# 曲線

## ベジェ曲線

自由な図をプログラムで表現するには**曲線**を扱う必要がありますが
.NET では、プログラムから自由に曲線を表現できる機能を用意しています

CGクリエイターやグラフィックシステムプログラマであれば
ある程度**ベジェ曲線**についてはご存知でしょう
しかし、グラフィカル操作になれないプログラマは馴染みがないかもしれません

Windows は OS レベルでベジェ曲線をサポートしています
これは、CG で用いられる曲線手法の一つでいくつかの座標から曲線を表現します
（たとえば、4つの点から描画するベジェ曲線は「3次ベジェ曲線」と呼ぶ）
このようなパラメータから表現する曲線を**パラメトリック曲線**と呼ぶこともあります

ベジェ曲線の数式を考えたのは **Pierre Bezier** 氏という人物で
他のプラットフォームなどでベジェをサポートする場合、数式を調べるのも面白いかもしれません
高度なグラフィックプログラマを目指すなら、数式も知っておいて損はありません

ベジェ曲線を描くには **Graphics.DrawBezier()** を用います
このメソッドは、4つの頂点から曲線を描きます

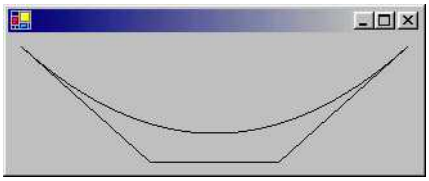
```
public void DrawBezier(
 Pen pen ,
 Point pt1 , Point pt2 , Point pt3 , Point pt4
);
public void DrawBezier(
 Pen pen,
 PointF pt1 , PointF pt2 , PointF pt3 , PointF pt4
);
public void DrawBezier(
 Pen pen ,
 float x1 , float y1 , float x2 , float y2 ,
 float x3 , float y3 , float x4 , float y4
);
```

pen には、曲線を描くペンを表す Pen オブジェクトを指定します
pt1 ～ pt4 には、それぞれ曲線を描くための4つの頂点を指定します
x1、y1 ～ x4、y4 の場合も同様に、float 型でそれぞれ頂点の座標を指定します

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

class WinMain : Form {
 public static void Main(string[] args) {
  WinMain win = new WinMain();
  Application.Run(win);
 }
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Pen myPen = new Pen(Color.FromArgb(0 , 0 , 0) , 1);
  Point[] pt = {
   new Point(10 , 10) , new Point(110 , 100) ,
   new Point(210 , 100) , new Point(310 , 10)
  };

  g.DrawBezier(myPen , pt[0] , pt[1] , pt[2] , pt[3]);
  g.DrawLines(myPen , pt);
 }
}
```



このプログラムは、4つの頂点を利用してベジェ曲線を描画しています
頂点がわかりやすいように、その後 DrawLines() を使って頂点を結んでいます

さらに複雑な曲線をプログラムから描く場合、ベジェ曲線を連続で繋げることができます
これを行うには **Graphics.DrawBeziers()** メソッドを使います

**public void DrawBeziers(Pen pen , Point[] points);**
**public void DrawBeziers(Pen pen , PointF[] points);**

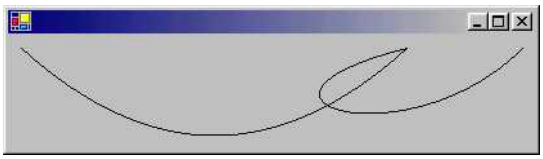
points には、複数の連続した3次ベジェ曲線の頂点を表す
Point または PointF 構造体の配列を指定します
同時に複数の3次ベジェ曲線を描く場合、このメソッドを使うと便利です

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

class WinMain : Form {
 public static void Main(string[] args) {
  WinMain win = new WinMain();
```

```
  Application.Run(win);
 }
 override protected void OnPaint(PaintEventArgs e) {
  Graphics g = e.Graphics;
  Pen myPen = new Pen(Color.FromArgb(0 , 0 , 0) , 1);
  Point[] pt = {
   new Point(10 , 10) , new Point(110 , 100) ,
   new Point(210 , 100) , new Point(310 , 10) ,
   new Point(150 , 50) , new Point(310 , 100) ,
   new Point(400 , 10)
  };

  g.DrawBeziers(myPen , pt);
 }
}
```



図を見てわかるように、このプログラムは2つのベジェ曲線を描いています
この機能を使えば、人間にはなかなか描けない綺麗な曲線を描画できるでしょう

---

## スプライン曲線

.NET はさらに、スプライン曲線をサポートしています
スプライン曲線もまた、頂点の情報を元に曲線を描画するパラメトリック曲線の一つです
この曲線は、カーブの曲がる度合いを設定できるなどの柔軟性があります

スプライン曲線を描くには **Graphics.DrawCurve()** メソッドを使います

```
public void DrawCurve(Pen pen , Point[] points);
public void DrawCurve(Pen pen , PointF[] points);
public void DrawCurve(Pen pen , Point[] points , float tension);
public void DrawCurve(Pen pen , PointF[] points , float tension);

public void DrawCurve(
 Pen pen , PointF[] points ,
 int offset , int numberOfSegments
);
public void DrawCurve(
 Pen pen , Point[] points ,
 int offset , int numberOfSegments , float tension
);
public void DrawCurve(
 Pen pen , PointF[] points ,
 int offset , int numberOfSegments , float tension
);
```

pen には曲線を描くペンを表す Pen オブジェクトを指定します
points には、曲線を定義する各頂点を示す構造体の配列を指定します
配列 points は、最低でも4つ以上の頂点を含んでいなければなりません

tension には、カーブの強さを表す float 型の数値を指定します
この値が 0 である時、線は直線であると考えることができます
このパラメータが指定されていない場合、メソッドは滑らかな曲線を描きます

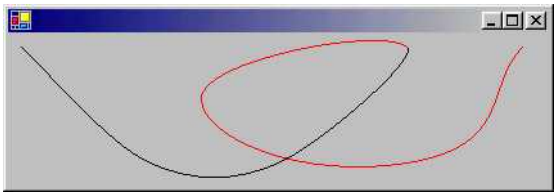
offset には、points から最初に使用する配列のオフセットを指定します
numberOfSegments は offset から数えて、使用する要素の数を指定します
例えば、offset が 2 で、numberOfSegments が 4 ならば
頂点を表す配列 points の、points[2] ～ points[5] 要素を用いて描画されます

```
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;
  Pen myPen = new Pen(Color.FromArgb(0 , 0 , 0) , 1);
  Point[] pt = {
   new Point(10 , 10) , new Point(110 , 100) ,
   new Point(210 , 100) , new Point(310 , 10) ,
   new Point(150 , 50) , new Point(310 , 100) ,
   new Point(400 , 10)
  };

  g.DrawCurve(myPen , pt , 0 , 3 , 0.5f);

  myPen = new Pen(Color.FromArgb(0xFF , 0 , 0) , 1);
  g.DrawCurve(myPen , pt , 3 , 3 , 1.0f);
 }
}
```



このプログラムでは、わかりやすいように2回に分けて DrawCurve() を呼び出しています
最初の DrawCurve() は黒いペンで、2回目は赤いペンで描画しています

---

前のページへ　戻る　次のページへ