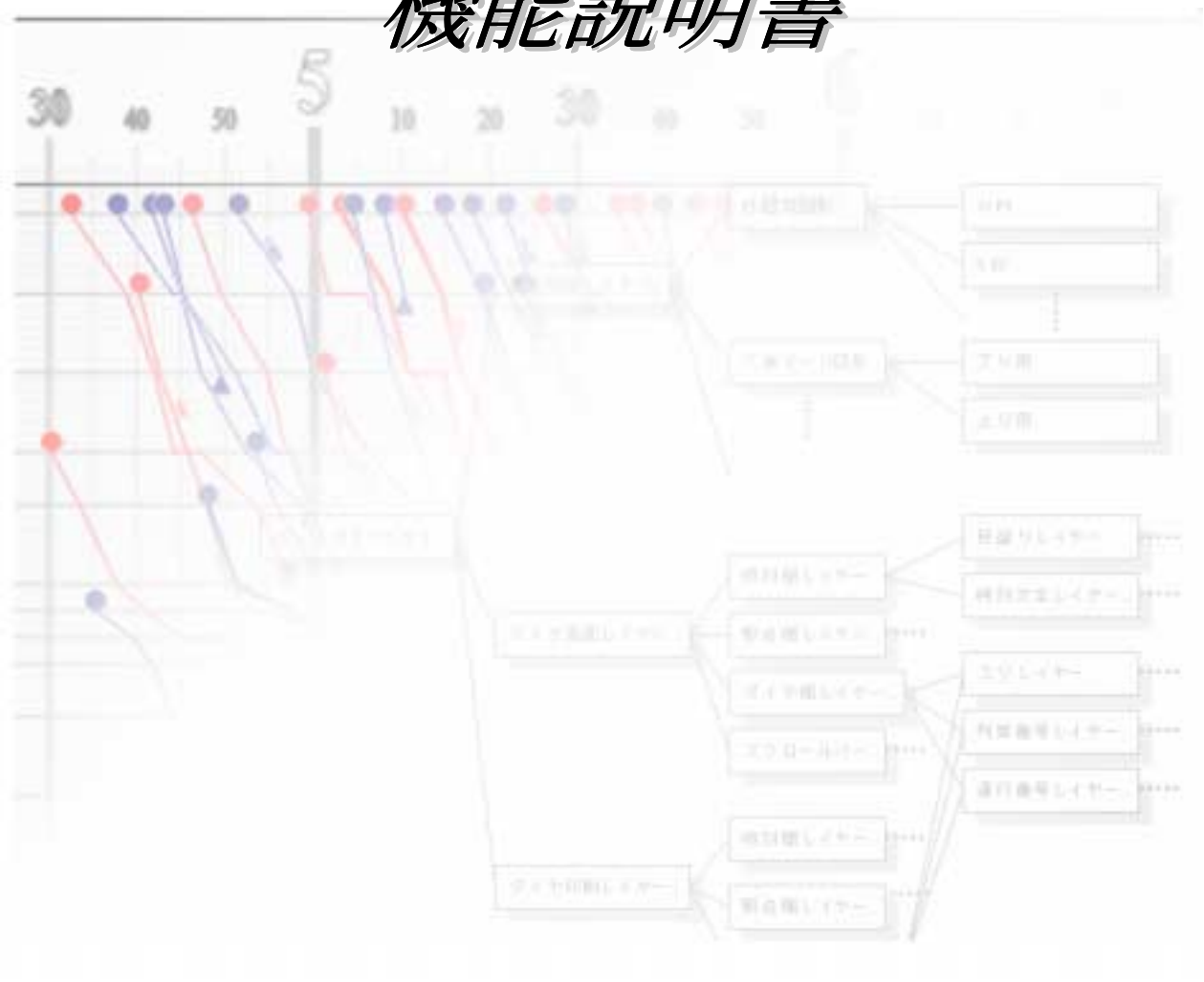
Gpp図面管理ライブラリ

Ver F1.2.0

機能説明書



2009/3/27 Designed by **SunGix**

目　次

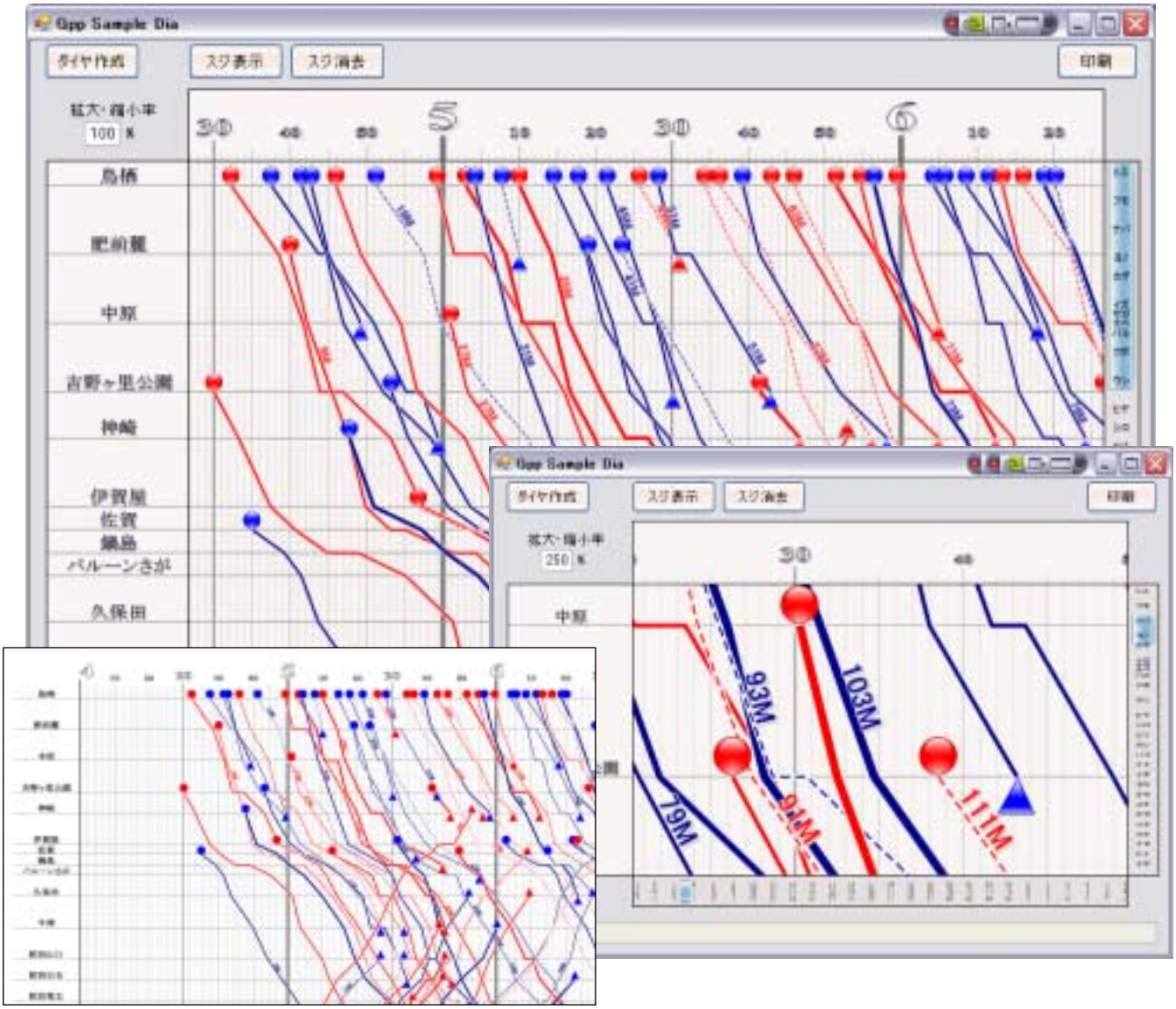
1. はじめに

Gpp(以下 Gpp とは Ver F1.2.0 での説明となります)ToolKit は複雑で広大な図面を画面に表示、またはプリンタやプロッタに印刷する際のデータマネージメントを支援します。

Windows アプリケーションではこれまで、図形や文字を画面やプリンタまたはプロッタ等のデバイスに出力する際、Windows が標準でサポートしているグラフィックスライブラリ(GDI+等)を利用してきました。 これらのグラフィックスライブラリでは、単に図形や文字を描画する基本的な機能しかありませんので、図面全体を管理するプログラムや図形データはアプリケーション毎に設計する必要がありました。

図面の大きさや複雑さによっては図面を管理するプログラムの量も無視できない程になってきますが、Gpp が提供する高度な図面管理支援機能によりこれらのプログラムを省略できます。
言い換えれば Gpp を利用する事でアプリケーションに高度な図面管理機能を容易に実装する事ができます。

Gpp を使った図面の作成例



2. Gpp の図面管理

Gpp は一枚の図面を図形の階層的なオブジェクトの集まりとして管理します。 図形群の上位となる層には図面として画面に表示したり、プリンタに印刷するための図形オブジェクト、または各図面で共通して使用する図形オブジェクトが並ぶ事になります。

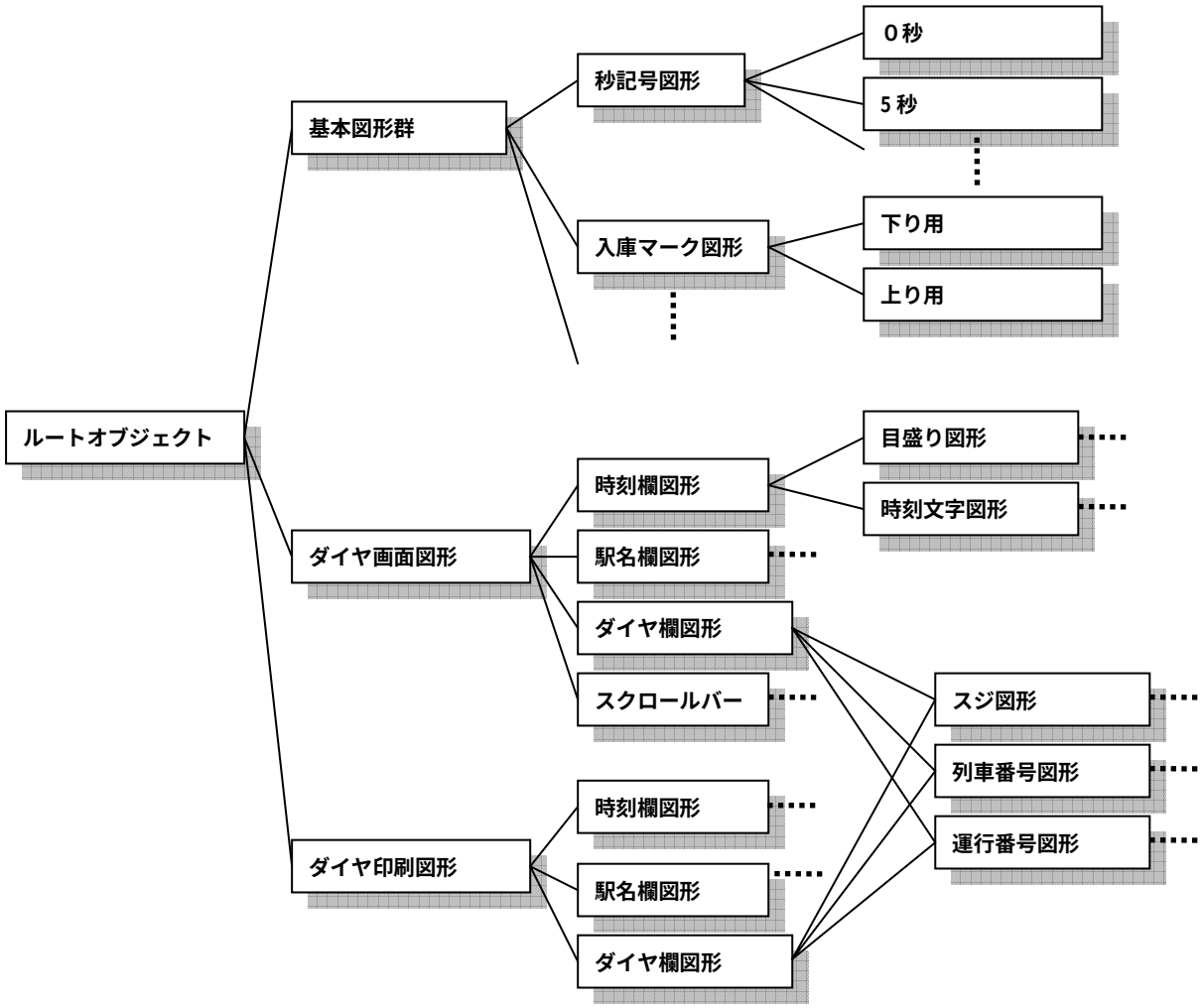
アプリケーションは図形を編集する際に図形オブジェクトのパス名（ファイルパスのように階層分けされた名称の列）を指定できますが、図形に把握しやすい名称を付与する事で、図形データを扱う処理が理解し易く、アプリケーションの機能追加または仕様変更にも柔軟な対応が可能になります。

図形オブジェクトは自身の上位となるオブジェクトと直接結び付けないようになっていますので、一度作成した図形オブジェクトを他の図形オブジェクトの下位オブジェクトとして流用するような図面設計ができ、リソースを最小限に抑える事ができます。

2.1. 図形オブジェクトの階層管理

図形オブジェクトはルートオブジェクトから始まる階層で作成しますが、Gpp の柔軟な図形オブジェクト設計により、アプリケーションが管理し易い構造で配置できます。

図 2-1 ダイヤ図の図形オブジェクト階層例



2.2. 図形オブジェクトの構築

図形オブジェクトの階層は部品ハンドル(GppParts)を使って作成します。
部品ハンドルには図形オブジェクトの階層を構築していくための様々なインターフェースを用意しています。

図 2-1 で示した階層の例において、欄に記載のある図形名称をそのまま図形オブジェクトの名称にできますので、次のように図形オブジェクトを構築できます。

```
// ダイヤ画面図形の図形オブジェクト取得例
GppRoot Root = new GppRoot("ROOT");
GppParts RootParts = new GppParts(Root);
  :
// ダイヤ画面図形オブジェクトを作成し、ルート下に配置
GppObjectFigure DiaScreenFigure = new GppObjectFigure(Root, "ダイヤ画面図形");
RootParts.Add(DiaScreenFigure);
// ダイヤ画面図形の図形ハンドルを作成
GppParts DiaScreenParts = RootParts +  "ダイヤ画面図形";
// 時刻欄図形オブジェクトをダイヤ画面図形下に配置
GppObjectFigure TimeFigure = new GppObjectFigure(Root, "時刻欄図形");
DiaScreenParts.Add(TimeFigure);
 :
// 1M のスジ図形を削除
GppParts DelParts = RootParts + "ダイヤ画面図形" + "ダイヤ欄図形" + "スジ図形" + "1M";
DelParts.Remove();
```

また、下位の図形オブジェクトにアクセスする際は、ファイルパス名のようにオブジェクト名を「.」で区切って羅列する事で深い層の図形オブジェクトにも１回の処理でアクセスできます。

```
// 時刻欄の目盛り作成
GppParts TimeFigure = DiaScreenFigure + "時刻欄図形.目盛り図形";
for( int T = 0; T <= 24; T++)
{
  GppObjectLineFigure Tm = new GppObjectLineFigure();
    :
    : 目盛り線図形オブジェクトを編集
    :
  TimeFigure.Add(Tm);  // 目盛り図形に目盛り線図形を追加
}

// 時刻欄の目盛りの色を変更      インデックス指定でオブジェクトを取得する例です。
GppPen pen = new GppPen(TimeFigure);
for (int T = 0; T <= 24; T++)
{
  GppObject Tm = TimeFigure[T].Renew();
  GppPen pen = new GppPen();
  pen.SetColor("時刻目盛り色");
  Tm.Pen = pen;
}
```

2.3. 図形オブジェクトの XML 入出力

Ver F1.2.0 から実装しましたが、XML 形式で定義した図形オブジェクトを取り込めますので、プログラムの変更無しに図形の変更や追加が行えます。

```
<?xml version="1.0" encoding="Shift_JIS" standalone="yes"?>
<!-- 図形オブジェクトの定義ファイル -->
<parts>
  <pen>
    <!-- 色と線種の定義 -->
    <color name="ダイヤ図欄背景色"  value="250: 245: 245"/>
    <brush name="ダイヤ図欄背景色"  color-value="250: 245: 245"/>
    <color name="警告色"            value="Red"/>
    <color name="時刻線"            value="120: 120: 120"/>
    <color name="30 分目線"          value="150: 150: 150"/>
  </pen>
  <!-- 基本図形作成 -->
  <figure name="マークペン">
    <pen shadow-color-value="200: 200: 200" shadow-offset="0.3"/>
  </figure>
  <figure name="基本図形群">
    <figure name="出庫マーク" pen="マークペン">
      <fill-figure>
        <arc rect="-2,-4, 4,4"/>
      </fill-figure>
    </figure>
    <figure name="入庫マーク" pen="マークペン">
      <fill-figure>
        <polygon point="0,0, -2,4, 2,4"/>
      </fill-figure>
    </figure>
  </figure>
</parts>
```

指定した Xml ファイルに定義されている図形オブジェクトを、図形ハンドルで指定されているオブジェクト下に取り込みます。
また、色や線種等の名前付けされた基本要素も併せて取り込むことが出来ます。

```
// 時刻欄の目盛り作成
GppParts TimeFigure= DiaScreenLayer + "時刻欄図形．目盛り図形";
TimeFigure.InputXmlFile(@"..\..\def\時刻目盛り定義.xml");
:
:
// 時刻目盛り線の色を赤に変更
TimeFigure.InputXml(@"<時刻目盛り <pen color='警告色'/> />");
:
```

このようにプログラム内に埋め込んだ Xml 定義で図形オブジェクトを編集する事も可能です。

XML 定義の図形データは取り込む他に、指定の図形オブジェクト以下を出力する事もできます。

```
GppParts TimeFigure= DiaScreenLayer + "時刻欄図形．目盛り図形";
TimeFigure.OutputXmlFile(@"output-時刻目盛り定義.xml", "Shift_JIS");
```

出力例 （注意 出力した XML には色や線種等の名前付けされた基本要素は出力できません）

```
<?xml version="1.0" encoding="Shift_JIS " standalone="yes"?>
<figure name="text1" rect="10,10,70,35" scale="100%,75%">
  <switch name="スイッチ" layer="9" select-name="Text">
    <pen color-value="255:0:0" />
    <figure name="Fill" location="0,0">
      <pen shadow-color-value="Gray" shadow-offset="1" />
        :
```

2.4. 図形オブジェクトの流用

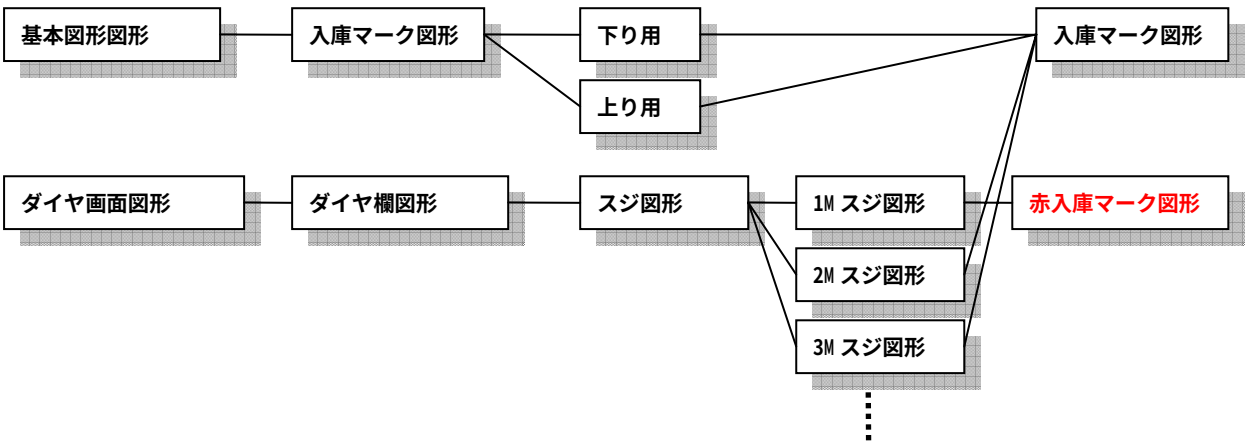
図形オブジェクトは自身と下位オブジェクトを結び付けて構築していきますが、自身と上位図形オブジェクトとは直接結び付けません。 その為、共通して利用する図形オブジェクトを１つ作成しておき、複数の図形オブジェクトから下位オブジェクトとして結び付けさせて流用する事ができます。

また、複数の上位オブジェクトから結び付けられて流用されている図形オブジェクトであっても、指定した図形オブジェクトのみ変更し、その他の流用されている図形オブジェクトを変更しない事も可能です。

```
// 入庫マークの作成
GppParts BasicFigure = new GppParts(Root) +"基本図形群";
GppObject MarkOfGarageIn = (BasicFigure + "入庫マーク").Get();
:
(BasicFigure +"入庫マーク図形.下り用").Add(MarkOfGarageIn); ①
(BasicFigure +"入庫マーク図形.上り用").Add(MarkOfGarageIn); ②
:
// 始発スジに入庫マークを付与
GppParts sujiParts = DiaScreenLayer + "ダイヤ欄図形.スジ図形" + 列車番号;
if (下り列車か？)
  sujiParts.Add((BasicFigure +"入庫マーク図形.下り用.入庫マーク").Get()); ③
else
  sujiParts.Add((BasicFigure +"入庫マーク図形.上り用.入庫マーク").Get()); ④
:
// 1M のスジの入庫マークのみ色替え
GppParts mgiParts = DiaScreenFigure + "ダイヤ欄図形.スジ図形"+ "1M"+ "入庫マーク";
GppObjectFigure mgi = mgiParts.Renew() as GppObjectFigure; ⑤
mgi.Pen.Color = Color.Red; // 赤に変更
```

上記例の場合、①～④までを実行した後の入庫マークの図形オブジェクトの実体は１つのみです。⑤を実行すると、入庫マークの図形オブジェクトは自動的にクローンオブジェクトが作成※1 され、クローンオブジェクトに対して赤色を設定します。

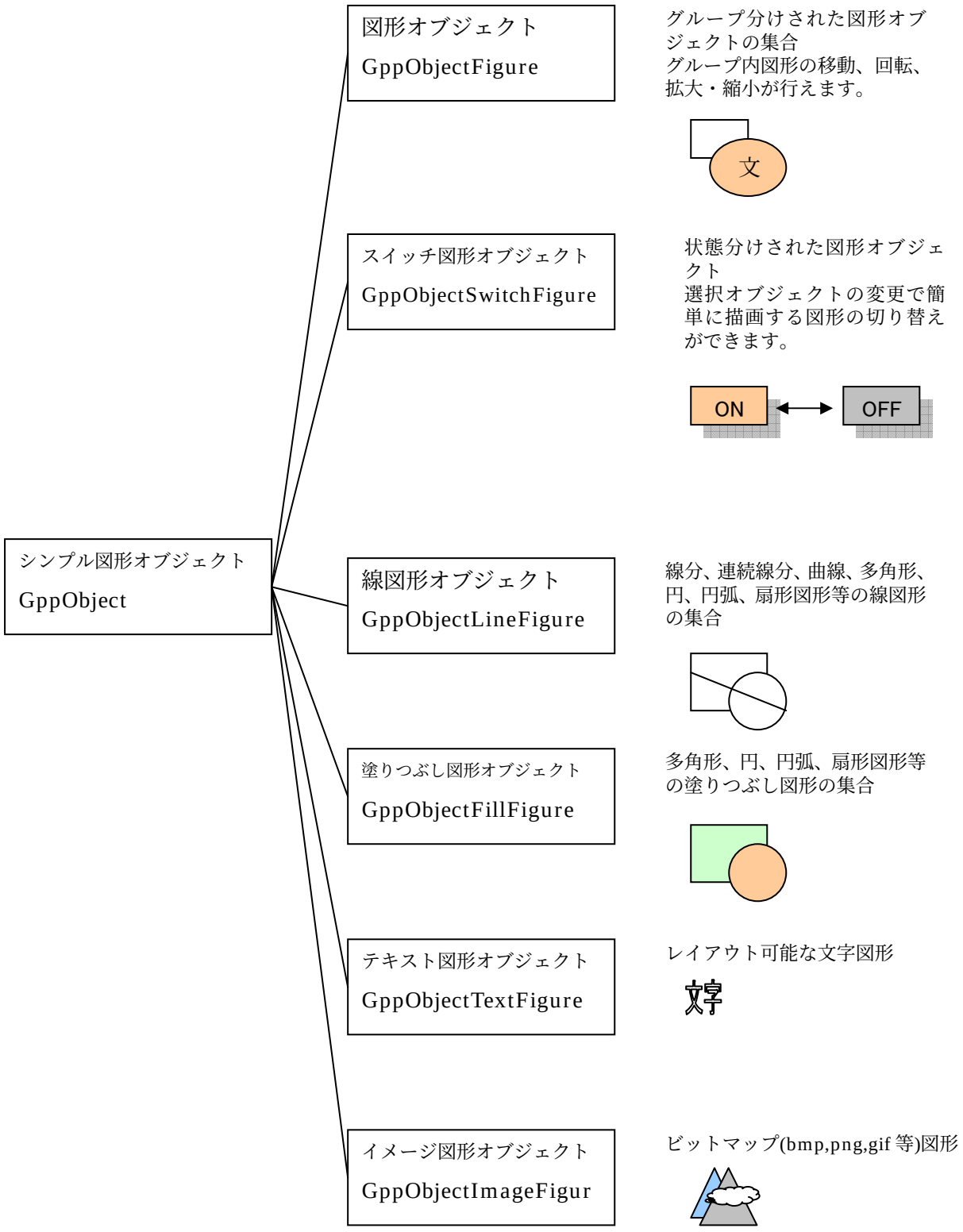
図 2.4-1 ダイヤ図の図形オブジェクト階層例



※1 クローンオブジェクトを作成するか否かの条件は、更新対象の図形オブジェクトに２つ以上の上位オブジェクトから結び付いているか否かで判断します。

2.5. Gpp が管理する図形オブジェクトの種類と機能

図 2.5-1 図形オブジェクトの種類



2.5.1. **図形オブジェクトの共通機能** – GppObject

図形オブジェクトは共通して次のプロパティを持っています。

① ID(int ID)
図形オブジェクトの番号です。 ID は図形オブジェクトが作成される度に新しい番号が付与され、同じルートオブジェクト下にある図形オブジェクトにとってユニークな番号になります。

② 名称(string Name)
任意の名称ですが、同じ上位オブジェクトを持つ図形オブジェクト同士では重複しない名称にする必要が有ります。 また、名称を付与しない図形オブジェクトを作成する事も可能です。

③ 属性情報(object Attribute)
任意の情報を設定・保存できます。 取得した図形オブジェクトが何に属する図形か知りたい時に使用します。

④ 表示フラグ(bool Display)
デフォルトでは true が設定されており、表示対象の図形オブジェクトとなりますが、false を設定する事で自身と下位の図形オブジェクトを表示しません。 図形オブジェクトを一時的に画面から消去したい時に使用します。
消去された図形は抽出図形の対象になりません。(3.8 指定地点の図形抽出機能参照)

⑤ 透明フラグ(bool Transparent)
デフォルトでは false が設定されており、図形オブジェクトは表示されますが、true を設定する事で図形オブジェクトを表示しません。 Display と異なり下位の図形オブジェクトには影響しません。 また、透明状態の図形オブジェクトであっても表示位置を保持していますので、抽出図形の対象となります。(3.8 指定地点の図形抽出機能参照)
図形のどの箇所が選択(マウスクリック)されたか知りたい時に使用します。

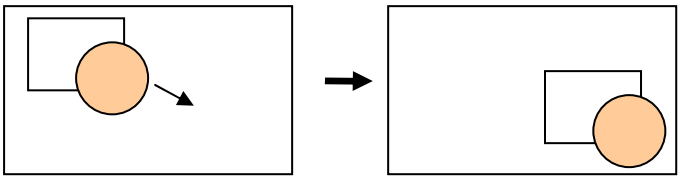
⑥ ペン情報(GppPen Pen)
図形を描画する為の基本要素(色、線種、フォント等)を示す情報です。
描画内容が下位オブジェクトに依存するような図形(図形オブジェクト、スイッチ図形オブジェクト)では、設定されている基本要素を下位の図形オブジェクトに引き継いでいきます。

2.5.2. 図形オブジェクトの機能 – GppObjectFigure

図形オブジェクトは自身の配下に複数の図形オブジェクトを持たせることができ、配下の図形オブジェクトを登録された順番で順に描画します。　また、図形オブジェクトのプロパティ設定により移動、回転、拡大・縮小、及びクリッピング（所定の範囲内の図形を切り取って描画する機能）が行えます。

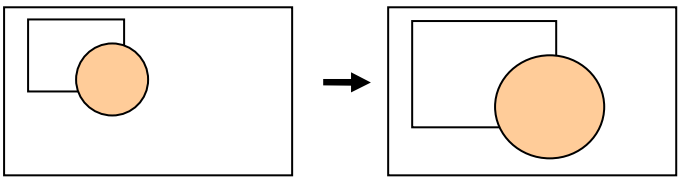
さらに、図形オブジェクト内の図形をビットマップに変換して高速に描画する機能を持っています。

① 配下にある図形オブジェクトは、上位にある図形オブジェクトと相対する位置に描画します。
そのため、上位の図形オブジェクトの座標プロパティ(Location)を変更すると、配下の図形オブジェクトも全て同じ位置に移動させる事ができます。



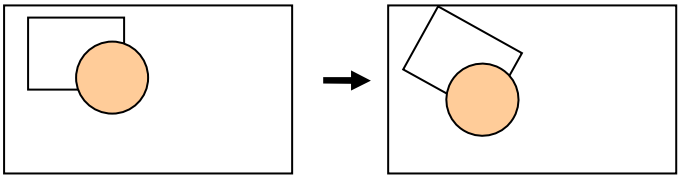
XML での記述例
<図形 location="20.5,10"/>　注意 座標単位は描画ハンドル作成時に指定(ミリメートル等)できます。

② 図形オブジェクトのスケールプロパティ(Scale)値を変更する事で、配下の図形オブジェクトを拡大・縮小して描画します。



XML での記述例
<図形 scale="2.0"/>

③ 図形オブジェクトの回転角度プロパティ(Angle)値を変更する事で、配下の図形オブジェクトを回転して描画します。
回転の中心位置は図形オブジェクトに設定されている矩形※1 の中心になります。

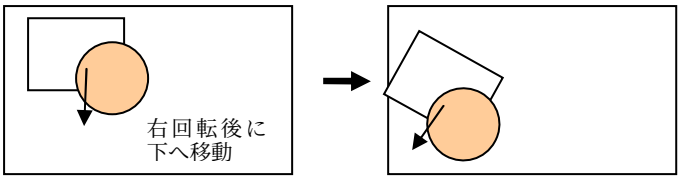


XML での記述例
<図形 angle="45.0"/>

※1 図形オブジェクトが大きさを持たない場合は、図形オブジェクトの原点(0,0)を中心に回転します。

④ 図形オブジェクトの回転後移動プロパティ(OffsetX, OffsetY)値を変更する事で、配下の図形オブジェクトを拡大・縮小し、回転させた後の表示位置の移動※2が行えます。

※2 移動値は座標プロパティ(Location)と異なり、拡大・縮小、回転の影響を受けます。

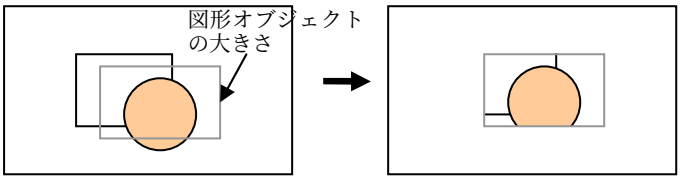


XML での記述例

```
<図形 angle="45.0" offset="0,5"/>
```

⑤ 図形オブジェクトのクリッピングプロパティ(Clipping)を true に設定する事で、図形オブジェクト内の図形を所定の範囲内で切り取って表示させる事ができます。

但し、図形オブジェクトが描画範囲を持つ場合(Width と Height に幅と高さが設定されている時)に限られます。



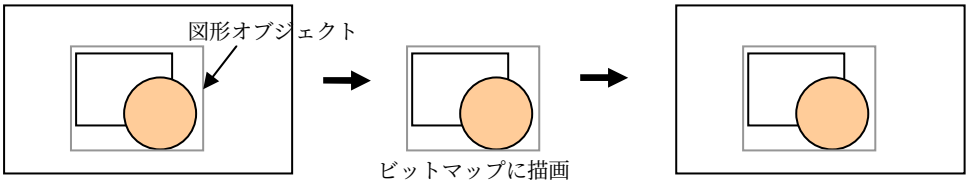
XML での記述例

```
<図形 size="20,10" clipping="true"/>
```

⑥ 図形オブジェクトのバッファリング描画プロパティ(**Buffering**)を true に設定する事で、図形オブジェクト内の図形をビットマップに一旦描画し、ビットマップのイメージデータを画面に転送して高速な描画が行えます。

ビットマップに描画された図形は、図形オブジェクトの描画内容が変更されるまで保持されます。

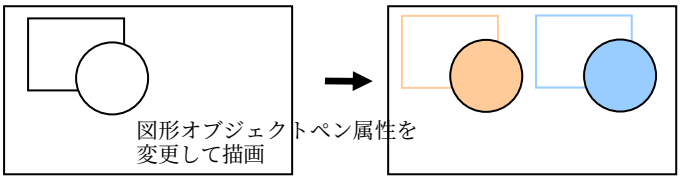
※注意 本機能は後述する Gpp 描画ハンドルのバッファリング描画機能(3.3 バッファリング描画機能参照)が使用されている時のみ有効となります。 プリンタやプロッタへの印刷等で Gpp 描画ハンドルのバッファリング描画機能が使用されていない時は通常の描画を行います。



XML での記述例

```
<図形 buffering="true"/>
```

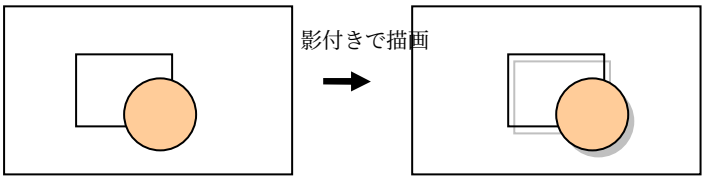
⑦ 下位の図形オブジェクトで、線色、塗りつぶし色、フォント、線種のペン属性が未設定の時は、上位の図形オブジェクトを辿り、設定されている属性があれば適用して描画します。
同じ図形の色を変えて描画したい場合などに使用します。



XML での記述例

```
<figure name="図形データ">
  <line-figure>
    <rectangle rect="0,0,20,15"/>
  </line-figure>
  <fill-figure>
    <arc rect="15,10,10,10"/>
  </fill-figure>
</figure>
<figure "図形">
  <figure name="図形１" location="0,0" color-value="Orange" brush-color-value="Orange">
    <figure add="root.図形データ"/>  ... 色指定が無い図形は全て橙色で描画されます。
  </figure>
  <figure name="図形２" location="30,0" color-value="Blue" brush-color-value="Blue">
    <figure add="root.図形データ"/>  ... 色指定が無い図形は全て青色で描画されます。
  </figure>
</figure>
```

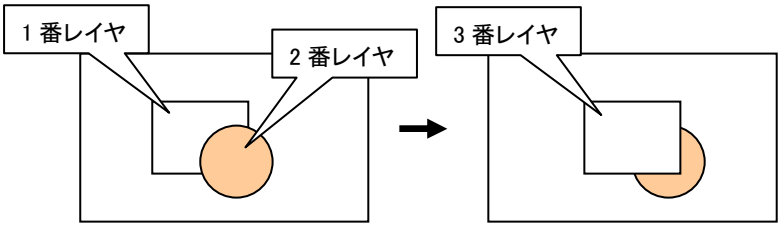
⑧ 図形オブジェクトのペン属性に影付き、または枠付きのオプションが付与されている場合は、図形オブジェクト内の図形全体を影付き、または枠付きで描画します。



XML での記載例

```
<図形 shadow-color-value="Gray" shadow-offset="1.0"/> ... 灰色の影を右下 1.0mm の所に描画します。
<図形 frame-color-value="Red" frame-width="0.5"/> ... 図形の赤い枠線を 0.5mm 幅で描画します。
```

⑨ 図形オブジェクトにレイヤ番号を設定する事で、最大 16 層の表示レイヤに分けて図形を描画できます。 レイヤ番号には 0～15(0 番がデフォルト値です)の番号を設定しますが、番号が大きくなる程上位の層に描画します。 尚、０番は上位オブジェクトのレイヤ番号を受け継ぐ番号です。 上位オブジェクトに５番のレイヤ番号が設定されていれば、レイヤ番号が０番の下位オブジェクトも５番の表示レイヤに描画します。



XML での記述例

```
<figure name="図形１" location="10,10" layer="1">  ... 図形１は１番の表示レイヤに表示
  <line-figure color-value="Black">
    <rectangle rect="0,0,20,15"/>
  </line-figure>
</figure>
<figure name="図形２" location="20,15" layer="2">  ... 図形２は２番の表示レイヤに表示
  <fill-figure brush-color-value="Orange">
    <arc rect="0,0,10,10"/>
  </fill-figure>
</figure>
  :
<図形１ layer="3"/>  ... 図形１を３番の表示レイヤに変更する事で図形１は図形２の上に表示します。
```

注意 バッファリング描画プロパティが true で設定されている図形オブジェクトは下位図形オブジェクトのレイヤ番号を無効としますのでご注意下さい。

2.5.3. スイッチ図形オブジェクトの機能 – GppObjectSwitchFigure

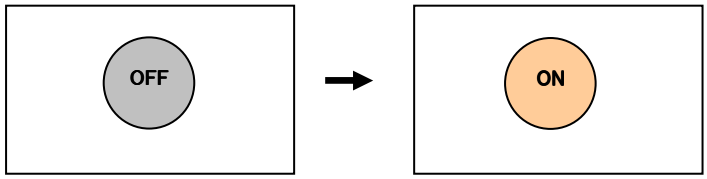
スイッチ図形オブジェクト(GppObjectSwitchFigure)は、個別に定義された図形オブジェクトを複数保持し、予め選択した図形オブジェクトを描画します。

① 描画する図形を選択する際は、番号(SelecctionNo)、または図形オブジェクトの名称(SelectionName)で選択します。

② 下位の図形オブジェクトで、線色、塗りつぶし色、フォント、線種のペン属性が未設定の時は、上位の図形オブジェクトを辿り、設定されている属性があれば適用して描画します。

③ 図形オブジェクトのペン属性に影付き、または枠付きのオプションが付与されている場合は、図形オブジェクト内の図形全体を影付き、または枠付きで描画します。

④ 図形オブジェクトにレイヤ番号を設定する事で、最大 16 層の表示レイヤに分けて図形を描画できます。



XML での記述例

```
<switch name="ボタン" location="30,10" select-name="OFF">  ... 初期表示は OFF 図形
  <figure name="OFF">
    <fill-figure color-value="Gray">
      <arc rect="0,0, 15,15"/>
    </fill-figure>
    <text rect="2.5,2.5, 10, 10"
          font-value=ＭＳゴシック, 2.0, bold, millimeter"
          brush-color-value="Black" string="OFF"/>
  </figure>
  <figure name="ON">
    <fill-figure color-value="Orange">
      <arc rect="0,0, 15,15"/>
    </fill-figure>
    < text rect="2.5,2.5, 10, 10"
          font-value=ＭＳゴシック, 2.0, bold, millimeter"
          brush-color-value="Black" string="ON"/>
  </figure>
</figure>

<ボタン select-name="ON"/>  ... ON 図形に表示を切り替えます。
```

2.5.4. **線図形オブジェクトの機能** – GppObjectLineFigure

線図形オブジェクト(GppObjectLineFigure)は、線で描画する図形データを複数保持します。

① 線図形として描画できる図形には以下のものがあります。
- ・線分（始端座標と終端座標を指定した１本の線分）
- ・連続線分（３点以上の指定した座標を通る連続した線分）
- ・ベジエ曲線（３点以上の指定した座標を通る連続した曲線）
- ・矩形（左上隅の座標と、幅と高さを指定した矩形）
- ・多角形（３点以上の角の座標を指定した多角形）
- ・楕円（指定した矩形に接する円）
- ・円弧（指定した矩形に接し、開始角度から描画角度分の円弧）
- ・扇形（指定した矩形に接し、開始角度から描画角度分の扇形図形）

② ペン属性※1 に線色、線種を設定する事で指定した色と線種で線分を描画します。
線色、線種が設定されていない未設定の状態であれば、上位の図形オブジェクトで設定されている線色、線種を適用※2 します。
※1 ペン属性の詳細は後述します。
※2 上位の図形オブジェクトでも線色、線種が設定されていない時はデフォルトの線色または線種を適用して描画します。

③ ペン属性に塗りつぶしオプションが付与されており、３点以上の座標が設定されている閉図形では指定されている色で塗りつぶします。

④ ペン属性に影付き、または枠付きのオプションが付与されている場合は、線図形全体を影付き、または枠付きで描画します。

XML での記述例

```
<figure name="線図形"
  <pen
    line-style="実線 1mm 幅"
    color="線色"
    fill-brush-value="LinearGradient,45, DarkBlue, White, 0.5, DarkBlue"
    shadow-color="影色" shadow-offset="1.0"
  >
  <line-figure>
    <line location="0,0" >
      <point> 3,1 3,4 9,5 0.01,3.001 </point>
      <point> 4,3 4,5 1,4 8.01,14.001 </point>
    </line>
    <close/>
    <rectangle location="0,0" rect="10,10,20,15" />
    <ellipse location="0,0" rect="11,11,18,13 />
    <polygon point="10,10, 5,15, 10,30, 15,15" />
    <pie angle="45" sweep="180" rect="15,15,20,15" />
  </line-figure>
</figure>
```

2.5.5. 塗りつぶし図形オブジェクトの機能 – GppObjectFillFigure

塗りつぶし図形オブジェクト(GppObjectFillFigure)は、塗りつぶして描画する閉図形データを複数保持します。

① 塗りつぶし線図形として描画できる図形には以下のものがあります。
- ・直線またはベジエ曲線で囲まれた閉図形
  ※注意 閉図形(３点以上の座標が指定され、面を持つ図形）でなければ描画されません。
- ・矩形（左上隅の座標と、幅と高さを指定した矩形）
- ・多角形（３点以上の角の座標を指定した多角形）
- ・楕円（指定した矩形に接する円）
- ・円弧（指定した矩形に接し、開始角度から描画角度分の円弧）
- ・扇形（指定した矩形に接し、開始角度から描画角度分の扇形図形）

② ペン属性※1 にブラシ色を設定する事で指定した色で図形を描画します。
ブラシ色が設定されていない未設定の状態であれば、上位の図形オブジェクトで設定されているブラシ色を適用※2 します。

※1 ペン属性の詳細は後述します。

※2 上位の図形オブジェクトでもブラシ色が設定されていない時はデフォルトのブラシ色を適用して描画します。

③ ペン属性に影付き、または枠付きのオプションが付与されている場合は、塗りつぶし図形全体を影付き、または枠付きで描画します。

XML での記述例

```
<figure name="塗りつぶし図形"
  <pen
    fill-color-value="Blue"
    shadow-color="影色" shadow-offset="1.0"
  >
  <fill-figure
     <line location="0,0" >
       <point> 3,1 3,4 9,5 0.01,3.001 </point>
       <point> 4,3 4,5 1,4 8.01,14.001 </point>
     </line>
     <close/>
     <rectangle location="5,5">
       <rect> 10,10,20,15 </rect>
       <rect> 32,12,5,3 </rect>
     </rectangle>
     <ellipse location="0,-10">
       <rect> 10,10,20,15 32,12,5,3</rect>
     </ellipse>
     <polygon point="10,10, 5,15, 10,30, 15,15" />
     <pie angle="45" sweep="180" rect="15,15,20,15" />
   </fill-figure>
</figure>
```

2.5.6. テキスト図形オブジェクトの機能 - GppObjectTextFigure

テキスト図形オブジェクト(GppObjecTextFigure)は、文字で描画する図形データを１つ保持します。

① テキスト図形として描画できる図形には下記のものがあります。
・テキストの指定した地点を基準として配置した文字の列
指定地点の種類は、上・下・中心、左・右・中心の指定ができます。
・指定した矩形内に配置した文字の列
矩形内に収まらない文字は改行して描画します。

② ペン属性にフォントを設定する事で指定した任意のアウトラインフォントで文字を描画します。
フォントが設定されていない未設定の状態であれば、上位の図形オブジェクトで設定されているフォントを適用※2します。
※1 ペン属性の詳細は後述します。
※2 上位の図形オブジェクトでもフォントが設定されていない時はデフォルトのフォントを適用して描画します。

③ ペン属性にブラシ色を設定する事で指定した色で文字を描画します。
ブラシ色が設定されていない未設定の状態であれば、上位の図形オブジェクトで設定されているブラシ色を適用※3します。
※3 上位の図形オブジェクトでもブラシ色が設定されていない時はデフォルトのブラシ色を適用して描画します。

④ テキスト図形オブジェクトのリージョン描画オプション(Outline)に true を設定する事で、文字のアウトラインを描画します。
この際、ペン属性には線色と線種を設定する必要がありますが、未設定であれば上位の図形オブジェクトで設定されている線色または線種を適用して描画します。
また、ペン属性に塗りつぶしオプションが設定されていれば、線図形オブジェクト同様にアウトライン内を指定色で塗りつぶします。

⑤ ペン属性に影付き、または枠付きのオプションが付与されている場合は、文字全体を影付き、または枠付きで描画します。

⑥ ペン属性の描画方向属性（横書きと縦書き、特に指定しなければ横書き）に従って文字を描画します。

⑦ ペン属性の基準位置属性に従って文字をレイアウトします。
横書きの場合は、中心(Center)、下(Bottom)、上(Top)の中から選択した基準位置に文字を描画します。
縦書きの場合は、中心(Center)、左(Left)、右(Right)の中から選択した基準位置に文字を描画します。

⑧ ペン属性の寄せ位置属性に従って文字をレイアウトします。
横書きの場合は、中心(Center)、左(Left)、右(Right)、均等(Equal, Fequal)の中から選択した寄せ位置に文字を描画します。
縦書きの場合は、中心(Center)、上(Top)、下(Bottom)、均等(Equal, Fequal)の中から選択した寄せ位置に文字を描画します。

XML での記述例

```
<figure name="文字図形"
  <text name="SunGix"
     location="5,30" size="40,5"
     direction="horizontal"
     base-position="bottom"
     side-position="equal"
     color="赤" brush="赤" font="Font"
     string="SunGix"/>
</figure>
```

2.5.7. イメージ図形オブジェクトの機能 – GppObjectImageFigure

イメージ図形オブジェクト(GppObjecImageFigure)は、ビットマップで描画する図形データを１つ保持します。

① ビットマップファイルに保存されているイメージデータを指定した矩形内に描画します。
ビットマップファイルの形式は Bitmap クラスがサポートしている形式であれば種類は問いません。

XML での記述例

```
<figure name="イメージ図形"
  <image name="Image" rect="10,10,20,10" bitmap="image1.gif"/>
</figure>
```

2.6. ペンの属性とオプション – GppPen

ペンは図形を描画する際に適用する線色、線種（実線、破線）、ブラシ（面の色）、フォント等の基本要素情報を保存します。

各要素には名称を設定する事が出来、オブジェクトに色等の要素を適用する際は、「ＸＸＸの色」といった名称を指定して適用させる事ができます。

ペンの情報はルートオブジェクトで一元管理されており、図形を描画する際にどのような基本要素が使用されているかを知る事ができます。　例えば図面全体で使用している全ての色を調べることができます。

2.6.1. 線色属性

線図形を描画する際の線の色として適用します。

```
//色の定義
<pen
  <color name="時刻線" color-value="255:50:50:50"/>
:
// 時刻線定義色の適用
GppPen linePen = new GppPen(Root);
linePen.SetColor("時刻線");
:
// 固定色の適用
linePen.Color = Color.FromARGB(255, 0, 0, 0);
```

2.6.2. ブラシ属性

塗りつぶし図形を描画する際の塗りつぶし色として適用します。

```
// 塗りつぶし色の定義
<pen
  <brush name="入庫マーク用"                    ... 単色塗りつぶし
    value="Solid, 255:0:0:0"/>
  <brush name="出庫マーク用"                    ... グラデーション塗りつぶし
    value="LinearGradient, 90, Black, White"/>  ... 90 度方向に黒から白
:
// 塗りつぶし色の適用
GppPen fillPen = new GppPen(Root);
fillPen.SetBrush("入庫マーク用");
:
// 固定色の適用
fillPen.SetBrush(Color.FromARGB(255, 0, 0, 0));
```

2.6.3. 線種属性

線図形を描画する際の線種として適用します。

```
// 線種の定義(mm 単位)
<pen
  <line-style name="実線" value="solid, 0.5"/>            ... 0.5mm 幅線の実線
  <line-style name="破線" value=" dash, 0.5, 1.0, 1.0"/>  ... 0.5mm 幅の破線(1mm 間隔)
 :
// 線種の適用
GppPen linePen = new GppPen(Root);
linePen.SetLineStyle("実線");
 :
// 固定線種の適用
linePen.SetLineStyle(1.0f, DashStyle.Dash, 1.0f, 2.0f);
```

2.6.4. フォント属性

テキスト図形を描画する際のフォントとして適用します。

```
// フォントの定義
<pen
  <font name="時刻文字用" value="Times New Roman, 9.0, bold, millimeter/>
  <font name="列車番号用" value="ＭＳ Ｐゴシック, 3.0 regular|italic, millimeter/>
 :
// フォントの適用
GppPen fontPen = new GppPen(Root);
fontPen.SetFont(“時刻文字用”);
 :
// 固定フォントの適用
fontPen.SetFont("ＭＳ Ｐ明朝", 3.0f, FontStyle.Bold, GraphicsUnit.Millimeter);
```

2.6.5. 線図形の塗りつぶしオプション

３点以上の座標を持つ線図形オブジェクトは、ペン情報の **SetFillBrush** メソッドで塗りつぶし用のブラシを指定することで、閉図形内を指定したブラシで塗りつぶす事ができます。

```
// 線図形オブジェクトの作成
GppObjectLineFigure LineFigure = new GppObjectLineFigure();
LineFigure.AddPolygon(points);
 :
// 線は赤で、図形内を青で塗りつぶす
GppPen pen = new GppPen(Root);
pen.Color = Color.Red;
pen.SetFillBrush(Color.Blue);
 :
// XML で指定する場合
parts.inputXml(@"<pen fill-color-value='Blue' />");
```

2.6.6. 枠付き描画オプション

図形オブジェクト、線図形オブジェクト、塗りつぶし図形オブジェクト、テキスト図形オブジェクトは、ペン情報の **FrameWidth プロパティと SetFrameColor** メソッドで図形の縁取った枠線を描画できます。

```
// テキストオブジェクトの作成
GppObjectTextFigure TextFigure = new GppObjectTextFigure();
 :
// 赤色文字のテキストに背景色の枠線を付ける
GppPen pen = new GppPen(Root);
pen.FrameWidth = 0.2f;      // 枠線幅は 0.2mm
pen.SetFrameColor("枠線色");

// XML で定義する場合
parts.inputXml(@"<pen frame-width='0.2' frame-color='枠線色' />");
```

2.6.7. 影付き描画オプション

図形オブジェクト、線図形オブジェクト、塗りつぶし図形オブジェクト、テキスト図形オブジェクトは、ペン情報の **ShadowOffsetX、ShadowOffsetY プロパティ、SetShadowColor** メソッドで図形の影を描画できます。

```
// テキストオブジェクトの作成
GppObjectTextFigure TextFigure = new GppObjectTextFigure();
 :
// 赤色文字のテキストに灰色の影を付ける
GppPen pen = new GppPen(Root);
pen.ShadowOffsetX = 0.2f; // 影の位置を右下に描画する
pen.ShadowOffsetY = 0.2f;
pen.SetShadowColor("影色");

// XML で定義する場合
parts.InputXml(@"<pen shadow-offset='0.2' shadow-color='影色' />");
```

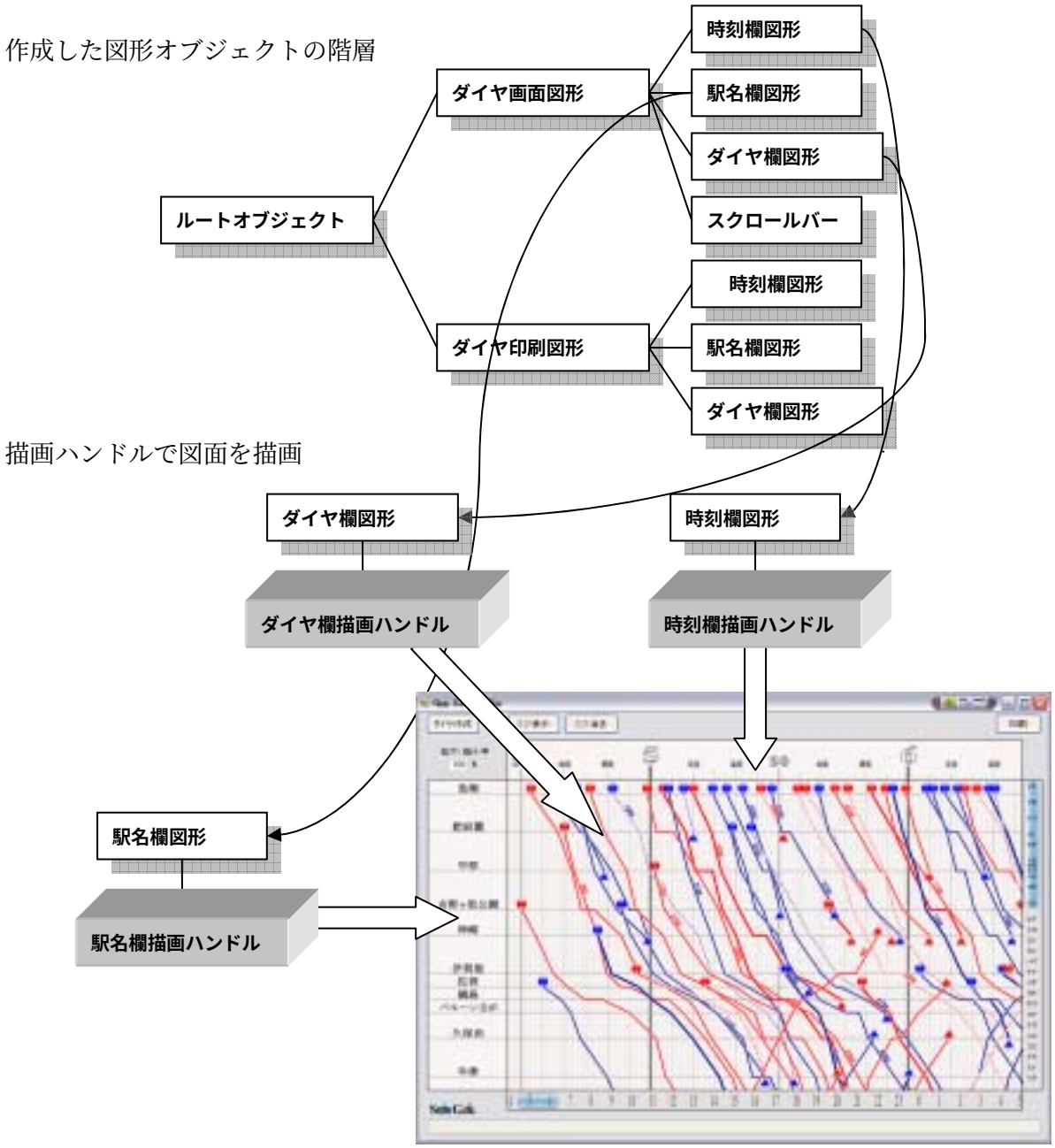
3. Gpp の図面描画 － GppView

部品ハンドル(GppParts)で作成した図形オブジェクトのリソースから、任意の図形オブジェクトを結び付けた描画ハンドル(GppView)を作成することで、その結び付けた図形オブジェクト（トップオブジェクト）から下位となる全ての図形オブジェクトを画面やプリンタに出力します。

描画ハンドルは画面、プリンタ（プロッタ）、ビットマップの各デバイスに対して同じ座標単位で図面を作成できます。 Gpp ではミリメートル単位の座標単位が指定できますが、図面を出力するデバイスが異なっていても座標単位を統一しておくことで同じ大きさの図面を描画します。

描画ハンドルは高速に図面を描画するためのバッファリング描画機能を持っています。 本機能を利用することで、図面のスムーズなスクロールやアニメーションをが可能です。

図 3-1 ダイヤ図の描画例



3.1. 描画ハンドルを使った図面の描画

描画ハンドルは任意に指定した図形オブジェクトをトップオブジェクトとし、その下位となる全ての図形オブジェクトを予め指定したデバイスに出力します。

3.1.1. 図面の画面表示

画面に図形を表示する際は、作成した描画ハンドルとピクチャーボックスから取得したグラフィックスハンドル(Graphics)を結び付け、図形の座標単位を設定します。

```
// ダイヤ画面用の図面を作成する。
GppParts parts = new GppParts(Root);
 :
// ダイヤ印刷用の描画ハンドルを作成する。
GppView viewer = new GppView(parts + "ダイヤ画面図形.ダイヤ欄図形");          ①
// ピクチャーボックスからグラフィックスハンドルを生成する。
Graphics DiaGraph = pictureBox.CreateGraphics();
// 描画ハンドルとピクチャーボックスを結び付け、図形オブジェクトの座用単位を設定する。
viewer.SetDevice(DiaGraph, GraphicsUnit.Millimeter);                          ②
// ピクチャーボックスの全域を表示範囲として設定する。
viewer.SetView(pictureBox);                                                   ③
// ダイヤ画面の背景色を設定する。
viewer.BackgroundBrush = new SolidBrush(Color.White);
 :

// ピクチャーボックスのリペイント処理
private void pictureBox_Repaint(object sender, PaintEventArgs e)
{
  // 表示範囲内を描画する。
  viewer.Draw(e.Graphics, pictureBox);  ④
}
```

① 描画ハンドルを作成する際はトップオブジェクト指定して作成します。

② 描画ハンドルとピクチャーボックスを結び付け、ミリメートルの座標単位を適用します。

③ 描画ハンドルに表示範囲を設定しています。
表示範囲が固定されているピクチャーボックスでは１回のみ設定すれば良いのですが、
画面の大きさに連動して表示範囲が変わるような場合では、変更があったタイミングで表示範囲の再設定が必要になります。
また、パネル内に貼り付けられたスクロール可能なピクチャーボックスでは、表示範囲が上位のパネルの大きさに絞られますので、SetView(panel, pictureBox)のようにピクチャーボックスの上位コントロールも併せて指定します。

④ ピクチャーボックスでのリペイントイベントが発生した時は描画ハンドルの Draw()メソッドを呼び出して表示更新します。

3.1.2. 図面の印刷

プリンタやプロッタ等に図面を印刷する際も画面と同じ様な処理手順になります。

```
// ダイヤ印刷用の図面を作成する。
GppParts parts = new GppParts(Root);
 :
// ダイヤ画面用の描画ハンドルを作成する。
GppView viewer = new GppView(parts + "ダイヤ印刷図形.ダイヤ欄図形");          ①
// PrintDocument クラスのインスタンスを生成する。
PrintDocument doc = new PrintDocument();
doc.PrintPage += PrintPageEventHandler(PrintPage);
 :
// 印刷開始
doc.Print();
 :
doc.Dispose();
// 描画ハンドルを破棄する。
viewer.Dispose();
 :

// ページ印刷イベント処理
private void PrintPage(object sender, PrintPageEventArgs e)
{
  // 描画ハンドルとページを結び付け、図形オブジェクトの座用単位を設定する。
  viewer.SetDevice(e.Graphics, GraphicsUnit.Millimeter);                    ②
  // ページ内の印刷範囲をミリ単位で算出して設定する。
  float PageWidth = (doc.DefaultPageSettings.Bounds.Width / 100.0f) * 25.4f;
  float PageHeight = (doc.DefaultPageSettings.Bounds.Height / 100.0f) * 25.4f;
  RectangleF ViewRect = new RectangleF(0, 0, PageWidth, PageHeight);
  viewer.SetView(ViewRect, GraphicsUnit.Millimeter);                        ③
  // ページ内を描画する。
  viewer.Draw(e.Graphics);    ④
  e.HasMorePages = false;     ⑤
}
```

① 描画ハンドルを作成する際はトップオブジェクト指定して作成します。

② 描画ハンドルと PrintPage イベントが発生した時に渡されるグラフィックスハンドルを結び付け、ミリメートルの座標単位を適用します。

③ 描画ハンドルにページの印刷範囲（ミリ単位に変更）を設定しています。

④ Draw()メソッドを呼び出してページ内を描画します。

⑤ 複数ページを印刷する際は、最終ページ以外では HasMorePages に true を設定してます。
ページ毎の図形の描画は、描画ハンドルをページ毎に作成するか、複数ページ分の図形を作成しておいて、③の処理で印刷する範囲を切り替えるかします。

3.2. 座標単位指定による描画機能

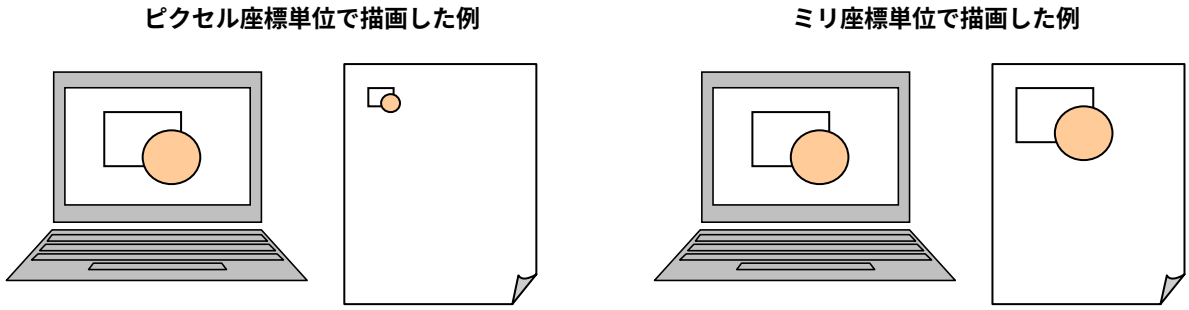
Gpp は図形オブジェクトに設定されている座標値(float)の単位を規定していませんが、図形の大きさを定めるために描画ハンドルとデバイスを結び付ける際、ピクセル座標単位かミリ座標単位かを選択するようにしています。

ピクセル座標単位を指定した際は、出力するデバイスの解像度により図形の大きさが異なってきますが、ミリ座標単位を指定することでどのようなデバイスでも同じ大きさの図形を描画します。

```
// ピクセル座標単位で描画ハンドルを作成
GppView pixviewer = new GppView(parts + "ダイヤ画面図形.ダイヤ欄図形");
pixviewer.SetDevice(e.Graphics, GraphicsUnit.Pixel);

// ミリメートル座標単位で描画ハンドルを作成
GppView mmviewer = new GppView(parts + "ダイヤ印刷図形.ダイヤ欄図形");
mmviewer.SetDevice(e.Graphics, GraphicsUnit.Millimeter);
```

図 3.2-2 座標単位を変えた描画例



3.3. バッファリング描画機能

複雑な図形を画面に表示する際はその複雑さの度合いにより描画に時間が掛かる場合があります。複雑な図形を描いたピクチャーボックスをスクロールさせる場合などは画面がちらつく（書き直しているのが見える）ようになります。

Gpp はこのちらつきを抑えるためにバッファリング描画機能を持っています。 バッファリング描画機能を使用すると、ピクチャーボックスが見えている範囲のビットマップを予め用意し、描画処理が呼ばれると一旦ビットマップに図形を描画し、ピクチャーボックスに描画イメージを転送するようにします。

```
// 描画ハンドルを作成
GppView viewer = new GppView(parts + "ダイヤ画面図形.ダイヤ欄図形");
// バッファリング機能を使って描画するようにデバイスを設定する。
viewer.SetBufferingDevice(e.Graphics, GraphicsUnit.Millimeter);
```

注意 プリンタなどの高解像度のデバイスを使用する際はバッファリングで使用するビットマップのサイズが大きくなり過ぎる場合がありますので、バッファリング機能を使用しないで下さい。

3.4. 図面の移動とスケール変更機能

図形オブジェクト(GppObjectFigure)にも図形の移動やスケールを変更する機能がありますが、描画ハンドルにも同様に図面の表示位置を変更する機能とスケールを変更する機能があります。 図面全体を縦・横にスクロール表示する場合、または図面全体を拡大や縮小して描画する際に使用します。

```
：
// 描画開始位置の補正
viewer.ViewX = 4.5f; // 横位置は 4.5mm 位置から描画
viewer.ViewY = 5.0f; // 縦位置は 5mm 位置から描画
// 1.5 倍に拡大して表示
viewer.ViewScaleX = 1.5f;
viewer.ViewScaleY = 1.5f;
：
```

3.5. 図面の連動描画機能

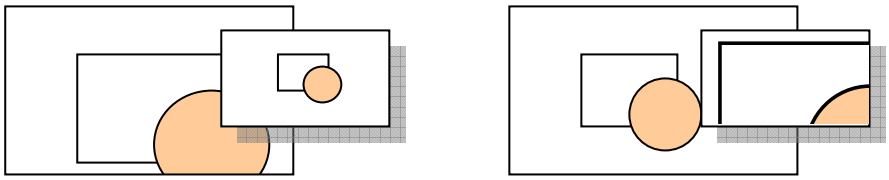
描画ハンドルは複製を作成することができます。 複製した描画ハンドルに別のデバイス（ピクチャーボックス等）を結び付ける事で複数のデバイスに同じ図面を描画できます。

図形オブジェクトは複製元と複製先の両描画ハンドルで共有しますので、図形オブジェクトの設定を変更すると複数デバイスの描画内容を同時に変更できます。

```
：
// 描画ハンドルを複製し、ピクチャーボックス２へ縮小した図形を表示する。
GppView cloneViewer = viewer.New();
cloneViewer.SetDevice(pictureBox2.CreateGraphics(),GraphicsUnit.Millimeter);
cloneViewer.ViewScaleX = 0.25f;
cloneViewer.ViewScaleY = 0.25f;
// 描画
pictureBox2.Invalidate()
：
```

図 3.5-3 連動描画の使用例

子画面で図面全体を縮小表示　　子画面で部分拡大表示



3.6. アニメーション描画機能

部品ハンドル(GppParts)を作成する際にアニメーションを表示したい画面の描画ハンドルを設定しておくと、部品ハンドルから指定された図形オブジェクトの座標、色等の情報を変更することで、その変更された表示内容を即時に画面に反映できます。

画面の表示更新範囲は変更された図形のみに限られますので、描画ハンドルのバッファリング描画機能と併せて使用することで滑らかなアニメーション表示が可能です。

```
 :
// 描画ハンドルを作成する。
GppView viewer = new GppView(TopParts);
viewer.SetDevice(pictureBox.CreateGraphics(), GraphicsUnit.Millimeter);
// 図形ハンドルを作成する。
GppParts parts = new GppParts(viewer);  ①
 :
// 図形の色を変えて表示更新する。
GppObjectFillFigure mark = (parts + "図形群.マーク").Renew() as GppObjectFillFigure;  ②
mark.Pen.Color = Color.Red;
 :
// 表示更新範囲を設定して再描画する。
Region ReDrawRegion = viewer.ReDrawRegion;  ③
if (ReDrawRegion != null)
{
  pictureBox.Invalidate(ReDrawRegion);  ④
  pictureBox.Update();
  ReDrawRegion.Dispose();
}
```

① 描画ハンドルを指定して部品ハンドルを作成します。
これ以降、部品ハンドルから指定されて変更を受けた図形は描画ハンドル側にも変更内容が通知され、次に表示更新した時に変化した図形のみを描画する事ができます。

② 変更する図形オブジェクトは部品ハンドルの Renew()メソッドで取得する事により、図形に変化があることを部品ハンドルに通知します。

③ 表示更新すべき範囲を描画ハンドルの ReDrawRegion プロパティから取得します。
本プロパティは null を返す事もありますが、それは表示範囲に変更された図形オブジェクトが無いことを示します。

④ 表示更新すべき範囲があればピクチャーボックスに **Invalidate** メソッドで表示更新範囲を指示します。　この後でピクチャーボックスのリペイント処理が呼ばれて変更された図形を画面に表示します。
変更した図形を即時に表示したい場合は、**Invalidate** メソッド実行後に同ピクチャーボックスの **Update** メソッドを実行します。

3.7. 省略描画機能

広大な図面に複雑な図形を描画した場合、画面のスクロールや拡大・縮小に時間が掛かる（もたつく）場合があります。　バッファリング機能で描画してもスムーズなスクロールなどが困難な場合は、スクロール途中の表示更新でシンプルな図形のみ描画することでスムーズなスクロール表示に改善できます。

描画ハンドルにはシンプルな図形のみを描画するオプションフラグを持っており、本フラグの設定により幾つかの描画時のオプションを無効にできます。

```
:
// スクロール中は省略描画にする。
viewer.OmittedDraw = GppOmittedFlag.Frame | GppOmittedFlag.Shadow;  ①
pictureBox.Location = new Point(newX, newY);
:
// スクロールが終わった頃に省略描画を通常描画に戻す。
viewer.OmittedDraw = GppOmittedFlag.Non;  ②
pictureBox.Invalidate();
```

① 描画ハンドルの省略描画フラグに枠付き描画と影付き描画を無効にするオプションを付加しています。　これ以降に同描画ハンドルを使用して描画した図形は枠無し、影無しとして描画します。

② 描画ハンドルの省略描画フラグをリセットしています。　これ以降は通常の図形描画を行います。

3.8. 指定地点の図形抽出機能

Gpp はマウスカーソルで指し示したデバイス座標にある図形を取得する機能を持っています。

```
// マウスカーソルボタン押下時イベント処理
private void pictureBox_MouseDown(object sender, MouseEventArgs e)
{
  // マウスカーソル座標にある図形オブジェクトを取得する。
  GppParts parts = viewer.Parts + "ダイヤ欄";
  GppSelectedObject select = GppSelectedObject.LineFigure; // 線図形オブジェクトのみ取得
  GppPartsList partsList = viewer.GetPartsOnPointer(parts, select, new Point(e.X, e.Y));  ①
  // スジの線図形を抽出する。
  foreach (GppParts pickup in partsList)
  {
     if (pickup.Object.Attribute != null)  ②
     {
         // 列車番号を取得する。
         string TrainName = pickup.Object.Attribute as string;
         break;
     }
  }
}
```

① 描画ハンドルの GetPartsOnPointer メソッドで指定座標にある全ての図形を取得します。

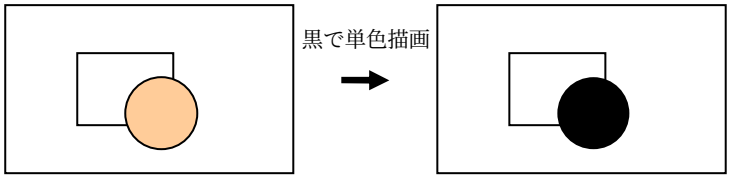
② 図形オブジェクトには object 型の Attribute プロパティが有り、アプリケーション側で自由な値を設定できます。　予め Attribute に意味のある object を設定しておく事で取得した図形が何の図形か判断できます。

※ 表示しない図形オブジェクト(Display=false)は取得しませんが、透明オブジェクト(Transparent=true)は取得できます。

3.9. 単色描画機能

指定した色で全図形を描画します。　印刷用の図面で単色印刷したい場合等で使用します。
指定色描画(3.10 参照)と併用されていない時は、図形に付く枠線を背景色で描画し、図形に付く影を描画しません。

```
// 図面の全図形を黒で描画する。
GppView viewer = new GppView(TopParts);
 :
viewer.SolidColor = Color.Black;
viewer.Draw(e.Graphics);
```

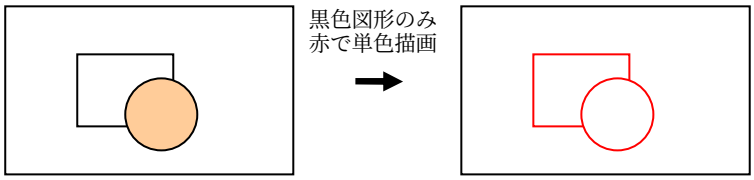


注意 パレット情報が無いイメージ図形は描画できません。

3.10. 指定色描画機能

指定した色の図形のみ描画します。　印刷用の図面で色別原稿を印刷したい場合等で使用します。

```
// 図面の黒色図形を赤で描画する。
GppView viewer = new GppView(TopParts);
 :
viewer.SpecifyColor = Color.Black; r
viewer.SolidColor = Color.Red;
viewer.Draw(e.Graphics);
```



注意 グラデーション塗りつぶし図形や、パレット情報が無いイメージ図形は描画できません。

4. Gpp ロードマップ

Gpp もスタートして約１年が経ち、Xml の入出力機能を加えた Ver F1.2.0 を準備出来ました。 今後も時間が許す限り機能の拡充を進めたいと考えています。
拡充される機能としてその一端をここでご紹介します。

Ver F1.0.0 ・リリース済み
Ver F1.1.0 ・リリース済み
・テキスト描画機能の機能追加（均等割付描画などの文字レイアウト機能）
・スイッチ図形オブジェクトの追加（図形オブジェクトのプロパティ値の変更でオブジェクトの描画内容を切り替える機能です。）
・レイヤ描画機能の追加
Ver F1.2.0 ・リリース済み
・XML 定義による図形オブジェクトの入出力機能を追加
Ver F1.3.0 ・自動レイアウト機能の追加
図形オブジェクト間の相対的な位置関係で図形の座標を動的に求める機能です。
本機能により図面のレイアウト定義を省略化できます。
Ver F1.4.0 ・表機能の追加（表形式の図形オブジェクトで、行・列指定でセル内の図形オブジェクトを操作できます。）
Ver F2.0.0 ・市販アプリケーション用のデータ出力インターフェースの実装
Adobe PDF（既に Adobe Distiller 等を使えば pdf は作成出来ます)
Adobe Illustrator
Excel
その他
Ver F3.0.0 ・3D 描画機能、マルチメディア機能（動画再生）の追加

その他
Ver Cn.n.0 ・C++版の Gpp です。 おそらく Gpp のインターフェースが若干変わります。

Ver En.n.0 ・サーバープロセスとして動作するバージョンです。
画面の構築から図形の描画までを全て１プロセスで行います。
他の制御用プロセスと連携して描画を行いますので、メニーコア環境で威力を発揮します。

Ver Un.n.0 ・UNIX 上で動作する Gpp のバージョンです。 Ver Cn または Ver En から分岐する予定です。