第49章 超簡単アニメーション

この章では、イベントを利用した超簡単アニメーションを作ってみます。(アニメーションと言えるかどうかははなはだ疑問ですが・・・)

疑似アニメーションを実現するには、カーソル位置を自由に指定できなくてはいけません。

C#2.0では、Consoleクラスに多数のメンバが追加されました。

この中に、カーソルに関連したプロパティがいくつかあります。

```
public static int CursorLeft { get; set; }
```

カーソルの列位置を取得または設定します。

```
public static int CursorTop { get; set; }
```

カーソルに行位置を取得または設定します。

また、いっぺんにカーソル位置を設定するメソッドも用意されています。

```
public static void SetCursorPosition (
 int left,
 int top
)
```

leftはカーソル位置の列、topは行の位置を指定します。

コンソールウインドウの幅とか高さを調べたり設定するプロパティも追加されました。

```
public static int WindowWidth { get; set; }
public static int WindowHeight { get; set; }
```

コンソールウィンドウ領域の幅、高さを取得または設定します。

カーソルを可視状態かどうかを取得、設定するプロパティもあります。

```
public static bool CursorVisible { get; set; }
```

カーソルを見える状態にするにはtrue,見えなくするにはfalseにします。

取得する場合も同じです。

コンソールの前景色(文字色)を取得、設定するにはForegroundColorプロパティを使います。

```
public static ConsoleColor ForegroundColor { get; set; }
```

ConsoleColorは、System名前空間で定義されているコンソールの色(前景・背景)を表す 列挙体です。メンバと実際の値は次のようになっています。

メンバ名	意味	値
Black	黒	0
Blue	青	9
Cyan	青緑(シアン)	11
DarkBlue	ダークブルー	1
DarkCyan	濃い青緑	3
DarkGray	濃い灰色	8
DarkGreen	濃い緑	2
DarkMagenta	濃い赤紫	5
DarkRed	濃い赤	4
DarkYellow	黄土色	6
Gray	灰色	7
Green	緑	10
Magenta	赤紫	13
Red	赤	12
White	白	15
Yellow	黄色	14

コンソール画面をクリアするには、Clearメソッドを使います。

```
public static void Clear ()
```

注意すべき点は、このメソッドを実行するとカーソル位置が(0,0)に戻ってしまうことです。

このくらいConsoleクラスのメンバを知っていれば、簡単アニメーションが作れます。

繰り返し構文を使って、画面クリアをしてカーソル位置を少しずつずらしながら 文字を表示します。これでアニメーションができます。

しかし、単純にwhile文で繰り返すと速過ぎて見えません。あるいは画面が極端にちらつきます。そこで必ずしも褒められた方法ではありませんが、スレッドを適当な時間休ませます。スレッドの詳しい説明は後の章で行います。

スレッドを休ませるには、ThreadクラスのSleepメソッドを使います。Threadクラスは System.Threading名前空間で定義されています。

```
public static void Sleep (
 int millisecondsTimeout
)
```

millisecondsTimeoutに、スレッドをブロックするミリセコンドを指定します。

では、さっそくプログラムを見てみましょう。

```
// event03.cs

using System;
using System.Threading;

delegate void MyEventHandler(int x, int y, ConsoleColor c);

class MyEvent
{
    public event MyEventHandler myevent;

    public void OnMyEvent(int x, int y, ConsoleColor c)
    {
        if (myevent != null)
            myevent(x, y, c);
    }
}


class ShowCircle
{
    public void show(int x, int y, ConsoleColor c)
    {
        Console.Clear();
        Console.CursorLeft = x;
        Console.CursorTop = y;
        Console.ForegroundColor = c;
        Console.Write("●");
    }
}

class event03
{
    public static void Main()
    {
        int x = 0, y = 0;
        bool dx = true, dy = true;
        ConsoleColor c = ConsoleColor.Black;

        MyEvent me = new MyEvent();
        ShowCircle sc = new ShowCircle();

        me.myevent += new MyEventHandler(sc.show);

        Console.CursorVisible = false;

        while (true)
        {
            Thread.Sleep(30);
             me.OnMyEvent(x, y, c);
            if (Console.KeyAvailable)
                break;
            if (dx)
            {
                x++;
                if (x > Console.WindowWidth - 2)
                {
                    x = Console.WindowWidth - 2;
                    dx = false;
                    c++;
                    if (c > ConsoleColor.Yellow)
```

```
                    c = ConsoleColor.Black;
                }
            }
            else
            {
                x--;
                if (x < 0)
                {
                    x = 0;
                    dx = true;
                }
            }

            if (dy)
            {
                y++;
                if (y > Console.WindowHeight - 1)
                {
                    y = Console.WindowHeight - 1;
                    dy = false;
                }
            }
            else
            {
                y--;
                if (y < 0)
                {
                    y = 0;
                    dy = true;
                }
            }

        }
    }
}
```

MyEventHandlerは、イベントハンドラのデリゲートです。

MyEventクラスがイベントを送信するクラスです。型どおりに書いています。

このクラスのOnMyEventメソッドが呼ばれるとイベントが送信されます。引数を3つ持っている点に注意してください。

ShowCircleクラスは、画面をクリアして指定の位置に●を表示するクラスです。showメソッドが実際に●を表示するメソッドです。

event03クラスがMainメソッドを有するクラスです。

Mainメソッドの中のwhileループを見てください。

これは条件式が常にtrueなので、無限ループになっています。これを抜け出すのは、ユーザーが何かキーを打って、KeyAvailableプロパティがtrueになる時です。

最初に、スレッドを30ミリ秒休ませます。

そして、OnMyEventメソッドを呼んでイベントを送信します。x,yはカーソル位置、cは前景色です。この色で●が描画されます。

次にxやyの値を変更しておきます。

dxがtrueの時はxの値を1つ増やします。xの値が、Console.WindowWidth - 2を超えるときは、xの値をConsole.WindowWidth - 2にして、dxをfalseにします。カーソル位置のx座標は0から始まるので、右端はConsole.WindowWidth - 1となります。しかし、この座標は文字単位なので、コンソールウィンドウの幅によっては中途半端なことになる場合があります。安全を見越してConsole.WindowWidth - 2を採用しています。

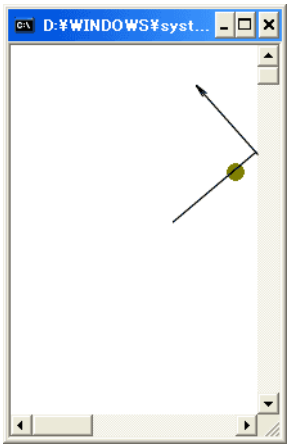
そして、dxをfalseにして●の進むx方向を左向きにします。

この時、前景色の色を変更しています。色が白にならないようにしている点に注意してください。

dxがfalseの時は、xの値を1減らしています。xが負になってしまったときは、xの値を0にして、dxをtrueにしています。

y方向についても同様のことをしています。

これで、コンソール画面を●が動き回るはずです。右の壁に当たるごとに●の色も変わります。何かキーを打つとプログラムが終了します。



単純なアニメーションですが、見ていると結構飽きません。

[C# Index] [総合Index] [Previous Chapter] [Next Chapter]

Update 24/Sep/2006 By Y.Kumei

当ホーム・ページの一部または全部を無断で複写、複製、転載あるいはコンピュータ等のファイルに保存することを禁じます。