イメージの描画

静止画を表示する

ディスクに保存された静止画を描画することができれば、プログラムの可能性は広がります
.NET は、デバイス独立ビットマップや JPEG 等の一般的な静止画を描画する機能を
クラスライブラリで提供しているため、従来よりのずっと簡単に描画することができます

イメージの描画には Graphics.DrawImage() メソッドを用います
このメソッドは、指定した位置に静止画を描画します

```
public void DrawImage(Image image , Point point);
public void DrawImage(Image image , Point[] destPoints);
public void DrawImage(Image image , PointF point);
public void DrawImage(Image image , PointF[] destPoints);
public void DrawImage(Image image , Rectangle rect);
public void DrawImage(Image image , RectangleF rect);
public void DrawImage(Image image , int x , int y);
public void DrawImage(Image image , float x , float y);

public void DrawImage(
 Image image , Point[] destPoints ,
 Rectangle srcRect , GraphicsUnit srcUnit
);
public void DrawImage(
 Image image , PointF[] destPoints ,
 RectangleF srcRect , GraphicsUnit srcUnit
);
public void DrawImage(
 Image image , Rectangle destRect ,
 Rectangle srcRect , GraphicsUnit srcUnit
);
public void DrawImage(
 Image image , RectangleF destRect ,
 RectangleF srcRect , GraphicsUnit srcUnit
);
public void DrawImage(
 Image image ,
 int x , int y , int width , int height
);
public void DrawImage(
 Image image , int x , int y ,
 Rectangle srcRect , GraphicsUnit srcUnit
);
public void DrawImage(
 Image image ,
 Point[] destPoints , Rectangle srcRect ,
 GraphicsUnit srcUnit , ImageAttributes imageAttr
);
public void DrawImage(
 Image image ,
 PointF[] destPoints , RectangleF srcRect ,
 GraphicsUnit srcUnit , ImageAttributes imageAttr
);
public void DrawImage(
 Image image , float x , float y ,
 RectangleF srcRect , GraphicsUnit srcUnit
);
public void DrawImage(
 Image image ,
 float x , float y , float width , float height
);
```

さて、ずいぶん数多くオーバーロードされていますが
この他にも、まだいくつか定義されています
このメソッドの完全な詳細はこの場で解説しきれないので、ヘルプを参照してください

image には、描画する静止画を表す Image オブジェクトを指定します
point にはイメージの左上角を描画する座標を示す構造体オブジェクトを
destPoints は平行四辺形をあらわす3つの頂点の座標を持つ配列を
rect には、イメージを描画する長方形をあらわす構造体オブジェクトを指定します

x と y は、イメージの左上角の X 座標と Y 座標をそれぞれ指定します
srcRect は描画するイメージの元の長方形を指定します
srcUnit は、srcRect で使われる計測単位を指定することができます

imageAttr は、イメージのガンマ情報を指定する ImageAttributes オブジェクトを
width にはイメージの幅、height にはイメージの高さを指定します

これらの中で最も重要なのは、イメージを表す System.Drawing.Image クラスです
このクラスは、静止画の情報をカプセル化するための抽象クラスです

```
Object
   MarshalByRefObject
      Image

public abstract class Image : MarshalByRefObject,
  ISerializable, ICloneable, IDisposable
```

静止画を扱うには、これを継承した System.Drawing.Bitmap クラスを使います
このクラスは、Bitmap という名前になっていますが *.bmp ファイルだけではなく
JPEG や GIF、PNG 等の Windows が扱える一般的なフォーマットをサポートしています

```
Object
```

```
   MarshalByRefObject
      Image
         Bitmap

public sealed class Bitmap : Image
```

このクラスは Image クラスを継承しています
一般的に、コンストラクタでファイル名を指定することで
ディスク上の静止画ファイルの情報を持つ Image オブジェクトを作れます

```
public Bitmap(Image original);
public Bitmap(Stream stream);
public Bitmap(string filename);
public Bitmap(Image original , Size newSize);
public Bitmap(int width , int height);
public Bitmap(Type type , string resource);
public Bitmap(Image original , int width , int height);
public Bitmap(int width , int height , Graphics g);
public Bitmap(int width , int height , PixelFormat format);

public Bitmap(
 int width , int height , int stride ,
 PixelFormat format , IntPtr scan0
);
```

original は、既存の Image オブジェクトから新しいインスタンスを作成します
stream は静止画のデータストリームを、filename はファイル名を指定します
newSize は新しい Image オブジェクトをのサイズを指定します
width と height の場合も同様で、イメージの幅と高さを指定します

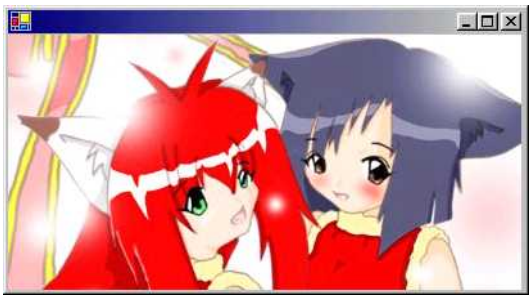
type にはリソースを抽出した Type オブジェクトを
resource にはリソースの名前を指定します
リソースに付いて解説していないので、これについては後記します

g には、解像度を指定するための Graphics オブジェクトを指定します
format はイメージの色データのフォーマットを表す PixelFormat 列挙型のメンバを
scan0 は、ピクセルでータのストリームアドレスを指定します

複雑なコンストラクタについては、この場で説明しきれません
この場では、単純に静止画をディスクから読み取り、それを描画する方法を試してみましょう
ただ単純に、静止画を指定位置に描画したい場合は
Bitmap クラスのコンストラクタでファイル名を指定し、DrawImage() メソッドで描画します

```
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;
  Image img = new Bitmap("test.jpg");
  g.DrawImage(img , 0 , 0);
 }
}
```



このプログラムは、実行ファイルと同じフォルダ内にある test.jpg というファイルを描画します
そこで、筆者は上の図に表示されているような JPEG ファイルを作成して
これを test.jpg という名前で実行ファイルと同じ場所に配置し、プログラムを実行してみました
結果は図の通り、ウィンドウの原点から指定したイメージが描画されます

DrawImage() メソッドは、描画範囲の長方形を指定することで
この長方形にイメージが収まるように、伸縮して描画しようと試みます
この機能を使えば、簡単にイメージを拡大、縮小することができます

```
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;
  Image img = new Bitmap("test.jpg");
  g.DrawImage(img , 0 , 0 , 200 , 100);
 }
```

```
}
```

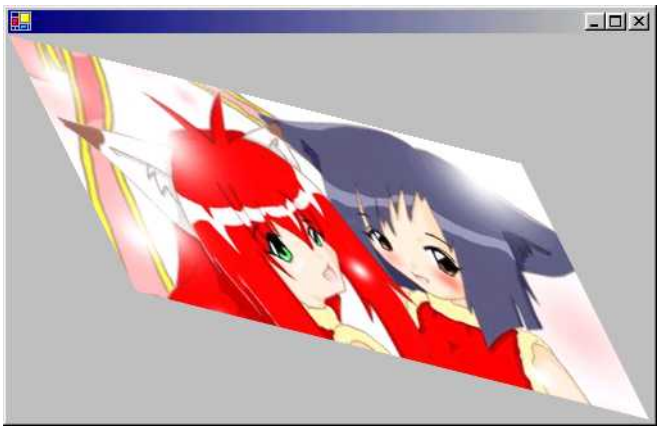
このプログラムは、先ほどのプログラムを少しだけ改良したものです
DrawImage() メソッドのパラメータに注目してください
先ほどは、イメージの左上かどの座標を表す値しか指定していませんでしたが
今回は描画範囲をあらわす幅と高さも指定しています

DrawImage() メソッドで、座標の配列を表す destPoints パラメータを指定すると
指定した3つの頂点を元にした、平行四辺形でイメージを描画します
このシグネチャを使えば、極めて柔軟にイメージを表現することができます

destPoints に指定する配列は、3つの頂点を持たなければなりません
最初の頂点はイメージの左上角、次の頂点はイメージの右上角
そして、最後の頂点は左下角の座標をあらわします
これらの情報から、右下角の座標は推論することができるため、メソッドが計算してくれます

```
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;
  Image img = new Bitmap("test.jpg");
  Point[] pt = {
   new Point(0 , 0) , new Point(400 , 100) ,
   new Point(100 , 200)
  };
  g.DrawImage(img , pt);
 }
}
```



いかがでしょうか、図を見てわかるように 2 組みの向かい合う辺がそれぞれ平行な
指定した座標に基く平行四辺形にイメージがスケーリングされて描画されています

また、ウィンドウに描画する出力の長方形だけではなく
srcRect パラメータを指定することで、静止画の特定の長方形を切り抜くことができます

```
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;
  Image img = new Bitmap("test.jpg");
  Rectangle dest = new Rectangle(0 , 0 , 200 , 200);
  Rectangle src = new Rectangle(100 , 70 , 100 , 100);
  g.DrawImage(img , dest , src , GraphicsUnit.Pixel);
 }
}
```

このプログラムは、先ほどの画像の一部の長方形のみを読み取り
これを拡大して表示しているものです

前のページへ　戻る　次のページへ