


Grafică în C#

Pentru lucrul în mod grafic, C# pune la dispoziția programatorului o clasă numită Graphics. Pentru a transla suprafața de desenare în cadrul ferestrei, se poate folosi un obiect de tip PictureBox, care va fi plasat acolo unde se dorește.

Important! Desenarea în fereastră, într-un PictureBox sau în orice alt control trebuie făcută numai în evenimentul Paint al controlului respectiv. În caz contrar, când fereastra este minimizată sau când controlul este acoperit de alte ferestre, desenul se pierde. 

Evenimentul Paint conține un parametru de tipul:

```
System.Windows.Forms.PaintEventArgs
```

Suprafața grafică a controlului va fi în acest caz e.Graphics, care conține metodele de desenare. Majoritatea controalelor au implementat un eveniment Paint. În afara acestuia, suprafața de desenare poate fi identificată prin metoda CreateGraphics, care returnează un obiect de tip Graphics. În cele ce urmează, vom aminti unele elemente de bază, care pot fi folosite în program ca atare (totuși, pentru aprofundarea acestor cunoștințe, se pot consulta alte manuale sau documentația MSDN). Deoarece majoritatea metodelor sunt supraîncărcate, vom da câte un exemplu simplu de utilizare pentru fiecare situație.

Desenarea unei linii

```
e.Graphics.DrawLine(Pen pen, int x1, int y1, int x2, int y2)
```

Primul parametru precizează culoarea și stilul de desenare a liniei. Se poate declara și folosi un nou obiect de tip Pen (Pen pen = new Pen(...)), însă pentru linii continue și culori predefinite se poate utiliza enumerarea Pens. De exemplu:

```
e.Graphics.DrawLine(Pens.Black, 10, 10, 20, 40);
```

Desenarea unui dreptunghi și a unei elipse

```
e.Graphics.DrawRectangle(Pen pen, int x, int y, int latime, int inaltime)
```

Desenează conturul unui dreptunghi, cu un anumit stil și culoare, folosind coordonatele unui vârf și respectiv lățimea și înălțimea sa. Pentru umplerea unui dreptunghi cu un anumit model și culoare, se folosește metoda:

```
e.Graphics.FillRectangle(Brush b, int x, int y, int latime, int inaltime)
```

Ca și în cazul unui Pen, se poate declara un nou obiect de tip Brush, de exemplu:

```
Brush b = new SolidBrush(Color.Blue);
```

sau se poate folosi enumerarea Brushes, care presupune că stilul de umplere va fi solid (compact).

Un dreptunghi alb cu contur negru se va desena astfel:

```
e.Graphics.FillRectangle(Brushes.White, 0, 0, 10, 20);  
e.Graphics.DrawRectangle(Pens.Black, 0, 0, 10, 20);
```

În mod analog se folosesc metodele DrawEllipse și FillEllipse.

Afișarea unui text în mod grafic

```
DrawString (string s, Font font, Brush brush, int x, int y)
```

Primul parametru este textul care trebuie afișat. Al doilea parametru reprezintă corpul de literă cu care se va desena șirul de caractere. De exemplu:

```
Font font = new Font("Arial", 10);
```

Al treilea parametru este utilizat pentru trasarea efectivă a textului (vezi paragraful despre desenarea unui dreptunghi și a unei elipse). Ultimii doi parametri sunt coordonatele ecran.

Double-buffering automat

Deoarece funcțiile grafice sunt în general lente, pentru desene suficient de complexe sau animații, unde se presupune ștergerea unor elemente și afișarea altora, ecranul pare să clipească (engl. “flickering”). O soluție este desenarea tuturor elementelor într-o zonă de memorie, de exemplu un Bitmap, și apoi afișarea directă a acestuia pe ecran. Această afișare presupune de obicei doar copierea unor informații dintr-o zonă de memorie în alta, fiind deci foarte rapidă. Deoarece se folosesc două zone de memorie, această tehnică se numește *double-buffering*. În anumite situații, se pot folosi mai multe zone de memorie.

În C# se poate face automat *double-buffering*, prin introducerea unei linii precum următoarea (de obicei, dar nu obligatoriu) în constructorul ferestrei:

```
SetStyle(ControlStyles.AllPaintingInWmPaint | ControlStyles.DoubleBuffer |  
ControlStyles.UserPaint, true);
```

Platforma .NET include și proprietatea DoubleBuffered pentru obiectele Form.