論理ペン

高度なペンの使い方

ペンといえば、文字や絵を書く時に使うペンを思い出しますが
すでに説明したように、GDI+ では線を引く時にシステムが参照するペンがあります
これを**論理ペン**と呼びますが、GUI プログラムでは重要な要素です

漫画家が、どのような線を描くかによってGペンだとかカブラペン、丸ペンというように
状況に応じてペンを選択するように、GDI+ でも描きたい線によってペンを選択します
以前簡単に説明した Pen オブジェクトは、単純にインスタンスを生成しただけでしたが
今回は、より具体的に、このクラスの使い方を説明します

Pen クラスには、以前紹介した2つのコンストラクタ以外に
論理ブラシを用いてペンが塗りつぶす内部を指定することも可能です
以下は、Pen クラスのコンストラクタの全容です

public Pen(Brush brush);
public Pen(Color color);
public Pen(Brush brush , float width);
public Pen(Color color, float width);

brush には、ペンの内部を塗りつぶす Brush オブジェクトを
color には、ペンの色を表す Color オブジェクトを指定します
width には、ペンの太さをピクセル単位で指定します

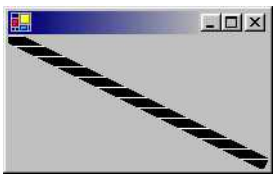
ここで設定した情報は、各種プロパティで取得、再設定することができます
ブラシは **Pen.Brush**、色は **Pen.Color**、
太さは **Pen.Width** プロパティでアクセスすることができます

public Brush Brush {get; set;}
public Color Color {get; set;}
public float Width {get; set;}

Brush はペンが用いているブラシ、Color はペンの色
Width はペンの太さを表すプロパティです

```
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Drawing2D;

class WinMain : Form {
 public static void Main(string[] args) {
  Application.Run(new WinMain());
 }
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Brush myBrush = new HatchBrush(
   HatchStyle.Horizontal  ,
   Color.FromArgb(0xFF , 0xFF , 0xFF)
  );
  Pen myPen = new Pen(myBrush , 10);
  g.DrawLine(myPen , 0 , 0 , 200 , 100);
 }
}
```



このプログラムは、ブラシを用いた太いペンでラインを引いています
単色の代わりに、指定したブラシが使われていることが確認できます

また、ペンで規則的な幾何学模様を描かせたい場合は
Pen.DashStyle プロパティを使うとよいでしょう

public DashStyle DashStyle {get; set;}

このプロパティは、ペンのダッシュスタイルを指定します
これは **System.Drawing.Drawing2D.DashStyle** 列挙型です

public enum DashStyle

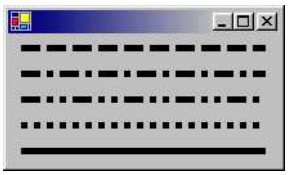
この列挙型には、以下のようなメンバが定義されています
これらのメンバを用いて、ペンのダッシュスタイルを指定します

定数	解説
Custom	ユーザ定義のカスタムダッシュスタイルを示す
Dash	ダッシュを含んだ線を示す

DashDot	ダッシュとドットを繰り返す線を示す
DashDotDot	ダッシュ、ドット、ドットを繰り返す線を示す
Dot	ドットを含んだ線を示す
Solid	切れ目のない通常の線を示す

```
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Drawing2D;

class WinMain : Form {
 public static void Main(string[] args) {
  Application.Run(new WinMain());
 }
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Pen myPen = new Pen(Color.FromArgb(0 , 0 , 0) , 5);
  DashStyle[] ds = {
   DashStyle.Dash , DashStyle.DashDot ,
   DashStyle.DashDotDot , DashStyle.Dot , DashStyle.Solid
  };
  for (int i = 0 , y = 10 ; i < ds.Length ; i++ , y += 20) {
   myPen.DashStyle = ds[i];
   g.DrawLine(myPen , 10 , y , 200 , y);
  }
 }
}
```



これは、定義された DashStyle 列挙型のメンバを用いて
ペンのダッシュスタイルを指定して、繰り返し処理によって各スタイルを列挙したものです

さらに、ダッシュスタイルは**独自のカスタムダッシュ**を定義できます
カスタムダッシュを設定するには **Pen.DashPattern** プロパティを使います

public float[] DashPattern {get; set;}

このプロパティで、ダッシュパターンを取得、及び設定することができます
ダッシュパターンは、float 型の配列で「線 , 間 , 線 , 間」というように構成されます
これをどのような間隔にするかを、カスタムダッシュで設定することができるのです

```
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;
  Pen myPen = new Pen(Color.FromArgb(0 , 0 , 0) , 5);
  float[] fDash = { 5.0f , 1.0f , 1.0f , 1.0f , 3.0f , 1.0f};
  myPen.DashPattern = fDash;
  g.DrawLine(myPen , 10 , 10 , 400 , 10);
 }
}
```

このペンは、5 , 1 , 1 , 3 , 1 という間隔でラインが引かれます
ユーザ定義のダッシュを用いれば、線の間隔は自在に指定することができます

ペン先

通常、線の描き始めや終わり部分は角のついた平坦なものです
しかし、場合によっては丸くしたり、尖らせたりしたいと思うかもしれません

GDI+ ではこれらの要求に応じられるように線の形を定義することができます
これを**キャップスタイル**と呼び、いくつかのプロパティで設定できます
Pen.StartCap プロパティは描き始めのキャップスタイルを
Pen.EndCap プロパティは描き終わりのキャップスタイルを指定することができます

public LineCap StartCap {get; set;}
public LineCap EndCap {get; set;}

このプロパティ型は **System.Drawing.Drawing2D.LineCap** 列挙型です
この列挙型は、定義済みキャップスタイルを提供します

public enum LineCap

この列挙型のメンバは、以下のものが定義されています

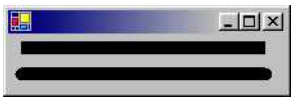
メンバ	解説
AnchorMask	線キャップがアンカー・キャップであるかどうかに関係なくチェックするために使われるマスクを示す
ArrowAnchor	矢形のアンカー・キャップを示す
Custom	ユーザー定義のキャップであることを示す
DiamondAnchor	ダイアモンド形のアンカー・キャップを示す
Flat	平坦なキャップを示す
NoAnchor	アンカーを指定しない
Round	丸いキャップを示す
RoundAnchor	丸いアンカー・キャップを示す
Square	正方形のキャップを示す
SquareAnchor	正方形のアンカー・キャップを示す
Triangle	三角のキャップを示す

これを、Pen のキャップ用プロパティに指定することで
キャップスタイルが変更され、特徴あるペン先を表現することができます

```
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Drawing2D;

class WinMain : Form {
 public static void Main(string[] args) {
  Application.Run(new WinMain());
 }
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Pen myPen = new Pen(Color.FromArgb(0 , 0 , 0) , 10);
  g.DrawLine(myPen , 10 , 10 , 200 , 10);

  myPen.StartCap = myPen.EndCap = LineCap.Round;
  g.DrawLine(myPen , 10 , 30 , 200 , 30);
 }
}
```



このプログラムは、デフォルトのキャップスタイルと
変更したキャップスタイルの違いを確認するためのものです
図の上の線はデフォルト、下は Round メンバのキャップスタイルで描画したものです
ペン先が丸くなり、柔らかい線になっていることが確認できます

キャップスタイルを、切れ目のあるダッシュスタイルと併用する場合
ダッシュの切れ目のキャップを設定したいと考えるかもしれません
この場合は Pan.DashCap プロパティを用いて設定できます

public DashCap DashCap {get; set;}

このプロパティは System.Drawing.Drawing2D.DashCap 列挙形です
この列挙形は、ダッシュの切れ目のキャップを表しています

public enum DashCap

この列挙形のメンバは、以下のようなものがあります

メンバ	解説
Flat	正方形の平坦なキャップを示す
Round	丸いキャップを示す
Triangle	三角形のキャップを示す

```
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Drawing2D;

class WinMain : Form {
 public static void Main(string[] args) {
  Application.Run(new WinMain());
 }
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Pen myPen = new Pen(Color.FromArgb(0 , 0 , 0) , 10);

  myPen.DashStyle = DashStyle.DashDotDot;
  g.DrawLine(myPen , 10 , 10 , 200 , 10);

  myPen.DashCap = DashCap.Round;
  g.DrawLine(myPen , 10 , 30 , 200 , 30);
```

```
  }
}
```

上の線は、ダッシュスタイルだけを設定して描画し
下の線は、さらにダッシュ用のキャップスタイルを設定して描画しています
ダッシュの切れ目に、指定したキャップスタイルが適応されているのがわかります

前のページへ　戻る　次のページへ