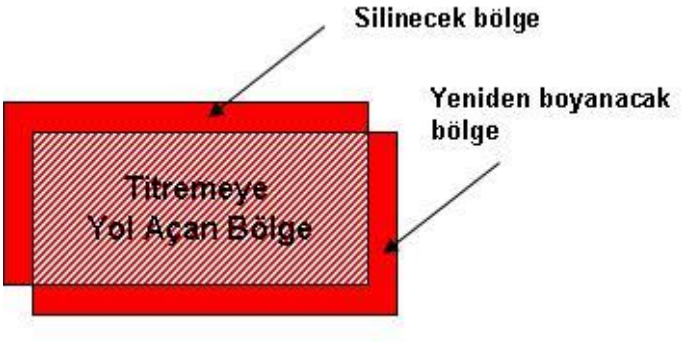
Titreme (Flickering) Problemi

23-Haziran-2009

Bu problemde programcı içi dolu bir şekli taşımak istemektedir. Taşıma sırasında eski şekli silerek yenisini çizer. Fakat şekli taşırken titreme (*flickering*) oluşmaktadır... Bu biçimdeki titreme probleminin nedeni silinecek bölgenin büyük bir kısmının önce zemin rengiyle hemen ardından da asıl renkle boyanmasıdır. (Problem genel olmasına karşın ben burada kod örneklerini *C/C++* ile *API* düzeyinde değil *C#* ile *.NET* üzerinde vereceğim. Hani benim de *C#* kullanmaya gönlüm razı olmuyor. Fakat belki böylece daha fazla kişiye hitap edebilirim :-))

Şimdi *m_rect* isimli içi kırmızıyla boyanmış bir dikdörtgenin fareyle sürükleneceğini düşünelim. Tüm çizimleri *WM_PAINT* mesajında yaptığımızı varsayıyorum. Biz bu durumda eski şeklin silinmesini ve yeni şeklin çizilmesini isteriz değil mi? Bildiğiniz gibi *Paint* mesajı oluştuğunda henüz *OnPaint* metodu çağrılmadan önce *Invalidate* edilmiş alanın zemini silinmektedir (yani *BackColor* ile belirtilen renkle boyanmaktadır). Fakat bu durumda biz kaydırılmış yeni şekli çizdiğimizde eski şekil ile yeni şeklin kesişim bölgesi üst üste iki kez boyanmış olur.



Yani bu kesişim bölgesi önce zemin rengiyle ve hemen sonra da bizim kırmızı renkle boyanacaktır. İşte titremenin nedeni... Peki bunu nasıl engelleyebiliriz? Çok eski çağlardan beri (!) kullanılan tipik yöntem önce ekran dışına (*offscreen*) çizim yapıp sonra sonucu ekrana aktarmaktır. Örneğin, *Paint* mesajında çizim doğrudan pencereye değil de bir *bitmap*'e yapılır. Sonra *bitmap*'te oluşturulan bu görüntü pencereye aktarılır.

Şimdi bu yöntemi uygulayalım:

```
namespace CSD
{
    public partial class Form1 : Form
    {
        private Rectangle m_rect;
        private Point m_prevPoint;
        private Bitmap m_bmp;
        private Graphics m_gbmp;
        private Brush m_backBrush;
```

```
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            m_rect = new Rectangle(100, 100, 200, 100);
            m_bmp = new Bitmap(Screen.PrimaryScreen.Bounds.Width,
                               Screen.PrimaryScreen.Bounds.Height);
            m_gbmp = Graphics.FromImage(m_bmp);
            m_gbmp.Clear(this.BackColor);   // Aslında buna gerek yok :-)
            m_backBrush = new SolidBrush(this.BackColor);
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            m_gbmp.FillRectangle(Brushes.Red, m_rect);

            e.Graphics.DrawImage(m_bmp, e.ClipRectangle,
                          e.ClipRectangle, GraphicsUnit.Pixel);
        }

        private void Form1_MouseMove(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                int dx = e.X - m_prevPoint.X;
                int dy = e.Y - m_prevPoint.Y;

                this.Invalidate(m_rect);
                m_rect.Offset(dx, dy);
                this.Invalidate(m_rect);
                m_prevPoint = e.Location;
            }
        }

        private void Form1_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                m_prevPoint = e.Location;
            }
        }

        private void Form1_BackColorChanged(object sender, EventArgs e)
        {
            m_backBrush = new SolidBrush(this.BackColor);
        }

        protected override void OnPaintBackground(PaintEventArgs e)
        {
            m_gbmp.FillRectangle(m_backBrush, e.ClipRectangle);
        }
    }
}
```

Bazı kütüphanelerde bu işlemi kolaylaştıracak mekanizmalar bulunduruluyor. Örneğin *.NET'te Framework 2.0* ile birlikte bu işlemin *Control* sınıfının *bool* türden *DoubleBuffered* property elemanıyla otomatize edildiğini görüyoruz. Bu *property*'nin

varsayılan değeri *false* biçimdedir ve eğer bu *property true* değerine çekilirse henüz *OnPaint* metodu çağrılmadan önce bir *bitmap* ve ona ilişkin de bir *Graphics* nesnesi yaratılır; *OnPaint* metoduna (dolayısıyla *Paint event* metoduna) da içerisinde bu *Graphics* nesnesi bulunan bir *PaintEventArgs* nesnesi geçirilir. Bu durumda biz *OnPaint* metodunda ya da *Paint event* metodunda çizim yaparken mesaj parametre sınıfına geçirilmiş olan *Graphics* nesnesini kullandığımızda aslında ekrana değil *Framework* tarafından yaratılmış bir *bitmap*'e çizim yapmış oluruz. *OnPaint* metodundan (ya da *Paint event* metodundan) çıkıldığında da bu *bitmap*'in içeriği *Framework* tarafından ekrana kopyalanmaktadır. O halde *Framework 2.0* ve sonrasında tek yapacağınız şey *DoubleBuffered property*'sini *true* değerine çekmek...

```
namespace CSD
{
    public partial class Form1 : Form
    {
        private Rectangle m_rect;
        private Point m_prevPoint;

        public Form1()
        {
            InitializeComponent();
            this.DoubleBuffered = true;
            m_rect = new Rectangle(100, 100, 200, 100);
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Graphics g = e.Graphics;

            g.FillRectangle(Brushes.Red, m_rect);
        }

        private void Form1_MouseMove(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                int dx = e.X - m_prevPoint.X;
                int dy = e.Y - m_prevPoint.Y;

                this.Invalidate(m_rect);
                m_rect.Offset(dx, dy);
                this.Invalidate(m_rect);
                m_prevPoint = e.Location;
            }
        }

        private void Form1_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                m_prevPoint = e.Location;
            }
        }
    }
}
```