角法の参照点(3時方向または15分方向)を原点として座標を渡す事ができるようになります。変換は実行中に行われます。

- 自動的に*Arial* を置き換える
  デフォルトでは、4D SVGは非ローマ文字(日本語など)との互換性を向上させるために、'Arial' フォントを 'Arial Unicode MS' で置き換えますが、特定の場合においてはこの機能を無効化したいこともあるでしょう。このオプションはArialフォントを置き換えなくてもよいという事を意味しています。

- 新しいキャンバスにおいて*shape-rendering='crispEdges'* をデフォルトとして設定
  このオプションを使用すると、crispEdges 属性(**SVG_SET_SHAPE_RENDERING** を参照の事)をデフォルトとして強制的に使用することができます。

- 引数をチェックする
  デフォルトでは、4D SVGはコマンドに渡された引数の有効性をチェックします。開発ステップが完了したら、コードの実行速度を上げるためにこのオプションは無効化する方が賢明でしょう。

- 余分な空白を削除しない(v14からの新機能)
  テキストオブジェクトにおいて隣り合った複数の空白の表示を許可します。

- オブジェクトの中心を軸にして回転(v14からの新機能)
  このオプションが有効化されているとき、*SVG_SET_TRANSFORM_ROTATE* コマンドは、第3と第4引数が省略されている場合にオブジェクトの中心を軸に回転を実行します。回転の中心はオブジェクトのx軸座標、y軸座標、そして高さと幅の属性を元に計算されます。参照しているオブジェクトがこれらの属性を持っていない場合、回転は(0,0)の点を中心にして実行されます。

例題

SVG_SET_OPTIONS コマンドを参照して下さい。

SVG_Get_root_reference

SVG_Get_root_reference (svgObject) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	⇒	SVGオブジェクト参照
戻り値	SVG_Ref	⇐	ルートSVG要素

説明

SVG_Get_root_reference は*svgObject* 引数に渡した参照を持つSVGオブジェクトのルート要素参照を返します。

SVG_Get_version

SVG_Get_version -> 戻り値

引数	型		説明
戻り値	文字		バージョン番号

説明

SVG_Get_version コマンドはコンポーネントのバージョンを文字と番号を組み合わせた形式で返します。返された文字列は常にバージョン番号とサブバージョン番号を含みます(ヴァージョン11であれば"11.0" を、ヴァージョン11の3つ目のアップデートであれば"11.3" を返します)。Betaバージョンであった場合、頭に”B”が付いた配布番号が指定されます(ヴァージョン11.3のBeta 1であれば"11.3B1"が返されます)。

SVG_Is_reference_valid

SVG_Is_reference_valid (svgObject) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	⇒	SVG要素の参照
戻り値	ブール	⇐	参照がSVG要素に所属する場合はTrue

説明

SVG_Is_reference_valid コマンドは、*svgObject* 引数に参照が渡されているオブジェクトがSVGツリーの要素である場合に**True** を返します。要素がSVGツリーに所属していない場合には、コマンドは**False** を返します。*svgObject* に渡した参照が有効ではなかった場合、エラーが生成されます。

SVG_Post_comment

SVG_Post_comment (svgObject ; comment) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクト参照
comment	テキスト	➡	コメントとして追加するテキスト
戻り値	SVG_Ref	⮌	コメントの参照

説明

SVG_Post_comment は*svgObject* で指定したSVGオブジェクトにXMLコメントを追加します。
メソッドはコメントのSVG参照を返します。

例題

以下のコードは:

```
C_TEXT($comment)
$comment:="Modified on "+String(Current date)
$ref:=SVG_Post_comment($svg;$comment )
```

$svg SVGオブジェクトに以下のコードを追加します:

```
<!--Modified on 12/12/2011-->
```

SVG_Read_element_type

SVG_Read_element_type (svgObject) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	⇒	SVG要素の参照
戻り値		⮌	要素の型

説明

SVG_Read_element_type コマンドは*svgObject* 引数に参照を渡した要素の型を返します。

svgObject 引数に渡した参照が有効な参照ではなかった場合、またはこの属性が存在しない場合には、エラーが生成されます。

SVG_Read_last_error

SVG_Read_last_error -> Function result

引数	型		説明
Function result	倍長整数		最後に置きたエラーの番号

説明

SVG_Read_last_error コマンドは4D SVG コンポーネントコマンドの実行中に最後に発生したエラーの番号を返し、そのエラーをリセットします。

返されるエラー番号はコンポーネントコマンド特有のものの場合と、4Dによって生成されたエラーの場合があります。以下のエラーは、コンポーネントで生成されるものです:

8850	引数が不足しています
8851	無効な引数の型です
8852	無効な参照です
8853	属性に対して不正な値です
8854	この要素はそのコマンドを受け付けません
8855	無効な(IDがドキュメントにない)オブジェクト名(シンボル、マーカー、フィルター、等)です
8856	DTD ファイルが見つかりません
8857	引数に対して不正な値です
8858	未知のエラーです

例題 1

SVG_Set_error_handler コマンドの例題の"SVG_error_mgmt" メソッドを元に、以下の様な場合を考えます:

```
 `エラーハンドリングメソッドの実装
$ Error_Method_Txt:=SVG_Set_error_handler("SVG_error_mgmt")
 `これ以降、エラーの場合に実行されるメソッドは SVG_error_mgmt メソッドとなります

 `新規SVGドキュメントの作成
$SVG:=SVG_New(1200;900;"SVG Component Test";"";True)
SVG_SET_VIEWBOX($SVG;0;0;1500;1000)

If(SVG_Read_last_error=0)

   ...

Else
 `SVG_error_mgmt メソッドが呼び出され、エラー番号を受け取ります
End if

 `エラーハンドリングメソッドの実装を解除
SVG_Set_error_handler
```

例題 2

以下の***SVG_error_mgmt*** メソッドがあった場合:

```
C_LONGINT($1)
C_TEXT($2)

 `エラーとコンテキストを保存する
```

```
errorNumber:=$1
commandName:=$2

 `OKシステム変数を0に設定する
OK:=0
```

このメソッドを以下の様に使う事ができます:

```
 ` エラーハンドリングメソッドの実装
$ Error_Method_Txt:=SVG_Set_error_handler("SVG_error_mgmt")

 ` 新規SVGドキュメントの作成
$SVG:=SVG_New(1200;900;" SVG Component Test ";"";True)
SVG_SET_VIEWBOX($SVG;0;0;1500;1000)
If(OK=1)

   ...

Else
   ALERT("Error No."+String(errorNumber)+" during execution of the command \""+commandName+"\"")
End if

 ` エラーハンドリングメソッドの実装を解除
SVG_Set_error_handler
```

SVG_References_array

SVG_References_array (refsArrayPointer) -> 戻り値

引数	型		説明
refsArrayPointer	ポインター	➡	ドキュメント参照の文字配列
戻り値	倍長整数	⮌	参照の数

説明

SVG_References_array コマンドはカレントのSVGドキュメントの一覧を*refsArrayPointer* 引数で指定してる配列内に返します。結果として、コマンドは見つかった参照の数を返すことになります。

SVG_References_array コマンドはデバッグ時に有用です。*SVG_New*、*SVG_Copy* または *SVG_Open_file* コンポーネントコマンドを使用してSVGドキュメントが作成される度に、コンポーネントはコマンドから返された参照を内部的な配列に追加します。*SVG_CLEAR* コマンドを使用してSVGドキュメントがリリースされたときには、コンポーネントはその参照を配列から削除します。

SVG_ROTATION_CENTERED

SVG_ROTATION_CENTERED (svgObject ; angle)

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクト参照
angle	実数	➡	回転の角度

説明

SVG_ROTATION_CENTERED コマンドは*svgObject* p引数に参照が渡されたSVGオブジェクトに対して中心点を軸とした回転を行います。この回転は、*x*、*y*、*width* そして *height* 属性をもつオブジェクトに対してのみ適用可能です。

angle 引数には、実行したい回転の角度を渡します。

SVG_SCALING_CENTERED

SVG_SCALING_CENTERED (svgObject ; scale {; x ; y})

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクト参照
scale	実数	➡	スケーリングの値
x	実数	➡	X軸
y	実数	➡	Y軸

説明

SVG_SCALING_CENTERED コマンドは、*svgObject* 引数に参照が渡されたSVG画像のセンタースケーリングを実行します。

scale 引数には、正のスケール値(>1)を渡します。1を渡した場合、縮尺は100%に設定されます。

任意の*x* と *y* 引数にはそれぞれ中心点のx-軸とy-軸の座標を渡します。これらの引数を渡さなかった場合、中心点は、オブジェクトの"x"、"y"、"width" そして "height"属性に基づいて決定されます。もしこの変形がこれらの属性をもたないオブジェクトに対して適用され、任意の*x* と *y* の引数が省略されていた場合、空の文字列が返されます。

SVG_Set_error_handler

SVG_Set_error_handler {(method)} -> 戻り値

引数	型		説明
method	文字	➡	実装するメソッド名
戻り値	文字	⮌	以前実装していたメソッド名

説明

SVG_Set_error_handler コマンドを使用すると、*method* 引数で指定したホストデータベースメソッドを、エラーが起きた際に呼び出されるメソッドとして実装し、以前実装していたメソッドの名前を返します。

このコンポーネントコマンドは、呼び出される際に最小限の認証を必要とします。コマンドが適用される要素に対して最小限の数の引数、参照の認証です。そのためコンポーネントはエラーを構造化された状態で管理し、ホストデータベースがどんなエラーでも取得できるようにします。

デフォルトの機能が変更されていない場合、エラーが発生した場合にはビープ音が鳴り、コマンドは中断されます。

ホストデータベースはこれらのメソッドを通して、エラー番号とエラーを起こしたコマンド名を取得することができます。これをするためにはホストデータベース側で、エラー番号を第一引数として、コマンド名を第二引数として受け取るメソッドを作成する必要があります。

SVG_Set_error_handler コマンドによってメソッドが実装されていた場合、このメソッドはエラーが発生した際に呼び出されます。このとき、コンポーネントのコードによってビープ音が鳴ることはありません。

method 引数が省略されている場合、または空の文字列が渡された場合、メソッドの実装は解除され、振る舞いはデフォルトの振る舞いへと戻ります。

注: コンポーネントから呼び出されるホストデータベースメソッドは、「コンポーネントとホストデータベース間で共有」プロパティにチェックがされている必要があります。

例題

SVG_error_mgmt メソッド(ホストデータベースメソッド)をエラーハンドリングメソッドとして実装する場合を考えます:

```
$error:=SVG_Set_error_handler("SVG_error_mgmt")
```

メソッドコード:

```
` SVG_error_mgmt メソッド
ALERT("Error No."+String($1)+" during execution of the command \""+$2+"\"")
```

SVG_SET_OPTIONS

SVG_SET_OPTIONS {(options)}

引数	型		説明
options	倍長整数	➡	4D SVG コンポーネントオプション

説明

SVG_SET_OPTIONS コマンドを使用すると、*options* 引数の倍長整数を使用することによって、4D SVGコンポーネントのオプションを設定することができます。*options* 引数のコンテンツについてのより詳細な情報に関しては、*SVG_Get_options* コマンドの詳細を参照して下さい。

全てのオプションが一度に設定されてしまうため、このコマンドを使用する前に*SVG_Get_options* コマンドを、使用した後に4Dの を使用する必要があります。

options 引数が渡されていない場合、全てのオプションがデフォルトの値へとリセットされます(*SVG_Get_options* コマンドを参照して下さい)。

例題 1

読みやすいコードを作成する場合を考えます:

```
$Options :=SVG_Get_options
$Options :=$Options ?+5 `オプションを有効化
SVG_SET_OPTIONS($Options)
```

例題 2

パイチャートのダイアグラムを描画する場合を考えます:

```
$svg:=SVG_New

 `オブジェクトを自動で閉じるオプションを有効化
SVG_SET_OPTIONS(SVG_Get_options ?+ 2)

SVG_New_arc($svg;100;100;90;0;105;"gray";"lightcoral";1)
SVG_New_arc($svg;100;100;90;105;138;"gray";"lightskyblue";1)
SVG_New_arc($svg;100;100;90;138;230;"gray";"lightgreen";1)
SVG_New_arc($svg;100;100;90;230;270;"gray";"lightsteelblue";1)
SVG_New_arc($svg;100;100;90;270;360;"gray";"lightyellow";1)
```

例題 3

不要な空白を削除しない オプション(13)を使用してテキストオブジェクトで複数の空白を表示(v14にて追加):

```
$Txt_buffer:="abc      def"
$Dom_text:=SVG_New_textArea($Dom_svg;$Txt_buffer;50;50)
```

これは"abc def" として表示されます。

```
SVG_SET_OPTIONS(SVG_Get_options?+13) // テキストオブジェクト内で空白を削除しない
$Txt_buffer:="abc      def"
$Dom_text:=SVG_New_textArea($Dom_svg;$Txt_buffer;50;50)
```

これは"abc　　def"として表示されます。

SVGTool_SET_VIEWER_CALLBACK

SVGTool_SET_VIEWER_CALLBACK (methodName)

引数	型		説明
methodName	テキスト	➡	4Dプロジェクトメソッドの名前

説明

SVGTool_SET_VIEWER_CALLBACKコマンドを使用すると、**SVGTool_SHOW_IN_VIEWER**コマンドで表示された画像上で発生するOn ClickedとOn Mouse Moveイベント時に呼び出されるプロジェクトメソッドとして、*methodName*を登録できます。

このメソッドはクリックされた、またはマウスが重なっている要素の (4Dの**SVG Find element ID by coordinates**コマンドで返される) IDを、テキスト引数として受け取ります。

この引数をホストデータベースの*methodName*プロジェクトメソッド内で宣言しなければなりません。これを行うには

```
C_TEXT($1)
```

を記述します。

methodName メソッドには"コンポーネントとホストデータベースで共有する"メソッドプロパティを選択していなければなりません。

メソッドが存在しないか共有されていない場合、4Dはエラー -10508 を生成します。

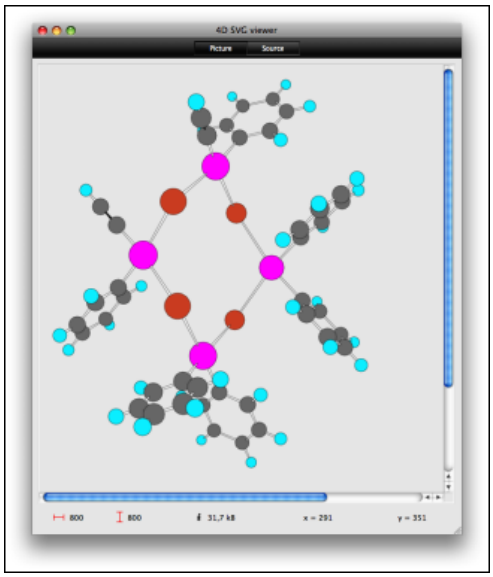
SVGTool_SHOW_IN_VIEWER

SVGTool_SHOW_IN_VIEWER (svgObject {; sources})

引数	型		説明
svgObject	SVG_Ref	➡	表示するピクチャーの参照
sources	文字	➡	ビューアーをソースページに直接開く

説明

SVGTool_SHOW_IN_VIEWER コマンドは*svgObject* 引数で指定したSVGピクチャーを、SVGビューアーウィンドウ内に表示します。このツールはSVGコンポーネントにて提供されているものです:



任意の*sources* 引数(v14から追加)を渡した場合、ビューアーはソースページ上に直接開かれます。

SVGビューアーについてのより詳細な情報に関しては、 の章を参照して下さい。

属性

- SVG_ADD_NAMESPACE
- SVG_GET_ATTRIBUTES
- SVG_Get_class
- SVG_Get_fill_brush
- SVG_Get_ID
- SVG_SET_ATTRIBUTES
- SVG_SET_ATTRIBUTES_BY_ARRAYS
- SVG_SET_CLASS
- SVG_SET_CLIP_PATH
- SVG_SET_DIMENSIONS
- SVG_SET_FILL_BRUSH
- SVG_SET_FILL_RULE
- SVG_SET_FILTER
- SVG_SET_ID
- SVG_SET_MARKER
- SVG_SET_OPACITY
- SVG_SET_ROUNDING_RECT
- SVG_SET_SHAPE_RENDERING
- SVG_SET_STROKE_BRUSH
- SVG_SET_STROKE_DASHARRAY
- SVG_SET_STROKE_LINECAP
- SVG_SET_STROKE_LINEJOIN
- SVG_SET_STROKE_MITERLIMIT
- SVG_SET_STROKE_WIDTH
- SVG_SET_TRANSFORM_FLIP
- SVG_SET_TRANSFORM_MATRIX
- SVG_SET_TRANSFORM_ROTATE
- SVG_SET_TRANSFORM_SCALE
- SVG_SET_TRANSFORM_SKEW
- SVG_SET_TRANSFORM_TRANSLATE
- SVG_SET_VIEWBOX
- SVG_SET_VIEWPORT_FILL
- SVG_SET_VISIBILITY
- SVG_SET_XY

SVG_ADD_NAMESPACE

SVG_ADD_NAMESPACE (svgObject ; prefix {; URI})

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクト参照
prefix	テキスト	➡	名前空間の接頭辞
URI	テキスト	➡	名前空間のURI

説明

SVG_ADD_NAMESPACEメソッドは、*svgObject* 引数で指定したSVGオブジェクトのDOMツリーのルートにXML名前空間属性を追加します。このメソッドを使用して、特に、SVGコードの一部に名前空間を追加できます。

prefix には名前空間属性の接頭辞に指定する文字列を渡します。以下の定数を使用できます:

- "svgNS" : 標準SVG名前空間 (http://www.w3.org/2000/svg)
- "xlinkNS" : 標準XLink名前空間 (http://www.w3.org/1999/xlink)

これらの定数を指定した場合、*URI*引数を省略できます。

prefix 引数にカスタム名前空間の接頭辞を渡すこともできます。この場合 *URI* 引数は必須であり、省略するとエラーが生成されます。

例題

以下のコードを実行すると:

```
SVG_ADD_NAMESPACE($svgRef;"svgNS")
```

SVGオブジェクトのルートに以下のコードが追加されます:

```
<xmlns="http://www.w3.org/2000/svg">
```

SVG_GET_ATTRIBUTES

SVG_GET_ATTRIBUTES (svgObject ; namesArrayPointer ; valuesArrayPointer)

引数	型		説明
svgObject	SVG_Ref	⇒	SVG 参照
namesArrayPointer	ポインター	⇒	属性名の文字配列
valuesArrayPointer	ポインター	⇒	属性の値の文字配列

説明

SVG_GET_ATTRIBUTES コマンドは、 *svgObject* 引数で参照を指定した要素の、属性の名前とその値を、それぞれ *namesArrayPointer* と *valuesArrayPointer* で指定した配列に返します。 *svgObject* が無効、もしくはその属性が存在しない場合にはエラーが生成されます。

SVG_Get_class

SVG_Get_class (svgObject {; classNames}) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	➡	SVGオブジェクト参照
classNames	ポインター	➡	クラス名の配列へのポインター
戻り値	テキスト	⮌	クラス名

説明

SVG_Get_class コマンドは、*svgObject* 引数に参照が渡されたSVG画像のクラス名を返します。クラス名は文字列で返され、クラス名が複数の場合は、クラス名がスペースで区切られた文字列で返されます。

任意の*classNames* 引数には、返されるクラス名を格納するための配列へのポインターを渡す事ができます。この引数を渡さなかった場合、クラス名は、それぞれのクラス名がスペースで区切られた文字列で返されます。

例題

```
 // 2つのスタイルを定義
SVG_Define_style($Dom_SVG;".colored {fill: yellow; fill-opacity: 0.6; stroke: red; stroke-width:
8; stroke-opacity: 0.6}")
SVG_Define_style($Dom_SVG;".blue {fill: blue}")

 // グループを作成しデフォルトのスタイルを設定
$Dom_g:=SVG_New_group($Dom_SVG)
SVG_SET_CLASS($Dom_g;"colored blue")

ARRAY TEXT($tTxt_Classes;0)
$Txt_buffer:=SVG_Get_class($Dom_g;->$tTxt_classes)

 // $Txt_buffer = "colored blue"
 // $tTxt_classes{1} = "colored"
 // $tTxt_classes{2} = "blue"
```

SVG_Get_fill_brush

SVG_Get_fill_brush (svgObject) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	⇒	SVGオブジェクト参照
戻り値	テキスト	⇒	fillの色

説明

SVG_Get_fill_brush コマンドは、*svgObject* p引数に参照が渡されたSVG画像のfillの色を返します。もし*svgObject* が"fill"属性を持っていない場合、コマンドは空の文字列を返します。

SVG_Get_ID

SVG_Get_ID (svgObject) -> 戻り値

引数	型		説明
svgObject	SVG_Ref	⇒	SVG要素の参照
戻り値	文字	⇐	要素名

説明

SVG_Get_ID commandコマンドは、*svgObject* 引数に参照を渡した要素の'id'属性の値を取得します。*svgObject* 引数に渡された参照が無効、またはその属性が存在しない場合にはエラーが生成されます。

SVG_SET_ATTRIBUTES

SVG_SET_ATTRIBUTES (svgObject ; attributeName ; attributeValue {; attributeName2 ; attributeValue2 ; ... ; attributeNameN ; attributeValueN})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
attributeName	文字	➡	設定する属性の名前
attributeValue	文字	➡	属性の値

説明

SVG_SET_ATTRIBUTES コマンドは、*svgObject* 参照で指定したSVG オブジェクトの、一つまたは複数の属性の値を設定するのに使用できます。もしこれらの属性の一つまたは複数の属性が既に存在している場合、それらの値は引数として渡された値で上書きされます。

属性とその値は、ペア要素として渡されます。

例題

```
$svg:=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2)
SVG_SET_ATTRIBUTES($object;"style";"fill:red; stroke:blue; stroke-width:3")
```

SVG_SET_ATTRIBUTES_BY_ARRAYS

SVG_SET_ATTRIBUTES_BY_ARRAYS (svgObject ; namesArrayPointer ; valuesArrayPointer)

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
namesArrayPointer	ポインター	➡	属性名
valuesArrayPointer	ポインター	➡	属性に同期した値

説明

SVG_SET_ATTRIBUTES_BY_ARRAYS コマンドは、*svgObject* 参照で指定したSVGオブジェクトの、一つまたは複数の属性の値を設定するのに使用することができます。もしこれらの属性の一つまたは複数の属性が既に存在している場合、それらの値は引数として渡された値で上書きされます。

属性とその値は、*namesArrayPointer* と *valuesArrayPointer* ポインターで指定された二つの配列を使用して渡されます。

例題

```
$svg:=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2)
ARRAY TEXT($attributes;0)
ARRAY TEXT($values;0)
APPEND TO ARRAY($attributes;"fill")
APPEND TO ARRAY($values;"red")
APPEND TO ARRAY($attributes;"stroke")
APPEND TO ARRAY($values;"blue")
APPEND TO ARRAY($attributes;"stroke-width")
APPEND TO ARRAY($values;"3")
SVG_SET_ATTRIBUTES_BY_ARRAYS($object;->$attributes;->$values)
```

SVG_SET_CLASS

SVG_SET_CLASS (svgObject ; class)

引数	型		説明
svgObject	SVG_Ref	➡	SVG要素の参照
class	テキスト	➡	class名

説明

SVG_SET_CLASSは*svgObject*に渡したオブジェクトに*class*クラスを設定します。*svgObject* が有効な参照でない場合エラーが生成されます。

参照: http://www.w3.org/TR/SVG/styling.html#ClassAttribute

例題

SVG_Define_styleコマンドの例題参照。

SVG_SET_CLIP_PATH

SVG_SET_CLIP_PATH (svgObject ; clipPathID)

引数	型		説明
svgObject	SVG_Ref	➡	SVG要素の参照
clipPathID	テキスト	➡	クリップパスの名前

説明

SVG_SET_CLIP_PATHコマンドは*svgObject*に渡したオブジェクトに*clipPathID*という名前のクリップパスを設定します。*svgObject*が有効な参照でない場合、またはクリップパスが設定されていない場合、エラーが生成されます。

参照: http://www.w3.org/TR/2001/REC-SVG-20010904/masking.html#EstablishingANewClippingPath

例題 1

円形のクリップを定義し、画像に割り当てます:

```
 // 円形のクリップパスを定義
$Dom_clipPath:=SVG_Define_clip_path($Dom_SVG;"theClip")
$Dom_circle:=SVG_New_circle($Dom_clipPath;150;100;100)

 // グループを作成
$Dom_g:=SVG_New_group($Dom_SVG)

 // 画像を挿入
$Txt_path:=Get 4D folder(6)+"logo.svg"
READ PICTURE FILE($Txt_path;$Pic_buffer)
$Dom_picture:=SVG_New_embedded_image($Dom_g;$Pic_buffer)
SVG_SET_ID($Dom_picture;"MyPicture")

 // グループにクリップパスを適用
SVG_SET_CLIP_PATH($Dom_g;"theClip")
```

例題 2

同じ画像に角の丸い四角のクリップパスを適用:


```
 // 四角のクリップパスを定義
$Dom_clipPath:=SVG_Define_clip_path($Dom_SVG;"theClip")
$Dom_rect:=SVG_New_rect($Dom_clipPath;5;10;320;240;10;10)

 // グループを作成
$Dom_g:=SVG_New_group($Dom_SVG)

 // 画像を挿入
$Txt_path:=Get 4D folder(6)+"logo.svg"
READ PICTURE FILE($Txt_path;$Pic_buffer)
$Dom_picture:=SVG_New_embedded_image($Dom_g;$Pic_buffer)
SVG_SET_ID($Dom_picture;"MyPicture")

 // グループにクリップパスを適用
SVG_SET_CLIP_PATH($Dom_g;"theClip")
```

SVG_SET_DIMENSIONS

SVG_SET_DIMENSIONS (svgObject ; width {; height {; unit}})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
width	倍長整数	➡	X 軸の寸法
height	倍長整数	➡	Y 軸の寸法
unit	文字	➡	サイズの単位

説明

SVG_SET_DIMENSIONS commandコマンドは、*svgObject* reference.参照で指定したSVG オブジェクトの寸法を設定するのに使用することができます。もしこれらの属性が既に存在している場合、それらの値は引数として渡された値で上書きされます。

unit 引数は、渡した場合にのみ使用されます。使用可能な値は、px、pt、pc、cm、mm、in、em、ex または % です。*unit* の値が不正な場合はエラーが生成されます。この引数が省略された場合、 *width* と *height* 引数の値はユーザー座標システムに則った値とみなされます。

例題

```
$svg :=SVG_New ` 新しいドキュメントを作成
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2)
SVG_SET_DIMENSIONS($object;-1;400) `新しい高さ
```

SVG_SET_FILL_BRUSH

SVG_SET_FILL_BRUSH (svgObject ; color)

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
color	文字	➡	カラー式

説明

SVG_SET_FILL_BRUSH commandコマンドは、*svgObject* 参照で指定したSVG オブジェクトのフィルカラーを指定するのに使用することができます。もしこれらの属性が既に存在している場合、それらの値は引数として渡された値で上書きされます。

カラーについての詳細は、“” セクションを参照して下さい。

例題

```
$svg:=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2)
SVG_SET_FILL_BRUSH($object;"blue")
```

SVG_SET_FILL_RULE

SVG_SET_FILL_RULE (svgObject ; fillRule)

引数	型		説明
svgObject	SVG_Ref	➡	SVG要素の参照
fillRule	テキスト	➡	オブジェクトの塗りつぶしモード

説明

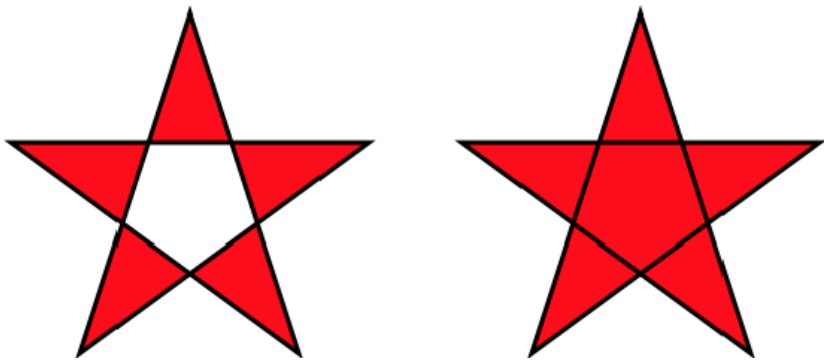
SVG_SET_FILL_RULEコマンドを使用して、*svgObject*で指定したSVGオブジェクトに塗りつぶしルールを適用できます。*svgObject*が有効な参照でない場合エラーが生成されます。

fillRule 引数は以下のいずれかでなければなりません: "nonzero", "evenodd" または "inherit"。これら以外の場合エラーが生成されます。

参照: http://www.w3.org/TR/SVG/painting.html#FillRuleProperty

例題

塗りつぶしモードのイラスト:



```
 // 'evenodd'塗りつぶしルールでパスを作成
$Dom_path:=SVG_New_path($Dom_SVG;250;75)
SVG_PATH_LINE_TO($Dom_path;323;301;131;161;369;161;177;301)
SVG_PATH_CLOSE($Dom_path)
SVG_SET_FILL_BRUSH($Dom_path;"red")
SVG_SET_STROKE_WIDTH($Dom_path;3)
SVG_SET_FILL_RULE($Dom_path;"evenodd")

 // 'nonzero'塗りつぶしルールで同様のオブジェクトを作成
$Dom_path:=SVG_New_path($Dom_SVG;250;75)
SVG_PATH_LINE_TO($Dom_path;323;301;131;161;369;161;177;301)
SVG_PATH_CLOSE($Dom_path)
SVG_SET_FILL_BRUSH($Dom_path;"red")
SVG_SET_STROKE_WIDTH($Dom_path;3)
SVG_SET_FILL_RULE($Dom_path;"nonzero")

 // 水平移動
SVG_SET_TRANSFORM_TRANSLATE($Dom_path;300)
```

SVG_SET_FILTER

SVG_SET_FILTER (svgObject ; id)

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
id	文字	➡	フィルター名

説明

SVG_SET_FILTER コマンドは、 *svgObject* 参照で指定したオブジェクトのフィルターを設定するのに使用することができます。*svgObject* 参照が無効である場合には、エラーが生成されます。属性が既に存在する場合、渡された値で上書きされます。

path 引数には、*SVG_Define_filter* コマンドによって指定された、使用するフィルター名を渡します。渡された名前が存在しない場合、エラーが生成されます。

例題

SVG_Define_filter コマンドを参照して下さい。

SVG_SET_ID

SVG_SET_ID (svgObject ; id)

引数	型		説明
svgObject	SVG_Ref	⇒	SVG 要素の参照
id	文字	⇒	オブジェクトに割り当てるID

説明

SVG_SET_ID コマンドは、 *svgObject* 参照で指定した SVG オブジェクトの'ID' プロパティを設定するのに使用することができます。もしこの属性が既に存在している場合、その値は引数として渡された値で上書きされます。

オブジェクトIDはオブジェクトを参照するのに使用されます。この参照は*SVG_Get_ID* コマンドを使用する事で取得することができます。このIDは、4Dの**SVG Find element ID by coordinates** コマンドによっても使用されます。

例題

```
$svg:=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";" white";2)
SVG_SET_ID($object;"border")
```

SVG_SET_MARKER

SVG_SET_MARKER (svgObject ; id {; position})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
id	文字	➡	マーカー名
position	文字	➡	マーカーの位置

説明

SVG_SET_MARKER コマンドを使用すると、*svgObject* 引数で参照しているオブジェクトにマーカーを関連付けたり、既存のマーカーを削除したりすることができます。*svgObject* 参照が 'line'、'path'、'polyline' もしくは 'polygon' 要素への参照ではない場合、エラーが生成されます。ある属性が既に存在する場合、その値は上書きされます。

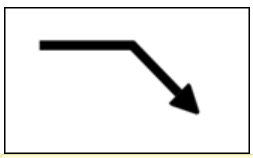
id 引数には *SVG_Define_marker* コマンドで指定したマーカー要素名を渡します。存在しない名前を渡した場合、エラーが生成されます。
既存のマーカーを削除するためには、 *id* p引数に"none" または空の文字列を渡します。

任意の*position* 引数を使用すると、オブジェクトにに対してのマーカーの位置を設定することができます。必要であれば、異なるマーカーをパスの始点、終点、またはどんなピークにも設定することができます。その際に使用できる値は以下の通りです:

- *start* を渡すと、マーカーはパスの始点に置かれます。
- *end* を渡すと、マーカーはパスの終点に置かれます。
- *middle* を渡すと、マーカーはパスの始点と終点以外の全てのピークに置かれます。
- *all* を渡すと、マーカーはパスの全てのピークに置かれます。
  この引数が省略された場合、マーカーはパスの終点に置かれます。

例題 1

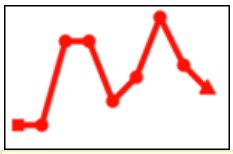
矢印を描画する場合を考えます:



```
$SVG:=SVG_New
 `矢印を設定します
$arrow:=SVG_Define_marker($SVG;"arrow";0;5;4;3;-1)
SVG_SET_VIEWBOX($arrow;0;0;10;10)
$path:=SVG_New_path($arrow;0;0)
SVG_SET_FILL_BRUSH($path;"black")
SVG_PATH_LINE_TO($path;10;5)
SVG_PATH_LINE_TO($path;0;10)
SVG_PATH_CLOSE($path)

$line:=SVG_New_path($SVG;100;75)
SVG_SET_STROKE_WIDTH($line;10)
SVG_PATH_LINE_TO($line;200;75)
SVG_PATH_LINE_TO($line;250;125)
 `パスの終わりに矢じりを置く
SVG_SET_MARKER($line;" arrow ")
```

例題 2

始まりと終わりに異なるマーカーがあるダイヤグラムを描画する場合を考えます:



```
$SVG:=SVG_New
SVG_SET_DEFAULT_BRUSHES("red";"red")

 `点をマークするための円を設定
$point:=SVG_Define_marker($SVG;"pointMarker";2;2;3;3)
SVG_SET_VIEWBOX($point;0;0;4;4)
SVG_New_circle($point;2;2;1)

 `開始地点の四角を設定
$start:=SVG_Define_marker($SVG;"startMarker";1;1;2;2)
SVG_New_rect($start;0;0;2;2)

 `終了地点の三角形を設定
$end:=SVG_Define_marker($SVG;"endMarker";5;5;3;3;60)
SVG_SET_VIEWBOX($end;0;0;10;10)
SVG_New_regular_polygon($end;10;3)

ARRAY LONGINT($tX;0)
ARRAY LONGINT($tY;0)
 `X軸
For($Lon_i;0;200;20)
   APPEND TO ARRAY($tX;$Lon_i+10)
End for
 `データ
APPEND TO ARRAY($tY;100)
APPEND TO ARRAY($tY;100)
APPEND TO ARRAY($tY;30)
APPEND TO ARRAY($tY;30)
APPEND TO ARRAY($tY;80)
APPEND TO ARRAY($tY;60)
APPEND TO ARRAY($tY;10)
APPEND TO ARRAY($tY;40)
APPEND TO ARRAY($tY;50)
APPEND TO ARRAY($tY;70)
$line:=SVG_New_polyline_by_arrays($SVG;->$tX;->$tY;"red";"none";5)
 `マーカーを設置:
SVG_SET_MARKER($line;"startMarker";"start")
SVG_SET_MARKER($line;"pointMarker";"middle")
SVG_SET_MARKER($line;"endMarker";"end")
```

SVG_SET_OPACITY

SVG_SET_OPACITY (svgObject ; background {; line})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
background	倍長整数	➡	不透明度
line	倍長整数	➡	不透明度

説明

SVG_SET_OPACITY コマンドは、 *svgObject* 参照で指定されたオブジェクトのフィリングとラインそれぞれの不透明度を設定するのに使用することができます。もしこれらの属性が既に存在している場合、それらの値は引数として渡された値で上書きされます。

不透明度として渡す値は、0から100の間でなければなりません。

例題

```
$svg :=SVG_New ` 新しいドキュメント作成
$object:=SVG_New_rect($svg ;10;10;200;100;0;0;"red";"blue")
SVG_SET_OPACITY($object;-1;50)   `ラインの不透明度を50%に設定
```

SVG_SET_ROUNDING_RECT

SVG_SET_ROUNDING_RECT (svgObject ; roundedX {; roundedY})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
roundedX	倍長整数	➡	X軸の半径
roundedY	倍長整数	➡	Y軸の半径

説明

SVG_SET_ROUNDING_RECT コマンドを使用すると、*svgObject* で参照してる長方形の角を丸めて長円形にするための、角の半径を設定することができます。これらの属性が既に存在していた場合、その値は引数として渡された値で上書きされます。*svgObject* が長方形の参照でなかった場合、エラーが生成されます。

渡される値は、ユーザー座標系内の値が期待されます。

例題

```
$svg:=SVG_New ` 新規にドキュメント作成
$object:=SVG_New_rect($svg;10;10;200;100)
SVG_SET_ROUNDING_RECT($object;20) `角を丸める
```

SVG_SET_SHAPE_RENDERING

SVG_SET_SHAPE_RENDERING (svgObject ; rendering)

引数	型		説明
svgObject	SVG_Ref	➡	SVG要素の参照
rendering	テキスト	➡	描画のタイプ

説明

SVG_SET_SHAPE_RENDERINGは*svgObject*で指定されるオブジェクトのグラフィック要素の描画に関わるトレードオフを設定するために使用します。*svgObject* がSVGオブジェクトでない場合エラーが生成されます。

rendering 引数は以下のいずれかでなければなりません: "auto", "optimizeSpeed", "crispEdges", "geometricPrecision" または "inherit"。これら以外の場合エラーが生成されます。

参照: http://www.w3.org/TR/2001/REC-SVG-20010904/painting.html#ShapeRenderingProperty

SVG_SET_STROKE_BRUSH

SVG_SET_STROKE_BRUSH (svgObject ; color)

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
color	文字	➡	カラー式

説明

SVG_SET_STROKE_BRUSH コマンドを使用すると、*svgObject* で参照しているSVGオブジェクトの線に使用されているカラーを設定することができます。これらの属性が既に存在していた場合、その値は引数として渡された値で上書きされます。

カラーについてのより詳細な情報に関しては、“” の章を参照して下さい。

例題

```
$svg:=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2)
SVG_SET_STROKE_BRUSH($object;"red")
```

SVG_SET_STROKE_DASHARRAY

SVG_SET_STROKE_DASHARRAY (svgObject ; dash {; value}{; value2 ; ... ; valueN})

引数	型		説明
svgObject	SVG_Ref	➡	SVG要素の参照
dash	実数	➡	最初の点線の長さ
value	倍長整数	➡	スペースと点線の長さ

説明

SVG_SET_STROKE_DASHARRAYを使用して、*svgObject*で指定したSVGオブジェクトの外枠を形作るパスの点線と間隔を設定するために使用します。*svgObject* が有効なSVG参照でない場合、エラーが生成されます。

dash 引数の値は点線パターンの最初の点の長さを示します。*value* 引数が省略されると、点線はこの値を点および間隔の長さに使用して描画されます。

dash 引数の小数値は、空でない場合、パターン内で点線が開始される距離を示します。

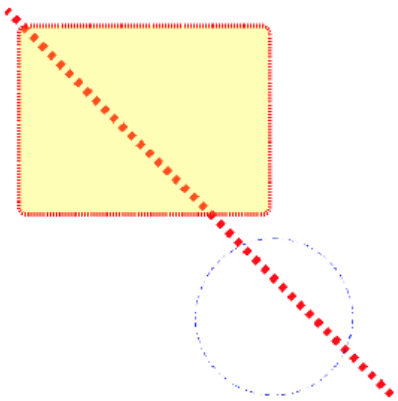
dash が 0 の場合、点線パターンは取り除かれます。

value 引数は先頭の点に続く間隔と点の長さを表します。(最初の点を含む) 奇数個の値が与えられると、値のリストはそれが偶数個の値を生成するまで繰り返されます。

参照: http://www.w3.org/TR/SVG/painting.html#StrokeProperties

例題

点線パス:



```
 // 線
$Dom_line:=SVG_New_line($Dom_SVG;10;10;500;500)
SVG_SET_STROKE_WIDTH($Dom_line;10)
SVG_SET_STROKE_DASHARRAY($Dom_line;8,099)
SVG_SET_STROKE_BRUSH($Dom_line;"red")

 // 四角
$Dom_rect:=SVG_New_rect($Dom_SVG;25;30;320;240;10;10;"red";"yellow:30")
SVG_SET_STROKE_WIDTH($Dom_rect;5)
SVG_SET_STROKE_DASHARRAY($Dom_rect;2)

 // 円
$Dom_circle:=SVG_New_circle($Dom_SVG;350;400;100;"blue";"none")
SVG_SET_STROKE_DASHARRAY($Dom_circle;2;4;6;8)
```

SVG_SET_STROKE_LINECAP

SVG_SET_STROKE_LINECAP (svgObject ; mode)

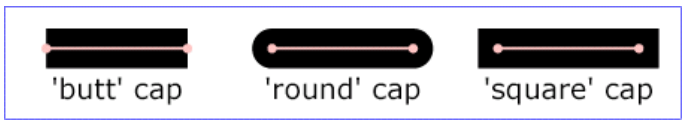
引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
mode	文字	➡	レンダリングモード

説明

SVG_SET_STROKE_LINECAP コマンドを使用すると、*svgObject* で参照しているSVGオブジェクトのパスの端の形状を指定することができます。この属性が既に存在していた場合、その値は引数として渡された値で上書きされます。

mode 引数には、SVGによって管理される以下の文字列のどれか一つを渡さなければなりません:

- *butt* (デフォルト): 標準
- *round*
- *square*
- *inherit*: 親オブジェクトから継承



mode 引数が上記以外の値を格納していた場合、エラーが生成されます。

SVG_SET_STROKE_LINEJOIN

SVG_SET_STROKE_LINEJOIN (svgObject ; mode)

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
mode	文字	➡	レンダリングモード

説明

SVG_SET_STROKE_LINEJOIN コマンドを使用すると、*svgObject* で参照しているSVGオブジェクトのパスのピークの形状を指定することができます。この属性が既に存在していた場合、その値は引数として渡された値で上書きされます。

mode 引数には、SVGによって管理される以下の文字列のどれか一つを渡さなければなりません:

- *miter* (デフォルト): 標準
- *round*
- *bevel*
- *inherit*: 親オブジェクトから継承

mode 引数が上記以外の値を格納していた場合、エラーが生成されます。

SVG_SET_STROKE_MITERLIMIT

SVG_SET_STROKE_MITERLIMIT (svgObject ; join)

引数	型		説明
svgObject	SVG_Ref	⇒	SVG要素の参照
join	倍長整数	⇒	結合値

説明

SVG_SET_STROKE_MITERLIMITは*svgObject*で指定したSVGオブジェクトのパスと外枠間の継ぎとめ部の長さに使用する境界を設定します。*svgObject* が有効なSVG参照でない場合エラーが生成されます。

*join*引数に-1を指定した場合、デフォルト値 (4) が使用されます。*join*引数に0を指定した場合、属性定義が取り除かれます。-1以外の負数を渡すとエラーになります。

参照: http://www.w3.org/TR/SVG/painting.html#StrokeProperties

SVG_SET_STROKE_WIDTH

SVG_SET_STROKE_WIDTH (svgObject ; strokeWidth {; unit})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
strokeWidth	実数	➡	線の太さ
unit	文字	➡	単位

説明

SVG_SET_STROKE_WIDTH コマンドを使用すると、*svgObject* 参照で指定したSVG オブジェクトの、線の太さを設定することができます。この属性が既に存在している場合、その値は引数として渡された値で上書きされます。

strokeWidth には線の太さの値を渡します。任意の*unit* 引数は、使用する単位を指定することができます。使用できる値は次の通りです: px、pt、pc、cm、mm、in、em、ex または %。*unit* 引数が省略された場合、*strokeWidth* 引数にはユーザー座標系システムの単位が期待されます。

例題

```
$svg :=SVG_New
SVG_SET_STROKE_WIDTH(SVG_New_rect($svg;10;10;200;200;0;0;"black";"white";2);10)
```

SVG_SET_TRANSFORM_FLIP

SVG_SET_TRANSFORM_FLIP (svgObject ; horizontal {; vertical})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
horizontal	ブール	➡	水平方向の反転
vertical	ブール	➡	垂直方向の反転

説明

SVG_SET_TRANSFORM_FLIP コマンドを使用すると、*svgObject* 参照で指定したSVG オブジェクトに水平/垂直方向の反転を適用することができます。

horizontal 引数がTrueに設定されている場合、水平方向の反転が適用されます。
vertical 引数がTrueに設定されている場合、垂直方向の反転が適用されます。

例題

テキストオブジェクトを反転する場合を考えます:


```
svgRef:=SVG_New
SVG_SET_VIEWBOX(svgRef;0;0;400;200)
$tx:=SVG_New_text(svgRef;"4D";10;0;"";96)
SVG_SET_FONT_COLOR($tx;"blue") `カラーを変更

 `エフェクト:
$tx:=SVG_New_text(svgRef;"4D";10;0;"";96) `同じテキストを使用
SVG_SET_FONT_COLOR($tx;"lightblue") ` カラーを変更
SVG_SET_TRANSFORM_FLIP($tx;Vrai) `垂直方向の反転を適用
SVG_SET_TRANSFORM_SKEW($tx;-10) `傾ける
SVG_SET_TRANSFORM_TRANSLATE($tx;-17;-193) `位置を調整
```

SVG_SET_TRANSFORM_MATRIX

SVG_SET_TRANSFORM_MATRIX (svgObject ; a ; b {; c ; d {; e ; f}})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
a	倍長整数	➡	transform行列のa要素
b	倍長整数	➡	transform行列のb要素
c	倍長整数	➡	transform行列のc要素
d	倍長整数	➡	transform行列のd要素
e	倍長整数	➡	transform行列のe要素
f	倍長整数	➡	transform行列のf要素

説明

SVG_SET_TRANSFORM_MATRIX コマンドは、*svgObject* 参照で指定したSVG オブジェクトに行列変換を適用します。
この型の変換は、例えば平行移動と回転などの変換を組み合わせたい場合に使用することができます。

例題


```
SVG_SET_TRANSFORM_MATRIX($ID;0,707;-0,707;0,707;0,707;255,03;111,21)
 `これは以下の三つの変換を適用したのと同等になります:
SVG_SET_TRANSFORM_TRANSLATE($ID;50;90)
SVG_SET_TRANSFORM_ROTATE($ID;-45)
SVG_SET_TRANSFORM_TRANSLATE($ID;130;160)
```

SVG_SET_TRANSFORM_ROTATE

SVG_SET_TRANSFORM_ROTATE (svgObject ; angle {; x ; y})

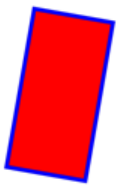
引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
angle	倍長整数	➡	回転の角度
x	倍長整数	➡	回転の中心のX軸座標
y	倍長整数	➡	回転の中心のY軸座標

説明

SVG_SET_TRANSFORM_ROTATE コマンドは、*svgObject* 参照で指定したオブジェクトに対して、角度を度数で指定した回転を適用します。
angle 引数は回転の角度を度数で受け取ります。また回転は時計回りに適用されます。

任意の*x* と *y* 引数が渡されなかった場合、回転はカレントのユーザー座標系システムの原点を中心にして行われます。これらの引数が渡された場合、回転は渡された座標(*x*、*y*)を中心として行われます。

例題



```
svgRef:=SVG_New
 `青い境界線の、赤い長方形を描画
$rec:=SVG_New_rect($svg;150;50;200;400;0;0;"blue";"red";10)
 `長方形の中心を中心として、10°の時計回りの回転を適用
SVG_SET_TRANSFORM_ROTATE($rec;370;175;225)
```

SVG_SET_TRANSFORM_SCALE

SVG_SET_TRANSFORM_SCALE (svgObject ; scaleX {; scaleY})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
scaleX	実数	➡	X軸の値
scaleY	実数	➡	Y軸の値

説明

SVG_SET_TRANSFORM_SCALE コマンドは、*svgObject* 参照で指定したオブジェクトに対して、水平/垂直方向の伸縮を適用します。

scaleX 引数の値がnullでない場合、オブジェクトは渡された値に応じて水平方向に拡大(値>1)または縮小(0<値<1)されます。値が1の時はオブジェクトの縮尺には何も変化はありません。

scaleY 引数が渡された場合、オブジェクトは渡された値に応じて垂直方向に拡大(値>1)または縮小(0<値<1)されます。値が1の時はオブジェクトの縮尺には何も変化はありません。この引数が省略された場合には、*scaleX* 引数の値が適用されます。

例題


```
$SVG:=SVG_New
$Text:=SVG_New_text($SVG;"Hello world!";5)
SVG_SET_TRANSFORM_SCALE($Text;3;12) `x軸方向に3倍、y軸方向に12倍のズーム
```

SVG_SET_TRANSFORM_SKEW

SVG_SET_TRANSFORM_SKEW (svgObject ; horizontal {; vertical})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
horizontal	倍長整数	➡	X軸方向の傾斜
vertical	倍長整数	➡	Y軸方向の傾斜

説明

SVG_SET_TRANSFORM_SKEW コマンドは、*svgObject* 参照で指定したオブジェクトに対して、水平/垂直方向の傾斜を指定します。

horizontal 引数の値がnullでない場合、オブジェクトは渡された値の分だけ水平方向に傾斜します。そうでない場合、この引数は無視されます。

vertical 引数の値がnullでない場合、オブジェクトは渡された値の分だけ垂直方向に傾斜します。

例題


```
$svg :=SVG_New
 `背景を描画
SVG_New_rect($svg;0;0;270;160;10;10;"black";"gray")
 `テキストを配置
$tx:=SVG_New_text($svg;"Hello world!";100;5;"";48)
 `フォントカラーを白に
SVG_SET_FONT_COLOR($tx;"white")
 `傾斜させる
SVG_SET_TRANSFORM_SKEW($tx;-50;10) `傾斜
```

SVG_SET_TRANSFORM_TRANSLATE

SVG_SET_TRANSFORM_TRANSLATE (svgObject ; x {; y})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
x	倍長整数	➡	X軸の座標
y	倍長整数	➡	Y軸の座標

説明

SVG_SET_TRANSFORM_TRANSLATE コマンドは、*svgObject* 参照で指定したオブジェクトに対して水平/垂直方向の平行移動を適用します。

x 引数の値がnullでない場合、オブジェクトは渡された値の分だけ水平方向に移動します。そうでない場合、この引数は無視されます。
y 引数の値が渡された場合、オブジェクトは渡された値の分だけ垂直方向に移動します。

例題

```
svgRef:=SVG_New
 `赤い長方形を描画
$Object:=SVG_New_rect(svgRef;0;0;200;100;0;0;"black";"red")
 `別の正方形を0,0の位置に描画
$Object:=SVG_New_rect(svgRef;0;0;20;20)
 `正方形を150,50の位置へと移動
SVG_SET_TRANSFORM_TRANSLATE($Object;150;50)
```

SVG_SET_VIEWBOX

SVG_SET_VIEWBOX (svgObject ; x ; y ; width ; height {; mode})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
x	実数	➡	ビューボックスのX軸の位置
y	実数	➡	ビューボックスのY軸の位置
width	実数	➡	ビューボックスの幅
height	実数	➡	ビューボックスの高さ
mode	テキスト	➡	ビューボックスへの調整

説明

SVG_SET_VIEWBOX コマンドを使用すると、*svgObject* 参照で指定したオブジェクトのビューボックスを指定することができます。この属性が既に存在している場合、その値は渡された引数で上書きされます。

値はユーザー座標系システムに沿う事が期待されます。

mode 引数を使用するとグラフィックがビューボックス内にぴったり合うようにするかどうか、合うようにするのであればビューボックスのサイズに対してどのように合わせるか、という点を指定することができます。*mode* 引数に渡せる値は、SVGによって認識される、以下の値のどれかでなければなりません:
‘none’、‘xMinYMin’、‘xMidYMin’、‘xMaxYMin’、‘xMinYMid’、‘xMidYMid’、‘xMaxYMid’、‘xMinYMax’、‘xMidYMax’、‘xMaxYMax’ または ‘true’ (xMidYMidに対して使用)です。

例題

```
 `4×8cmのサイズのSVGドキュメントを作成
$svg:=SVG_New
SVG_SET_DIMENSIONS($SVG;4;8;"cm")
 `ユーザー座標系システムを1 cm = 250 ユーザーポイントと指定
SVG_SET_VIEWBOX($svg;0;0;1000;2000;"true")
```

SVG_SET_VIEWPORT_FILL

SVG_SET_VIEWPORT_FILL (svgObject {; color {; opacity}})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
color	文字	➡	塗りのカラー
opacity	倍長整数	➡	不透明度のパーセント

説明

SVG_SET_VIEWPORT_FILL コマンドを使用すると、*svgObject* 参照で指定したSVGドキュメントの背景色を設定することができます。

この属性が既に存在している場合、その値は渡された引数で上書きされます。*svgObject* で指定したSVG要素がこの属性を受け取れない要素であった場合、エラーが生成されます。

任意の*color* 引数は、ピクチャーの背景に使用されるカラーを指定します。この引数が省略された場合もしくはこの引数に空の文字列を渡した場合、白が使用されます。カラーについての詳細な情報については、 の章を参照して下さい。

任意の*opacity* 引数を使用すると、この背景色に対する不透明度を指定することができます。この引数が省略された場合もしくはドキュメントに対して不透明度が何も指定されなかった場合、100%の値が使用されます。

SVG_SET_VISIBILITY

SVG_SET_VISIBILITY (svgObject {; hide})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
hide	ブール	➡	True = 表示、False = 非表示

説明

SVG_SET_VISIBILITY コマンドは、*svgObject* 参照で指定したSVG オブジェクトを表示/非表示を切り替えることができます。*svgObject* に渡された参照が非表示にできないオブジェクトだった場合、エラーが生成されます。

任意の*hide* 引数がTrueに設定、または省略された場合には、オブジェクトは表示されます。Falseに設定された場合には、オブジェクトは非表示になります。

例題

```
$svg :=SVG_New
$object:=SVG_New_rect($svg;10;10;200;200;0;0;"black";" white";2)
SVG_SET_VISIBILITY($object;False) `オブジェクトの記述はここにありますがレンダリングはされません。
```

SVG_SET_XY

SVG_SET_XY (svgObject ; x {; y})

引数	型		説明
svgObject	SVG_Ref	➡	SVG 要素の参照
x	倍長整数	➡	X軸の座標
y	倍長整数	➡	Y軸の座標

説明

SVG_SET_XY コマンドを使用すると、*svgObject* 参照で指定した長方形エリアのSVGオブジェクトの上左端の座標を設定します。これの属性が既に存在している場合、その値は渡された引数で上書きされます。*svgObject* で指定したSVG要素がこの属性を受け付けない要素であった場合、エラーが生成されます。

値はユーザー座標系システムに沿う事が期待されます。

例題

```
$svg:=SVG_New `新規ドキュメントを作成
$object:=SVG_New_image($svg;"#Pictures/logo4D.png") `ロゴを配置
SVG_SET_XY($object;10;40) `ピクチャーの位置を編集
```

描画

- SVG_Add_object
- SVG_ADD_POINT
- SVG_New_arc
- SVG_New_circle
- SVG_New_ellipse
- SVG_New_ellipse_bounded
- SVG_New_embedded_image
- SVG_New_image
- SVG_New_line
- SVG_New_path
- SVG_New_polygon
- SVG_New_polygon_by_arrays
- SVG_New_polyline
- SVG_New_polyline_by_arrays
- SVG_New_rect
- SVG_New_regular_polygon
- SVG_PATH_ARC
- SVG_PATH_CLOSE
- SVG_PATH_CURVE
- SVG_PATH_LINE_TO
- SVG_PATH_MOVE_TO
- SVG_PATH_QCURVE
- SVG_Use

SVG_Add_object

SVG_Add_object (targetSVGObject ; sourceSVGObject) -> 戻り値

引数	型		説明
targetSVGObject	SVG_Ref	➡	親要素の参照
sourceSVGObject	SVG_Ref	➡	追加するオブジェクトの参照
戻り値	SVG_Ref	➲	SVGオブジェクトの参照

説明

SVG_Add_objectを使用して、*targetSVGObject* で指定したSVGコンテナー中に、*sourceSVGObject* で指定されたSVGオブジェクトを配置します。そしてこのメソッドはその参照を返します。SVGコンテナーはドキュメントのルート、またはこのタイプの要素を含むことのできるSVGオブジェクトです。

SVG_ADD_POINT

SVG_ADD_POINT (parentSVGObject ; x ; y {; x2 ; y2 ; ... ; xN ; yN})

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
x	倍長整数	➡	新しい点のX軸の座標
y	倍長整数	➡	新しい点のY軸の座標

説明

SVG_ADD_POINT コマンドは*parentSVGObject* 引数によって参照されるパスに対して一つまたは複数のセグメントを追加します。パスは‘path’、‘polyline’ または ‘polygon’ タイプのどれかを渡す事ができます。*parentSVGObject* 引数のパスがこれらのタイプのいづれかへのパスでない場合、エラーが生成されます。

複数の座標のペアが渡された場合、それらの順に点が追加されていきます。この場合、最後の座標が不完全である(y座標がない)場合、その座標は無視されます。

例題

SVG_New_path コマンドの例題を参照して下さい。

SVG_New_arc

SVG_New_arc (parentSVGObject ; x ; y ; radius ; start ; end {; foregroundColor {; backgroundColor {; strokeWidth}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
x	倍長整数	➡	中心のX軸座標
y	倍長整数	➡	中心のY軸座標
radius	倍長整数	➡	円の半径
start	倍長整数	➡	弧の開始地点の値(度)
end	倍長整数	➡	弧の終了地点の値(度)
foregroundColor	文字	➡	線のカラー名またはグラデーション名
backgroundColor	文字	➡	背景のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	↩	弧の参照

説明

SVG_New_arc コマンドは*parentSVGObject* 引数で指定したSVGコンテナ内に新しい円弧を作成し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

任意の*foregroundColor* と *backgroundColor* 引数には、それぞれ線のカラー名と背景のカラー名を渡します(カラーの詳細については、 テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

例題 1

0° から 90° までの弧を描画します(塗りカラーと境界線カラー、線の太さはデフォルト):

```
svgRef:=SVG_New
objectRef:=SVG_New_arc(svgRef;100;100;90;90;180)
```

例題 2

90° から 180° までの弧を描画します(塗りカラーはlightblueで境界線カラーはblue、線の太さは2ピクセル):

```
svgRef:=SVG_New
objectRef:=SVG_New_arc(svgRef;100;100;90;180;270;"blue";"lightblue";2)
```

SVG_New_circle

SVG_New_circle (parentSVGObject ; x ; y ; radius {; foregroundColor {; backgroundColor {; strokeWidth}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
x	倍長整数	➡	中心のX軸座標
y	倍長整数	➡	中心のY軸座標
radius	倍長整数	➡	円の半径
foregroundColor	文字	➡	線のカラー名またはグラデーション名
backgroundColor	文字	➡	背景のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	⮌	円の参照

説明

SVG_New_circle コマンドは*parentSVGObject* 引数で指定したSVGコンテナ内に新しい円を作成し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

円は、中心の座標(*x* と *y* 引数)と*radius* 引数で指定した半径によってその位置とサイズが決定されます。

任意の*foregroundColor* と *backgroundColor* 引数には、それぞれ線のカラー名と背景のカラー名を渡します(カラーの詳細については、 テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

例題 1

円を描画します(塗りカラーと境界線カラー、線の太さはデフォルト):

```
svgRef:=SVG_New
objectRef:=SVG_New_circle(svgRef;100;100;90)
```

例題 2

塗りカラーはlightblueで境界線カラーはblue、線の太さは2ピクセルの円を描画します:

```
svgRef:=SVG_New
objectRef:=SVG_New_circle(svgRef;100;100;90;"blue";"lightblue";2)
```

SVG_New_ellipse

SVG_New_ellipse (parentSVGObject ; x ; y ; xRadius ; yRadius {; foregroundColor {; backgroundColor {; strokeWidth}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
x	倍長整数	➡	楕円の中心のX軸の座標
y	倍長整数	➡	楕円の中心のY軸の座標
xRadius	倍長整数	➡	X軸の半径
yRadius	倍長整数	➡	Y軸の半径
foregroundColor	文字	➡	線のカラー名またはグラデーション名
backgroundColor	文字	➡	背景のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	⬅	楕円の参照

説明

SVG_New_ellipse コマンドは*parentSVGObject* 引数で指定したSVGコンテナ内に新しい楕円を作成し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

楕円は、 *x*、*y*、*width* と *height* 引数で指定した値によってその位置とサイズが決定されます。

任意の*foregroundColor* と *backgroundColor* 引数には、それぞれ線のカラー名と背景のカラー名を渡します(カラーの詳細については、 テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

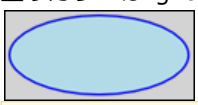
例題 1

楕円を描画します(塗りカラーと境界線カラー、線の太さはデフォルト):

```
svgRef:=SVG_New
objectRef:=SVG_New_ellipse(svgRef;100;50;90;40)
```

例題 2

塗りカラーはlightblueで境界線カラーはblue、線の太さは2ピクセルの楕円を描画します:



```
svgRef:=SVG_New
objectRef:=SVG_New_ellipse(svgRef;100;50;90;40;"blue";"lightblue";2)
```

SVG_New_ellipse_bounded

SVG_New_ellipse_bounded (parentSVGObject ; x ; y ; width ; height {; foregroundColor {; backgroundColor {; strokeWidth}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
x	倍長整数	➡	左上端のX軸の座標
y	倍長整数	➡	左上端のY軸の座標
width	倍長整数	➡	外接する長方形の幅
height	倍長整数	➡	外接する長方形の高さ
foregroundColor	文字	➡	線のカラー名またはグラデーション名
backgroundColor	文字	➡	背景のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	↩	楕円の参照

説明

SVG_New_ellipse_bounded コマンドは*parentSVGObject* 引数で指定したSVGコンテナ内に新しい楕円を作成し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

楕円は*x*、*y*、*width* と *height* 引数によって設定された長方形に収まるように作成されます。

任意の*foregroundColor* と *backgroundColor* 引数には、それぞれ線のカラー名と背景のカラー名を渡します(カラーの詳細については、 テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

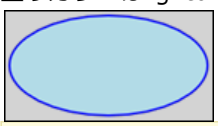
例題 1

楕円を描画します(塗りカラーと境界線カラー、線の太さはデフォルト):

```
svgRef:=SVG_New
objectRef:=SVG_New_ellipse_bounded(svgRef;10;10;200;100)
```

例題 2

塗りカラーはlightblueで境界線カラーはblue、線の太さは2ピクセルの楕円を描画します:



```
svgRef:=SVG_New
objectRef:=SVG_New_ellipse_bounded(svgRef;100;100;200;100;"blue";"lightblue";2)
```

SVG_New_embedded_image

SVG_New_embedded_image (parentSVGObject ; picture {; x {; y}}{; codec}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
picture	ピクチャー	➡	埋め込むピクチャー
x	倍長整数	➡	左上端のX軸の座標
y	倍長整数	➡	左上端のY軸の座標
codec	テキスト	➡	使用するコーデック
戻り値	SVG_Ref	↩	SVGオブジェクト参照

説明

SVG_New_embedded_image コマンドを使用すると、*parentSVGObject* 引数で指定したSVGコンテナ内に*picture* 引数で指定したピクチャーを埋め込み、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

ピクチャーはbase64でエンコードされた後、ドキュメント内に埋め込まれます。

picture 引数には4Dピクチャーフィールドもしくは変数を渡して下さい。

任意の*x* と *y* 引数を使用すると、SVGコンテナ内でのピクチャーの左上端の位置を指定することができます(デフォルトの値は0です)。

任意の*codec* 引数は、*picture* 引数で指定したピクチャーに対して使用するコーデックを指定します。この引数が省略された場合、デフォルトでのコーデックは".png" です。

例題

'Resources'フォルダ内にある'logo4D.png'ピクチャーを埋め込む場合を考えます:

```
svgRef:=SVG_New
$Path :=Resources folder+"logo4D.png")
READ PICTURE FILE($Path;$Picture)
If(OK=1)
   objectRef:=SVG_New_embedded_image(svgRef;$Picture)
End if
```

SVG_New_image

SVG_New_image (parentSVGObject ; url {; x ; y {; width ; height}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
url	文字	➡	ピクチャーのアドレス
x	倍長整数	➡	左上端のX軸の座標
y	倍長整数	➡	左上端のY軸の座標
width	倍長整数	➡	ピクチャーの幅
height	倍長整数	➡	ピクチャーの高さ
戻り値	SVG_Ref	⮌	SVGオブジェクト参照

説明

SVG_New_image コマンドを使用すると、*parentSVGObject* 引数で指定したSVGコンテナ内に*url* 引数で指定したアドレスにあるピクチャーを参照し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

url 引数はピクチャーの場所を指定します。その際、いくつかの形式を受け取ることができます:

- **ローカル URL** (file://… という形式のパス名): この場合、ピクチャーはレンダリング時に実際にアクセス可能だった場合にのみ表示されます。このローカルURLは相対パス(“#Pictures/myPicture.png”の形式)も使用可能です。この場合、コマンドはその渡されたパス名に対し、ホストデータベースの**Resources** フォルダのパス名を先頭に付加します。*width* と *height* 引数が省略された場合、それらはコマンドによって計算されます(ただし、サイズを渡した場合より動作が遅くなるので、推奨されません)。相対パスの場合、有効でなかったときにはエラーが生成されます。
- **非ローカル URL** (http://mySite.com/pictures/myPicture.jpeg): この場合には、リンクの有効性についての検証は行われません。また、*width* と *height* 引数が省略された場合にはエラーが生成されます。
- **相対 URL** ("../picture.png"): これはクライアント/サーバーモードにおいて、"Resources" フォルダ内にファイルが保管されている場合には特に有用です。相対URLの冒頭は、以下の様な形で渡す事ができます:
  - "**/**" を渡した場合には"~/Resources/SVG/" のパスを指定
  - "**./**" を渡した場合には"~/Resources/" のパスを指定
  - "**../**" を渡した場合にはデータベースフォルダを指定

任意の*x* と *y* 引数を使用すると、SVGコンテナ内でのピクチャーの左上端の位置を指定することができます(デフォルトの値は0です)。

width と *height* 引数はピクチャーが表示される長方形のサイズを指定するので、結果としてピクチャーのサイズとアスペクト比を決定します。これらの引数はホストデータベースの**Resources** フォルダ内のピクチャーを相対パスで参照している場合のみ省略可能です。*width* または *height* 引数が0だった場合、ピクチャーはレンダリングされません。

例題 1

‘Resources’フォルダ内の‘Pictures’フォルダ内にある‘logo4D.png’ピクチャーを配置する場合を考えます:

```
svgRef:=SVG_New
objectRef:=SVG_New_image(svgRef;"#Pictures/logo4D.png")
```

例題 2

‘4d.com’ サイトの’pictures’ ディレクトリ内にアクセス可能な‘4dlogo.gif’ ピクチャーを配置する場合を考えます:

```
svgRef:=SVG_New
objectRef:=SVG_New_image(svgRef;"http://www.4d.com/pictures/4dlogo.gif";20;20;39;53)
```

例題 3

以下は、相対URLを使用してピクチャーへとアクセスする例です:

```
SVG_New_image($Dom_svg;"./images/picture.png";10;10)
 // この場合のベースは"Resources" フォルダとなります
 // XML コードは xlink:href="../images/picture.png" となります
```

```
SVG_New_image($Dom_svg;"../picture.png";70;180)
 // この場合のベースはデータベースのフォルダとなります
 // XML コードは xlink:href="picture.png"となります
```

```
SVG_New_image($Dom_svg;"/sample pictures/picture.png";110;90;100;100)
 // この場合のベースは"Resources" フォルダ内の"SVG" フォルダとなります
 // XML コードは xlink:href="sample%20pictures/picture.gif"となります
```

SVG_New_line

SVG_New_line (parentSVGObject ; startX ; startY ; endX ; endY {; color {; strokeWidth}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
startX	倍長整数	➡	水平方向の開始地点
startY	倍長整数	➡	垂直方向の開始地点
endX	倍長整数	➡	水平方向の終了地点
endY	倍長整数	➡	垂直方向の終了地点
color	文字	➡	線のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	↩	線の参照

説明

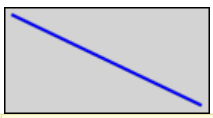
SVG_New_line コマンドは、*parentSVGObject* 引数で指定したSVGコンテナ内に新しい線を作成し、その参照を返します。オブジェクトの位置は、*startX*、*startY*、*endX* そして *endY* 引数によって決定されます。SVGコンテナは、ドキュメントのルート、またはこのタイプの要素を内包できる他のSVGオブジェクトを指定可能です。

任意の*color* 引数には線のカラー名を渡します(カラーの詳細については、 テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

例題

太さが3ピクセルの、blueの線を描画します:



```
svgRef:=SVG_New
objectRef:=SVG_New_line(svgRef;10;10;200;100;"blue";3)
```

SVG_New_path

SVG_New_path (parentSVGObject ; x ; y {; foregroundColor {; backgroundColor {; strokeWidth}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
x	倍長整数	➡	パスの開始地点のX軸の座標
y	倍長整数	➡	パスの開始地点のY軸の座標
foregroundColor	文字	➡	線のカラー名またはグラデーション名
backgroundColor	文字	➡	背景のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	↩	SVGオブジェクト参照

説明

SVG_New_path コマンドは*parentSVGObject* 引数で指定したSVGコンテナ内に新しいパスを作成し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

パスは、形状のアウトラインを表します。パスはカレントポイントという概念を呼び出すことによって描画されます。紙と鉛筆に例えて言うなら、カレントポイントとはペンの位置に相当するものです。このポイントが移動することにより、ペンが直線または曲線をなぞるように、形状の(開いた、または閉じた)アウトラインをトレースすることができます。
パスは、オブジェクトのアウトラインの形状を表し、その形状は以下の要素の宣言によって定義されます:
SVG_PATH_MOVE_TO (新規にカレントポイントを設置)、*SVG_PATH_LINE_TO* (直線を描画)、*SVG_PATH_CURVE* (3次ベジェ曲線を使用した曲線を描画)、*SVG_PATH_ARC* (円または楕円の弧を描画)、 そして *SVG_PATH_CLOSE* (最後のパスの開始点まで線を描画することによってカレントの形状を閉じる)。複合パス(別の言い方をすれば、複数のサブパスを持つパス)を使用することは可能です。これを使用することにより、オブジェクトに「ドーナッツホール」のようなエフェクトを加えることも可能です。

x と *y* 引数を使用することによってSVGコンテナ内でのパスの開始位置を指定することができます。

任意の*foregroundColor* と *backgroundColor* 引数には、それぞれ線のカラー名と背景のカラー名を渡します(カラーの詳細については、 テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

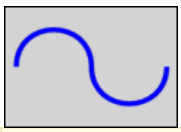
例題 1

閉じられた折れ線を描画する場合を考えます:

```
svgRef:=SVG_New
objectRef:=SVG_New_path(svgRef;20;20;"red";"none";5)
SVG_PATH_LINE_TO(objectRef;40)
SVG_PATH_LINE_TO(objectRef;40;40)
SVG_PATH_LINE_TO(objectRef;80;40;80;20;100;20;100;100;80;100;80;80;40;80;40;100;20;100)
SVG_PATH_CLOSE(objectRef)
```

例題 2

ベジェ曲線を描画する場合を考えます:

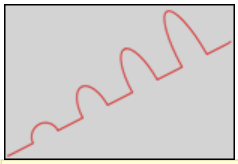


```
svgRef:=SVG_New
objectRef:=SVG_New_path(svgRef;100;200;"aquamarine";"none";10)
```

```
SVG_PATH_CURVE(objectRef;250;200;100;100;250;100)
SVG_PATH_CURVE(objectRef;400;200;400;300)
```

例題 3

パスデータ内に弧を描画する場合を考えます:



```
svgRef:=SVG_New
objectRef:=SVG_New_path(svgRef;20;300;"red";"none";2)
SVG_SET_OPTIONS(SVG_Get_options?-4)   `相対座標に変化
SVG_PATH_LINE_TO(objectRef;50;-25)
For($Lon_i;1;4;1)
   SVG_PATH_ARC(objectRef;25;25*$Lon_i;50;-25;-30)
   SVG_PATH_LINE_TO(objectRef;50;-25)
End for
```

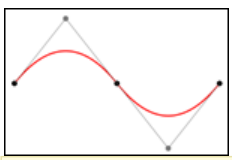
例題 4

複合パス(3次ベジエ曲線)を描画する場合を考えます:

```
 `新規SVGツリーを作成
$Txt_svg:=SVG_New(174,96;125,04;"4D Logo";"";True)

 `新規パスを作成
$Txt_path:=SVG_New_path($Txt_svg;150,665;13,021)
 `Set colors
SVG_SET_STROKE_BRUSH($Txt_path;"#212a6f")
SVG_SET_FILL_BRUSH($Txt_path;"#212a6f")
...
SVG_PATH_CURVE($Txt_path;-9,683;-6,54;-20,842;-8,888;-33,06;-10,462)
SVG_PATH_CURVE($Txt_path;-7,042;-0,915;-14,587;-0,877;-22,087;-0,877)
SVG_PATH_CURVE($Txt_path;-1,725;0;-4,312;-0,405;-5,761;0,24)
SVG_PATH_CURVE($Txt_path;-1,762;0;-5,092;-0,382;-6,479;0,24)
...
SVG_PATH_CURVE($Txt_path;181,489;70,216;177,236;30,976;150,665;13,021)
SVG_PATH_MOVE_TO($Txt_path;146,03;98,078)
...
SVG_PATH_CURVE($Txt_path;153,11;78,668;151,407;89,558;146,03;98,078)
```

例題 5

2次ベジエ曲線を描画する場合を考えます:



```
 `新規SVGツリーを作成
$svg:=SVG_New

 `線のカラーをblackに設定し、塗りカラーを無しに設定します
```

```
SVG_SET_DEFAULT_BRUSHES("";"none")

 `赤い2次ベジエ曲線を描画します
$qCurve:=SVG_New_path($svg;200;300)
SVG_SET_STROKE_BRUSH($qCurve;"red")
SVG_SET_STROKE_WIDTH($qCurve;5)
SVG_PATH_QCURVE($qCurve;400;50;600;300)
SVG_PATH_QCURVE($qCurve;1000;300)

 `終着点をblackで描画します
$g:=SVG_New_group($svg)
SVG_Set_description($g;"End points")
SVG_SET_DEFAULT_BRUSHES("black";"black")
SVG_New_circle($g;200;300;10)
SVG_New_circle($g;600;300;10)
SVG_New_circle($g;1000;300;10)

 `コントロールポイントと、コントロールポイント間の線をgrayで描画します
$g:=SVG_New_group($svg)
SVG_Set_description($g;"Control points and lines from end points to control points")
SVG_SET_DEFAULT_BRUSHES(SVG_Color_grey(50);"none")
$path:=SVG_New_path($svg;200;300)
SVG_SET_STROKE_WIDTH($path;2)
SVG_PATH_LINE_TO($path;400;50;600;300;800;550;1000;300)
$gray:=SVG_Color_grey(50) `grey 50%
SVG_SET_DEFAULT_BRUSHES($gray;$gray)
SVG_New_circle($g;400;50;10)
SVG_New_circle($g;800;550;10)
```

SVG_New_polygon

SVG_New_polygon (parentSVGObject {; points {; foregroundColor {; backgroundColor {; strokeWidth}}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
points	文字	➡	パス
foregroundColor	文字	➡	線のカラー名またはグラデーション名
backgroundColor	文字	➡	背景のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	⮌	ポリゴンの参照

説明

SVG_New_polygon コマンドは*parentSVGObject* 引数で指定したSVGコンテナ内に新しい閉じたフォームを作成し、その参照を返します。*parentSVGObject* 引数が有効な参照ではない場合、エラーが生成されます。

任意の*points* 引数を使用すると、SVGスタンダードに則ったポリゴンのパスポイントを指定することができます。この引数が省略される、または空だった場合、*SVG_ADD_POINT* コマンドを使用してポイントを設定することができます。

任意の*foregroundColor* と *backgroundColor* 引数には、それぞれ線のカラー名と背景のカラー名を渡します(カラーの詳細については、"カラー&グラデーション" テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

SVG_New_polygon_by_arrays

SVG_New_polygon_by_arrays (parentSVGObject ; xArrayPointer ; yArrayPointer {; foregroundColor {; backgroundColor {; strokeWidth}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
xArrayPointer	ポインター	➡	ポイントのX軸の座標
yArrayPointer	ポインター	➡	ポイントのY軸の座標
foregroundColor	文字	➡	線のカラー名またはグラデーション名
backgroundColor	文字	➡	背景のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	⬅	ポリゴンの参照

説明

SVG_New_polygon_by_arrays コマンドは、*parentSVGObject* 引数で指定したSVGコンテナ内に連結した直線群から構成される閉じたフォームを描画し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

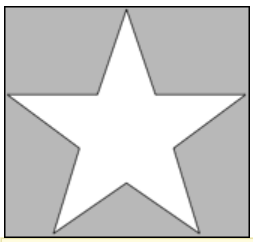
全ての座標の値はユーザー座標系に従います。

任意の*foregroundColor* と *backgroundColor* 引数には、それぞれ線のカラー名と背景のカラー名を渡します(カラーの詳細については、 テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

例題

デフォルトのカラーと太さの境界線で星マークを描画する場合を考えます:



```
ARRAY LONGINT($tX;0)
ARRAY LONGINT($tY;0)

APPEND TO ARRAY($tX;129)
APPEND TO ARRAY($tY;10)
APPEND TO ARRAY($tX;158)
APPEND TO ARRAY($tY;96)
APPEND TO ARRAY($tX;248)
APPEND TO ARRAY($tY;96)
APPEND TO ARRAY($tX;176)
APPEND TO ARRAY($tY;150)
APPEND TO ARRAY($tX;202)
APPEND TO ARRAY($tY;236)
APPEND TO ARRAY($tX;129)
APPEND TO ARRAY($tY;185)
APPEND TO ARRAY($tX;56)
APPEND TO ARRAY($tY;236)
APPEND TO ARRAY($tX;82)
APPEND TO ARRAY($tY;150)
APPEND TO ARRAY($tX;10)
APPEND TO ARRAY($tY;96)
APPEND TO ARRAY($tX;100)
APPEND TO ARRAY($tY;96)
```

```
objectRef:=SVG_New_polygon_by_arrays(svgRef;->$tX;->$tY)
```

SVG_New_polyline

SVG_New_polyline (parentSVGObject {; points {; foregroundColor {; backgroundColor {; strokeWidth}}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
points	文字	➡	パス
foregroundColor	文字	➡	線のカラー名またはグラデーション名
backgroundColor	文字	➡	背景のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	⮌	線の参照

説明

SVG_New_polyline コマンドは、*parentSVGObject* 引数で指定したSVGコンテナ内に新しい開かれた折れ線を作成し、その参照を返します。*parentSVGObject* 引数が有効な参照ではない場合、エラーが生成されます。

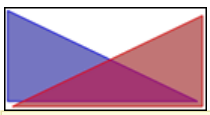
任意の*points* 引数を使用すると、SVGスタンダードに則った線のパスポイントを指定することができます。この引数が省略される、または空だった場合、*SVG_ADD_POINT* コマンドを使用してポイントを設定することができます。

任意の*foregroundColor* と *backgroundColor* 引数には、それぞれ線のカラー名と背景のカラー名を渡します(カラーの詳細については、 テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

例題

二つの三角形を描画する場合を考えます:



```
$polyline:=SVG_New_polyline($svg;"10,10 200,100 10,100 10,10";"blue";"blue:50")
$polyline:=SVG_New_polyline($svg;"";"red";"red:50")
SVG_ADD_POINT($polyline;205;15)
SVG_ADD_POINT($polyline;15;105)
SVG_ADD_POINT($polyline;205;105)
SVG_ADD_POINT($polyline;205;15)
```

SVG_New_polyline_by_arrays

SVG_New_polyline_by_arrays (parentSVGObject ; xArrayPointer ; yArrayPointer {; foregroundColor {; backgroundColor {; strokeWidth}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
xArrayPointer	ポインター	➡	ポイントのX軸の座標
yArrayPointer	ポインター	➡	ポイントのY軸の座標
foregroundColor	文字	➡	線のカラー名またはグラデーション名
backgroundColor	文字	➡	背景のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	⬅	線の参照

説明

SVG_New_polyline_by_arrays コマンドは、*parentSVGObject* 引数で指定したSVGコンテナ内に連結した直線群から構成される折れ線を描画し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

通常、'polyline '要素は開かれたフォームを描画しますが、閉じたフォームを描画することもできます。この場合、最後のポイントを最初のポイントと等しい箇所に指定する必要があります。

全ての座標の値はユーザー座標系に従います。

任意の*foregroundColor* と *backgroundColor* 引数には、それぞれ線のカラー名と背景のカラー名を渡します(カラーの詳細については、 テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

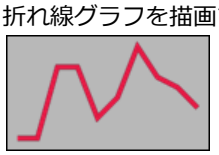
例題 1

デフォルトのカラーと太さの境界線で三角形を描画する場合を考えます:

```
ARRAY LONGINT($tX;0)
ARRAY LONGINT($tY;0)

APPEND TO ARRAY($tX;10)
APPEND TO ARRAY($tY;10)
APPEND TO ARRAY($tX;200)
APPEND TO ARRAY($tY;100)
APPEND TO ARRAY($tX;10)
APPEND TO ARRAY($tY;100)
APPEND TO ARRAY($tX;10)
APPEND TO ARRAY($tY;10)

svgRef:=SVG_New
objectRef:=SVG_New_polyline_by_arrays(svgRef;->$tX;->$tY)
```

例題 2

折れ線グラフを描画する場合を考えます:



```
ARRAY LONGINT($tX;0)
ARRAY LONGINT($tY;0)
 `X軸
For($Lon_i;0;200;20)
   APPEND TO ARRAY($tX;$Lon_i)
End for
 `値
APPEND TO ARRAY($tY;100)
APPEND TO ARRAY($tY;100)
APPEND TO ARRAY($tY;30)
APPEND TO ARRAY($tY;30)
APPEND TO ARRAY($tY;80)
APPEND TO ARRAY($tY;60)
APPEND TO ARRAY($tY;10)
APPEND TO ARRAY($tY;40)
APPEND TO ARRAY($tY;50)
APPEND TO ARRAY($tY;70)

objectRef:=SVG_New_polyline_by_arrays(svgRef;->$tX;->$tY;"crimson";"none";5)
```

SVG_New_rect

SVG_New_rect (parentSVGObject ; x ; y ; width ; height {; roundedX {; roundedY {; foregroundColor {; backgroundColor {; strokeWidth}}}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
x	倍長整数	➡	左上端のX座標
y	倍長整数	➡	左上端のY座標
width	倍長整数	➡	長方形の幅
height	倍長整数	➡	長方形の高さ
roundedX	倍長整数	➡	水平方向の曲線
roundedY	倍長整数	➡	垂直方向の曲線
foregroundColor	文字	➡	線のカラー名またはグラデーション名
backgroundColor	文字	➡	背景のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	↩	長方形の参照

説明

SVG_New_rect コマンドは*parentSVGObject* 引数で指定したSVGコンテナ内に新しい長方形を作成し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントではない場合、エラーが生成されます。

長方形は、*x*、*y*、*width* そして*height* 引数の値に応じて位置とサイズが決定されます。

任意の*roundedX* と *roundedY* 引数を使用すると、指定した値に応じて角を丸めることができます。*roundedY* 引数が省略された場合(または-1を渡した場合)、曲線は通常のものになります。この引数をコマンドに無視させたい場合には、この引数に-1を渡して下さい。

任意の*foregroundColor* と *backgroundColor* 引数には、それぞれ線のカラー名と背景のカラー名を渡します(カラーの詳細については、 テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

例題 1

デフォルトのカラーと太さの境界線で長方形を描画する場合を考えます:

```
svgRef:=SVG_New
objectRef:=SVG_New_rect(svgRef;10;10;200;100)
```

例題 2

3ピクセル幅の赤い境界線をもった、青い長方形を描画する場合を考えます:

```
svgRef:=SVG_New
objectRef:=SVG_New_rect(svgRef;10;10;200;100;0;0;"red";"blue";3)
```

例題 3

丸い角の、デフォルトのカラーと太さの境界線で三角形を描画する場合を考えます:

```
svgRef:=SVG_New
objectRef:=SVG_New_rect(svgRef;10;10;100;100;20)
```

例題 4

左右の端が丸い、青の境界線を持った水色の長方形を描画する場合を考えます:

```
svgRef:=SVG_New
objectRef:=SVG_New_rect(svgRef;10;10;200;100;-1;50;"blue";"lightblue")
```

SVG_New_regular_polygon

SVG_New_regular_polygon (parentSVGObject ; width ; number {; x {; y {; foregroundColor {; backgroundColor {; strokeWidth}}}}}) -> 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
width	倍長整数	➡	周囲の円の直径
number	倍長整数	➡	辺の数
x	倍長整数	➡	中心のX軸の座標
y	倍長整数	➡	中心のY軸の座標
foregroundColor	文字	➡	線のカラー名またはグラデーション名
backgroundColor	文字	➡	背景のカラー名またはグラデーション名
strokeWidth	実数	➡	線の太さ
戻り値	SVG_Ref	↩	多角形の参照

説明

SVG_New_regular_polygon コマンドは、*parentSVGObject* 引数で指定したSVGコンテナ内に*width* 引数で半径を指定した円に内接する多角形を描画し、その参照を返します。*parentSVGObject* 引数が有効な参照ではない場合、エラーが生成されます。

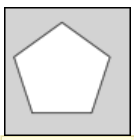
任意の*x* と *y* 引数を使用すると、円の中心を指定することができます。これらが省略された場合、円はドキュメントの左上端に接するように描画されます。

任意の*foregroundColor* と *backgroundColor* 引数には、それぞれ線のカラー名と背景のカラー名を渡します(カラーの詳細については、 テーマのコマンドを参照して下さい)。

任意の*strokeWidth* 引数には、ペンのサイズ(線の太さ)の値を渡します(単位：ピクセル)。デフォルトの値は1です。

例題 1

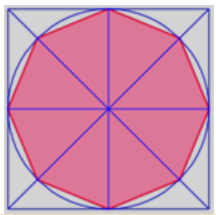
塗りカラーと境界線カラー、線の太さがデフォルトの、五角形を描画する場合を考えます:



```
svgRef:=SVG_New
objectRef:=SVG_New_regular_polygon(svgRef;100;5)
```

例題 2

八角形と、それに外接する円と、対角線を描画する場合を考えます:



```
svgRef:=SVG_New
$width:=200
$sides:=8
objectRef:=SVG_New_regular_polygon(svgRef;$width;$sides;0;0;"crimson";"palevioletred";2)

$radius:=$width/2
objectRef:=SVG_New_rect(svgRef;0;0;$width;$width;0;0;"blue";"none")
objectRef:=SVG_New_line(svgRef;0;$radius;$width;$radius;"blue")
objectRef:=SVG_New_line(svgRef;$radius;0;$radius;$width;"blue")
```

```
    objectRef:=SVG_New_line(svgRef;0;0;$width;$width;"blue")
    objectRef:=SVG_New_line(svgRef;$width;0;0;$width;"blue")
    objectRef:=SVG_New_circle(svgRef;$radius;$radius;$radius;"blue";"none")
```

SVG_PATH_ARC

SVG_PATH_ARC (parentSVGObject ; xRadius ; yRadius ; x ; y {; rotation {; arcpath}})

引数	型		説明
parentSVGObject	SVG_Ref	➡	パス要素の参照
xRadius	倍長整数	➡	楕円のX軸の半径
yRadius	倍長整数	➡	楕円のY軸の半径
x	倍長整数	➡	終着点のX軸の座標
y	倍長整数	➡	終着点のY軸の座標
rotation	倍長整数	➡	回転の値
arcpath	倍長整数	➡	弧の描画方法を設定

説明

SVG_PATH_ARC コマンドはカレントポイントから、*parentSVGObject* 引数で参照しているパスの終着点(*x*、*y*)までの楕円状の弧を描画します。*parentSVGObject* がパス参照('path'要素)でない場合、エラーが生成されます。

楕円のサイズと座標は、二つの半径(*xRadius*、*yRadius*)と*rotation* の値によって決定されます。*rotation* 引数の値はカレントの座標系に対して楕円全体をどれぐらい回転させるかを指定します。

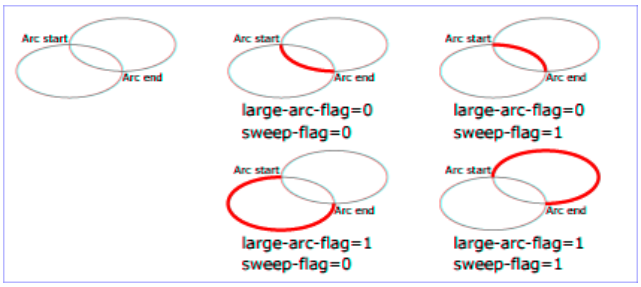
任意の*arcpath* 引数を使用すると、弧がどのように描画されるかを決定する制約の組み合わせを指定することができます。large-arc-flag制約は、二つの弧のうち、大きい方の弧(180°を超える方)を選択するかどうかを決定し、sweep-flag制約は弧が描画ささされる方向を選択します(正の角度か負の角度)。

2種類の制約の、4通りの組み合わせを表す以下の値を渡す事ができます:

- 0: large-arc-flag = 0, sweep-flag = 1
- 1: large-arc-flag = 1, sweep-flag = 0
- 2: large-arc-flag = 0, sweep-flag = 0
- 3: large-arc-flag = 1, sweep-flag = 1

large-arc-flag が1のとき、大きい方の弧が描画されます(反対に0のときには小さい方が描画されます)。sweep-flag が1のとき、弧は正の角度に描画されます(0のときには負の角度に描画されます)。

以下の図は4通りの組み合わせを表します:



デフォルトでは、*arcpath* 引数の値は0です(large-arc-flag=0, sweep-flag=1)。

例題

SVG_New_path コマンドの例題を参照して下さい。

SVG_PATH_CLOSE

SVG_PATH_CLOSE (parentSVGObject)

引数	型		説明
parentSVGObject	SVG_Ref	➡	パスの参照

説明

SVG_PATH_CLOSE コマンドは、*parentSVGObject* 引数によって参照されているカレントのサブパスを、カレントポイントから最初のポイントまで直線を引くことで閉じます。*parentSVGObject* 引数がパス参照('path'要素)ではない場合、エラーが生成されます。

例題

SVG_New_path コマンドの例題を参照して下さい。

SVG_PATH_CURVE

SVG_PATH_CURVE (parentSVGObject {; controlStartX ; controlStartY} ; controlEndX ; controlEndY ; x ; y)

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
controlStartX	倍長整数	➡	コントロールポイント(始点)のX軸の座標
controlStartY	倍長整数	➡	コントロールポイント(始点)のY軸の座標
controlEndX	倍長整数	➡	コントロールポイント(終点)のX軸の座標
controlEndY	倍長整数	➡	コントロールポイント(終点)のY軸の座標
x	倍長整数	➡	終点のX軸の座標
y	倍長整数	➡	終点のY軸の座標

説明

SVG_PATH_CURVE コマンドは、*parentSVGObject* 引数によって参照されているパスに対して、座標(*x*、*y*)を指定したポイントから始まる3次ベジエ曲線を追加します。*parentSVGObject* 引数がパス参照('path' 要素)ではない場合、エラーが生成されます。

任意の*controlStartX* と *controlStartY* 引数を使用すると、曲線の始点におけるコントロールポイントの位置を指定することができます。この引数が省略された場合、最初のコントロールポイントはカレントポイントに対する、直前のコマンドの二つ目のコントロールポイントの点対称の位置が用いられます。

controlEndX と *controlEndY* 引数を使用すると、曲線の終点に対するコントロールポイントの位置を指定することができます。

例題

SVG_New_path コマンドの例題を参照して下さい。

SVG_PATH_LINE_TO

SVG_PATH_LINE_TO (parentSVGObject ; x {; y}{; x2 ; y2 ; ... ; xN ; yN})

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
x	倍長整数	➡	新しいポイントのX軸の座標
y	倍長整数	➡	新しいポイントのY軸の座標

説明

SVG_PATH_LINE_TO コマンドは*parentSVGObject* 引数によって参照されているパスに対して、一つまたは複数の直線のセグメントを追加します。*parentSVGObject* 引数がパス参照('path' 要素)ではない場合、エラーが生成されます。

x と *y* 引数を使用すると、SVGコンテナ内のパスの始点の位置を指定することができます。

- *x* 引数のみが渡された場合、直線はカレントポイント(xc、yc) からポイント(x、yc) に対して垂直の線が描画されます。
- *x* と *y* と、両方の引数が渡された場合、直線はカレントポイント(xc、yc) からポイント(x、y) に対して描画されます。
- 複数の座標のペアが渡された場合、それぞれの座標に応じて異なる点が順次追加されていきます。このとき、最後のペアが不完全である(y座標が欠けている)場合、その座標は無視されます。

例題

SVG_New_path コマンドの例題を参照して下さい。

SVG_PATH_MOVE_TO

SVG_PATH_MOVE_TO (parentSVGObject ; x ; y)

引数	型		説明
parentSVGObject	SVG_Ref	➡	パスの参照
x	倍長整数	➡	X軸の座標
y	倍長整数	➡	Y軸の座標

説明

SVG_PATH_MOVE_TO コマンドは*parentSVGObject* 引数によって参照されているパス内に、与えられた座標(*x*、*y*) の点からの新しいサブパスを作成します。*parentSVGObject* 引数がパス参照('path' 要素)でない場合、エラーが生成されます。

これは、"ペン"を紙面から浮かせて新しい場所へと動かしたようなエフェクトをもたらします。カレントポイントは新しい始点となり、*SVG_PATH_CLOSE* コマンドによって考慮されるようになります。

例題

SVG_New_path コマンドの例題を参照して下さい。

SVG_PATH_QCURVE

SVG_PATH_QCURVE (parentSVGObject {; controlX ; controlY} ; x ; y)

引数	型		説明
parentSVGObject	SVG_Ref	➡	親要素の参照
controlX	倍長整数	➡	コントロールポイントのX軸の座標
controlY	倍長整数	➡	コントロールポイントのY軸の座標
x	倍長整数	➡	終点のX軸の座標
y	倍長整数	➡	終点のY軸の座標

説明

SVG_PATH_CURVE コマンドは、*parentSVGObject* 引数によって参照されている線に、カレントポイントから座標(*x*、*y*)で指定した点への2次ベジエ曲線を追加します。*parentSVGObject* 引数がパス参照('path' 要素)でない場合、エラーが生成されます。

任意の*controlX* と *controlY* 引数を使用すると、曲線の始点でのコントロールポイントの位置を指定することができます。省略された場合、最初のコントロールポイントはカレントポイントに対する、直前のコマンドの二つ目のコントロールポイントの点対称の位置が用いられます。

例題

SVG_New_path コマンドの例を参照して下さい。

SVG_Use

SVG_Use (parentSVGObjec ; id {; x ; y ; width ; height {; mode}}) -> 戻り値

引数	型		説明
parentSVGObjec	SVG_Ref	➡	親要素の参照
id	文字	➡	シンボル名
x	倍長整数	➡	ビューボックスのX位置
y	倍長整数	➡	ビューボックスのY位置
width	倍長整数	➡	ビューボックスの幅
height	倍長整数	➡	ビューボックスの高さ
mode	文字	➡	ビューボックスへの調整
戻り値	SVG_Ref	↩	SVGオブジェクト参照

説明

SVG_Use コマンドは、*parentSVGObject* 引数によって参照されているSVGコンテナ内にシンボルを配置し、その参照を返します。*parentSVGObject* 引数がSVGドキュメントでない場合、または*id* がSVGドキュメントのオブジェクト名でない場合、エラーが生成されます。

グラフィックオブジェクトを指定するためにシンボルが使用されます。これは直接レンダリングされる訳ではありませんが、*SVG_Use* コマンドを使用することによってインスタンスが作成されます。

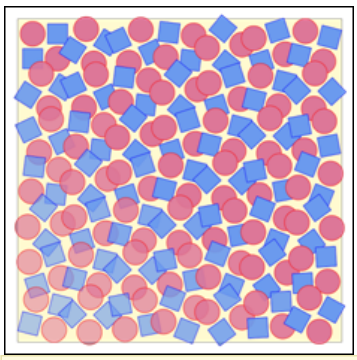
id 引数には、シンボル名を指定します。

任意の*x*、*y*、*width* と *height* 引数は、ビューボックス長方形('viewBox' 属性)を指定します。

任意の*mode* 引数を使用すると、ビューボックスのサイズに対してグラフィックを詰めるかどうか、詰めるとしたらどのように詰めるかを指定します(*SVG_New* コマンドを参照して下さい)。

例題

二つの赤い円と二つの青い四角形で構成されるグラフィックを指定します。このグラフィックをループで使用し、そのオリジナルのグラフィックの位置、透明度、回転を変化させながら36回作成します。



```
$SVG:=SVG_New
 `Draw a background
SVG_New_rect($SVG;20;20;650;650;0;0;"gray";"lemonchiffon")
 `二つの赤い円と二つの青い四角で構成されるシンボルを指定します
$Symbol:=SVG_Define_symbol($SVG;"MySymbol";0;0;110;110;"true")
SVG_New_circle($Symbol;30;30;25;"red";"palevioletred")
SVG_New_rect($Symbol;10;60;40;40;0;0;"blue";"cornflowerblue")
SVG_New_rect($Symbol;60;10;40;40;0;0;"blue";"cornflowerblue")
SVG_New_circle($Symbol;80;80;25;"red";"palevioletred")
 `グループ内において、
$g:=SVG_New_group($SVG)
 `…ドキュメントの左上角から20組配置します
SVG_SET_TRANSFORM_TRANSLATE($g;20;20)
 `…36パータンを位置、透明度、回転角度を変化させながら配置します
For($x;0;540;90)   `6 columns
```

```
   For($y;0;540;90)    `6 rows
      $use:=SVG_Use($g;"MySymbol";$x;$y;110;110)
      SVG_SET_OPACITY($use;100-($y/12)+($x/12)
      SVG_SET_TRANSFORM_ROTATE($use;($x*(18/50))+($y*(18/50));($x+55);($y+55))
   End for
End for
```

付録

付録A カラー一覧

以下の一覧は、SVGによって認識されているカラーの一覧です。より詳細な情報に関しては、 の章を参照して下さい。

aliceblue	rgb(240, 248, 255)
antiquewhite	rgb(250, 235, 215)
aqua	rgb(0, 255, 255)
aquamarine	rgb(127, 255, 212)
azure	rgb(240, 255, 255)
beige	rgb(245, 245, 220)
bisque	rgb(255, 228, 196)
black	rgb(0, 0, 0)
blanchedalmond	rgb(255, 235, 205)
blue	rgb(0, 0, 255)
blueviolet	rgb(138, 43, 226)
brown	rgb(165, 42, 42)
burlywood	rgb(222, 184, 135)
cadetblue	rgb(95, 158, 160)
chartreuse	rgb(127, 255, 0)
chocolate	rgb(210, 105, 30)
coral	rgb(255, 127, 80)
cornflowerblue	rgb(100, 149, 237)
cornsilk	rgb(255, 248, 220)
crimson	rgb(220, 20, 60)
cyan	rgb(0, 255, 255)
darkblue	rgb(0, 0, 139)
darkcyan	rgb(0, 139, 139)
darkgoldenrod	rgb(184, 134, 11)
darkgray	rgb(169, 169, 169)
darkgreen	rgb(0, 100, 0)
darkgrey	rgb(169, 169, 169)
darkkhaki	rgb(189, 183, 107)
darkmagenta	rgb(139, 0, 139)
darkolivegreen	rgb(85, 107, 47)
darkorange	rgb(255, 140, 0)
darkorchid	rgb(153, 50, 204)
darkred	rgb(139, 0, 0)
darksalmon	rgb(233, 150, 122)
darkseagreen	rgb(143, 188, 143)
darkslateblue	rgb(72, 61, 139)
darkslategray	rgb(47, 79, 79)
darkslategrey	rgb(47, 79, 79)
darkturquoise	rgb(0, 206, 209)
darkviolet	rgb(148, 0, 211)
deeppink	rgb(255, 20, 147)
deepskyblue	rgb(0, 191, 255)
dimgray	rgb(105, 105, 105)
dimgrey	rgb(105, 105, 105)
dodgerblue	rgb(30, 144, 255)
firebrick	rgb(178, 34, 34)
floralwhite	rgb(255, 250, 240)
forestgreen	rgb(34, 139, 34)
fuchsia	rgb(255, 0, 255)
gainsboro	rgb(220, 220, 220)
ghostwhite	rgb(248, 248, 255)
gold	rgb(255, 215, 0)
goldenrod	rgb(218, 165, 32)
gray	rgb(128, 128, 128)
grey	rgb(128, 128, 128)
green	rgb(0, 128, 0)
greenyellow	rgb(173, 255, 47)
honeydew	rgb(240, 255, 240)
hotpink	rgb(255, 105, 180)
indianred	rgb(205, 92, 92)
indigo	rgb(75, 0, 130)
ivory	rgb(255, 255, 240)
khaki	rgb(240, 230, 140)
lavender	rgb(230, 230, 250)
lavenderblush	rgb(255, 240, 245)
lawngreen	rgb(124, 252, 0)
lemonchiffon	rgb(255, 250, 205)
lightblue	rgb(173, 216, 230)
lightcoral	rgb(240, 128, 128)
lightcyan	rgb(224, 255, 255)
lightgoldenrodyellow	rgb(250, 250, 210)
lightgray	rgb(211, 211, 211)
lightgreen	rgb(144, 238, 144)
lightgrey	rgb(211, 211, 211)

lightpink	rgb (255, 182, 193)
lightsalmon	rgb (255, 160, 122)
lightseagreen	rgb (32, 178, 170)
lightskyblue	rgb (135, 206, 250)
lightslategray	rgb (119, 136, 153)
lightslategrey	rgb (119, 136, 153)
lightsteelblue	rgb (176, 196, 222)
lightyellow	rgb (255, 255, 224)
lime	rgb (0, 255, 0)
limegreen	rgb (50, 205, 50)
linen	rgb (250, 240, 230)
magenta	rgb (255, 0, 255)
maroon	rgb (128, 0, 0)
mediumaquamarine	rgb (102, 205, 170)
mediumblue	rgb (0, 0, 205)
mediumorchid	rgb (186, 85, 211)
mediumpurple	rgb (147, 112, 219)
mediumseagreen	rgb (60, 179, 113)
mediumslateblue	rgb (123, 104, 238)
mediumspringgreen	rgb (0, 250, 154)
mediumturquoise	rgb (72, 209, 204)
mediumvioletred	rgb (199, 21, 133)
midnightblue	rgb (25, 25, 112)
mintcream	rgb (245, 255, 250)
mistyrose	rgb (255, 228, 225)
moccasin	rgb (255, 228, 181)
navajowhite	rgb (255, 222, 173)
navy	rgb (0, 0, 128)
oldlace	rgb (253, 245, 230)
olive	rgb (128, 128, 0)
olivedrab	rgb (107, 142, 35)
orange	rgb (255, 165, 0)
orangered	rgb (255, 69, 0)
orchid	rgb (218, 112, 214)
palegoldenrod	rgb (238, 232, 170)
palegreen	rgb (152, 251, 152)
paleturquoise	rgb (175, 238, 238)
palevioletred	rgb (219, 112, 147)
papayawhip	rgb (255, 239, 213)
peachpuff	rgb (255, 218, 185)
peru	rgb (205, 133, 63)
pink	rgb (255, 192, 203)
plum	rgb (221, 160, 221)
powderblue	rgb (176, 224, 230)
purple	rgb (128, 0, 128)
red	rgb (255, 0, 0)
rosybrown	rgb (188, 143, 143)
royalblue	rgb (65, 105, 225)
saddlebrown	rgb (139, 69, 19)
salmon	rgb (250, 128, 114)
sandybrown	rgb (244, 164, 96)
seagreen	rgb (46, 139, 87)
seashell	rgb (255, 245, 238)
sienna	rgb (160, 82, 45)
silver	rgb (192, 192, 192)
skyblue	rgb (135, 206, 235)
slateblue	rgb (106, 90, 205)
slategray	rgb (112, 128, 144)
slategrey	rgb (112, 128, 144)
snow	rgb (255, 250, 250)
springgreen	rgb (0, 255, 127)
steelblue	rgb (70, 130, 180)
tan	rgb (210, 180, 140)
teal	rgb (0, 128, 128)
thistle	rgb (216, 191, 216)
tomato	rgb (255, 99, 71)
turquoise	rgb (64, 224, 208)
violet	rgb (238, 130, 238)
wheat	rgb (245, 222, 179)
white	rgb (255, 255, 255)
whitesmoke	rgb (245, 245, 245)
yellow	rgb (255, 255, 0)
yellowgreen	rgb (154, 205, 50)

付録B 外部リンク

以下はSVG標準に関連するリンクの一覧です:

概要

http://www.w3.org/Graphics/SVG/

仕様

http://www.w3.org/TR/SVG12/ (Status: Working Draft 13 April 2005)
http://www.yoyodesign.org/doc/w3c/svg1/ (フランス語版(バージョン1.0))

チュートリアル

http://www.w3.org/Consortium/Offices/Presentations/SVG/1.svg

コミュニティ

http://svg.org/
http://svgfr.org
http://www.openclipart.org/
http://www.gosvg.net/

SVGコンポーネント - コマンドリスト (文字順)

- SVG_ABOUT
- SVG_ADD_NAMESPACE
- SVG_Add_object
- SVG_ADD_POINT
- SVG_APPEND_TEXT_TO_TEXTAREA