文字の描画

フォントと文字列

今回は、ウィンドウのクライアント領域に文字列を描画してみたいと思います
しかし、GDI+ の文字列関連クラスは予想以上に膨大で複雑になっています
これは、高度なレベルでフォントや文字列の属性を表現できるように設計されているためです

フォントなどの関係があるため、GUI での文字列の描画はコンソールのようにはいきません
これまでの図形メソッドと同じ様に、色や座標などが複雑に関連してくるのです
文字列を描画するには **Graphics.DrawString()** メソッドを使います

```
public void DrawString(
 string s , Font font ,
 Brush brush , PointF point
);
public void DrawString(
 string s , Font font ,
 Brush brush , RectangleF layoutRectangle
);
public void DrawString(
 string s , Font font ,
 Brush brush , PointF point , StringFormat format
);
public void DrawString(
 string s , Font font ,
 Brush brush , RectangleF layoutRectangle , StringFormat format
);
public void DrawString(
 string s , Font font ,
 Brush brush , float x , float y
);
public void DrawString(
 string s , Font font ,
 Brush brush , float x , float y , StringFormat format
);
```

s には描画する文字列を、font には描画に使う Font オブジェクトを指定します
brush には、描画する文字列の色などに使われるブラシを指定します

point は、描画する文字列の左上の座標を示す PointF 構造体を
layoutRectangle には文字列を描画する長方形を示す RectangleF 構造体を指定します
後者を指定した場合、この内部にのみ文字列が描画されます

x と y には、文字列の描画する開始座標を示す X 座標と Y 座標を指定します
format は、描画する文字列の属性を示す StringFormat オブジェクトを指定します

さて、新しい型が登場してきましたが、一つずつ使い方を解説していきましょう
まず、描画するフォントは **System.Drawing.Font** クラスで表されます

```
Object
   MarshalByRefObject
      Font

public sealed class Font : MarshalByRefObject,
 ICloneable, ISerializable, IDisposable
```

このクラスのコンストラクタは、次のようなものが定義されています

```
public Font(Font prototype , FontStyle newStyle);
public Font(FontFamily family , float emSize);
public Font(string familyName , float emSize);
public Font(FontFamily family , float emSize , FontStyle style);
public Font(FontFamily family , float emSize ,  GraphicsUnit unit);
public Font(string familyName , float emSize , FontStyle style);
public Font(string familyName , float emSize , GraphicsUnit unit);

public Font(
 FontFamily family , float emSize ,
 FontStyle style , GraphicsUnit unit
);
public Font(
 string familyName , float emSize ,
 FontStyle style , GraphicsUnit unit
);
```

prototype には元となる既存のフォントを、newStyle には新しい FontStyle を指定します
family には、新しいフォントのフォントファミリを表す FontFamily オブジェクトを指定します
emSize は、フォントのサイズを数値で指定します

familyName にはフォントファミリを表す名前を直接文字列で指定します
例えば "MS ゴシック" や "MS Serif" という形になるでしょう
もちろん、システムがインストールしているフォントでなければなりません

style には、新しいフォントのフォントスタイルを表す FontStyle オブジェクトを
unit は新しいフォントの計測単位を表す GraphicsUnit オブジェクトを指定します

また、ずいぶんと新しい型が現れましたが、この場では省略します
すぐ後で、このクラスについて詳しく解説するので、そこを参照してください
今回は familyName でフォント名を指定する方法を採用します

DrawString() メソッドでは、StringFormat 型を指定することもありましたが

これは System.Drawing.StringFormat クラスを用いています

```
Object
   MarshalByRefObject
      StringFormat

public sealed class StringFormat :
 MarshalByRefObject, ICloneable, IDisposable
```

このクラスは、文字列操作情報や言語特有の性質をカバーするために用います
例えば、右から左に文字が進むヘブライ語やアラビア語なども、これでサポートできます
このクラスのコンストラクタは次のように定義されています

public StringFormat();
public StringFormat(StringFormat format);
public StringFormat(StringFormatFlags options);
public StringFormat(StringFormatFlags options , int language);

format は新しいインスタンスの生成の初期化に
既存の StringFormat を用いたい場合に指定します
options には、StringFormatFlags 列挙型から目的のフラグを指定します
language は、文字列の言語を表す数値を指定します

options で指定するのは System.Drawing.StringFormatFlags 列挙型です
この列挙型は、文字列フォーマットを表すフラグを定義しています

public enum StringFormatFlags

文字列フォーマットを表すメンバは、以下のように定義されています
これを用いれば、日本語の垂直方向の文字列表現なども可能になります

メンバ	解説
DirectionRightToLeft	文字列は右から左へ向かうことを示す
DirectionVertical	文字列は垂直に表示されることを示す
DisplayFormatControl	文字列は左から右へ向かうことを示す
FitBlackBox	制御用長方形から標識がはみ出さないことを示す デフォルトでは、必要に応じて長方形からはみ出ることがある
LineLimit	完全な行だけは、長方形の外に配置されることを示す 長方形は、少なくとも1行分以上の高さを指定しなければならない
MeasureTrailingSpaces	デフォルトでは、制御用長方形を返すと MeasureString() メソッドによって各行の終わりのスペースが除外される そのスペースを測定値に含めるためにこのフラグを指定する
NoClip	制御用長方形からはみ出た標識を表示することを許可する
NoFontFallback	要求されたフォントがサポートされていない場合 代理フォントに頼るものを使用不能にする
NoWrap	長方形の範囲内でフォーマットするならば 行の間を包むことを使用不能にする

以下のプログラムは、最も単純な文字列の描画です

```
using System.Windows.Forms;
using System.Drawing;

class WinMain : Form {
 public static void Main(string[] args) {
  Application.Run(new WinMain());
 }
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Font ft = new Font("Mural Script" , 40);
  PointF pt = new PointF(0 , 0);
  g.DrawString("Kitty on your lap" , ft , Brushes.Black , pt);
 }
}
```

このプログラムでは Mural Script というフォントを用いていることがわかります
もちろん、ここで指定する文字列はインストールされているフォントにしてください

文字列フォーマットを指定すれば、日本語の垂直に進む文字列や
長方形で描画領域を指定した場合の処理などを指定することができるようになります

```
using System.Windows.Forms;
using System.Drawing;

class WinMain : Form {
 public static void Main(string[] args) {
  Application.Run(new WinMain());
 }
```

```
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Font ft = new Font("MS Serif" , 20);
  PointF pt = new PointF(0 , 0);
  g.DrawString("猫耳LOVE" , ft , Brushes.Black , pt ,
   new StringFormat(StringFormatFlags.DirectionVertical));
 }
}
```

このプログラムは、文字列を垂直に表示するように指定してあります
そのため、文字列は原点から下に向かって描画されています

フォントの属性

Font クラスを使いこなすことによって、より柔軟で
演出力のあふれる文字列表現を行うことができるようになります

最も必要とされるのは**フォントスタイル**でしょう
これは、**System.Drawing.FontStyle** 列挙型で定義されています

public enum FontStyle

この列挙型は、次のようなメンバを定義しています
これを、Font クラスのコンストラクタで指定すれば、そのスタイルが適応されます
また、複数のスタイルを論理和で組み合わせて指定することもできます

メンバ	解説
Bold	太字
Italic	斜体
Regular	デフォルトの標準スタイル
Strikeout	打消し文字
Underline	アンダーライン

```
using System.Windows.Forms;
using System.Drawing;

class WinMain : Form {
 public static void Main(string[] args) {
  Application.Run(new WinMain());
 }
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Font ftBold = new Font("MS Serif" , 20 , FontStyle.Strikeout);
  Font ftItalic = new Font(ftBold , FontStyle.Italic | FontStyle.Underline);

  PointF pt = new PointF(0 , 0);
  g.DrawString("Kitty on your lap" , ftBold , Brushes.Black , pt);
  pt.Y += 25;
  g.DrawString("Maiden Breeder" , ftItalic , Brushes.Black , pt);
 }
}
```

このプログラムは、打消し線付きの文字列とアンダーライン付き斜体文字列を描画します
よく使われるテキストの演出ですね

また、通常は文字の大きさで使われる単位はピクセルですが
System.Drawing.GraphicsUnit 列挙型を用いることによって
インチやミリメートルなどの単位で、フォントの大きさを指定できるようになります
印刷関係でよく用いられるポイントなども使えるため、この機能もまた重要です

public enum GraphicsUnit

この列挙型のメンバは、次のようなものが定義されています
これを Font クラスのコンストラクタで指定すれば、計測方法が適応されます

メンバ	解説
Display	1/75 インチ単位

Document	文書単位（1/300インチ）
Inch	インチ単位
Millimeter	ミリメートル単位
Pixel	デバイスピクセル単位
Point	プリンタのポイント単位（1/72インチ）
World	ワールド単位

```
using System.Windows.Forms;
using System.Drawing;

class WinMain : Form {
 public static void Main(string[] args) {
  Application.Run(new WinMain());
 }
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Font ft = new Font("MS Serif" , 1 , GraphicsUnit.Inch);
  PointF pt = new PointF(0 , 0);
  g.DrawString("Kitty on your lap" , ft , Brushes.Black , pt);
 }
}
```

このプログラムは、計測単位をインチ単位の設定しています
そのため、このフォントのサイズは 1 ですが、1 ピクセルと異なり大きいです

前のページへ　戻る　次のページへ