Course 2DCis: 2D-Computer Graphics with C#
Chapter C4: The Animation Project

Projekt anim1 mit leerem Fenster

Anleitung für **Visual Studio 2008**

`1)` Main Menu nach dem Start von VS 2008: `File -> New Project... ->`
`Visual Studio installed templates: Windows Forms Application`
`Name: anim1 -> Location: C:\temp -> Create directory for solution:` ausschalten `-> OK`
Es meldet sich `Form1.cs[Design]`.

`2)` Sie müssen zwei überflüssige Files löschen: `Form1.Designer.cs` und `Program.cs`.
Sie erreichen diese Files über das `Solution Explorer - anim1`-Window: Klicken Sie das Pluszeichen vor `anim1` und dann das Pluszeichen vor `Form1.cs`.
Klicken Sie mit der **rechten** Maustaste auf den Ast `Program.cs`. Es öffnet sich ein Kontextmenu. Sie Klicken auf `Delete`. Eine Message Box erscheint: `'Program.cs' will be deleted permanently.` Sie quittieren mit `OK`.
Klicken Sie mit der **rechten** Maustaste auf den Ast `Form1.Designer.cs` und löschen auch dieses File.

`3)` Klicken Sie mit der **rechten** Maustaste auf das graue Fenster `Form1`. Es öffnet sich ein kleines Kontextmenü. Klicken Sie auf `View Code`.
Sie sehen jetzt den vorprogrammierten Code von `Form1.cs`. Löschen Sie den gesamten Code vollständig.

`4)` Schreiben Sie in das vollständig leere File `Form1.cs` folgende 3 Zeilen:

```
public class Form1 : System.Windows.Forms.Form
{ static void Main() { System.Windows.Forms.Application.Run( new Form1() ); }
}
```

`5)` Klicken Sie `Debug` im Main Menu oberhalb des Hauptfensters.
Es öffnet sich ein Untermenü. Klicken Sie auf `Start Without Debugging Ctrl F5`.

Wichtig: Immer zuerst alle Instanzen von `anim1` beenden, bevor neuer Code eingegeben und übersetzt wird !

Malprogramm mit dynamischem Array

Falls Sie den Code nicht mehr sehen, klicken Sie in Visual Studio auf den Karteireiter **`Form1.cs`**, das bringt den Code zum Vorschein.
Sie löschen alles und schreiben in das leere Codefenster `Form1.cs` folgenden Code:

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

public class Form1 : Form
{ public static void Main() { Application.Run( new Form1() ); }
  static Graphics g;
  static Single zoom       = 1.01f;
  static Single cosinus    = (Single)Math.Cos( Math.PI/180.0 );
  static Single sinus      = (Single)Math.Sin( Math.PI/180.0 );
  static Brush  redbrush   = new SolidBrush( Color.Red );
  static Pen    blackpen   = SystemPens.ControlText;
  static Font   arial10    = new Font( "Arial", 10 );
  static Int32  myWidth, myHeight;
  static PointF[] pf;
  ArrayList polygon = new ArrayList();
  Point p0 = new Point();
  Point p1 = new Point();

  public Form1()
  { Text = "Anim1: Draw an Endless Animation";
    Width = 800;
    Height = 600;
    g = this.CreateGraphics();
    SetStyle(ControlStyles.ResizeRedraw,true);
  }
  protected override void OnMouseDown( MouseEventArgs e )
  { polygon.Clear(); Invalidate();
    p0.X = e.X;
    p0.Y = e.Y;
    polygon.Add( p0 );
  }
  protected override void OnMouseMove( MouseEventArgs e )
  { if ( e.Button == MouseButtons.None ) return;
    p1.X = e.X;
    p1.Y = e.Y;
    Int32 dx = p1.X - p0.X;
    Int32 dy = p1.Y - p0.Y;
    if ( dx*dx + dy*dy < 100 ) return;
    g.DrawLine( blackpen, p0, p1 );
    polygon.Add( p1 );
    p0 = p1;
  }
  protected override void OnMouseUp( MouseEventArgs e )
  { if ( polygon.Count < 2 ) return;
    pf = new PointF[polygon.Count];
    for ( Int32 i=0; i < polygon.Count; i++ ) pf[i] = (Point)polygon[i];
  }
  protected override void OnPaint( PaintEventArgs e )
  { e.Graphics.DrawString( "Press the left mouse button and move!", Font,
                           redbrush, Width/2-50, 0 );
  }
  protected override void OnResize( System.EventArgs e )
  { g = this.CreateGraphics();
    g.Clear( SystemColors.Control );
    myWidth  = ClientRectangle.Width;
    myHeight = ClientRectangle.Height;
  }
}
```

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`.
Erproben Sie das Programm.

Timer

Version2: Beenden Sie Ihr Programm `anim1`.
Schreiben Sie in den Kopf von `Form1` unterhalb der Zeile `Point p1 = new Point();`
folgende weitere Zeile:

```
  Timer myTimer = new Timer();
```

Schreiben Sie in den Konstruktor `public Form1()` unterhalb der Zeile
`SetStyle(ControlStyles.ResizeRedraw,true);` folgende zwei Zeilen:

```
    myTimer.Tick += new EventHandler( OnTimer );
    myTimer.Interval = 1;
```

Schreiben Sie als **ersten** Befehl der Funktion `protected override void OnMouseDown(`
`MouseEventArgs e )`:

```
    myTimer.Stop();
```

Schreiben Sie als **letzten** Befehl der Funktion `protected override void OnMouseUp(`
`MouseEventArgs e )`:

```
    myTimer.Start();
```

Schreiben Sie eine neue Funktion `protected static void OnTimer(...)` hinter die Funktion
`protected override void OnResize( SystemEventArgs e )`,
aber noch vor die letzte geschweifte Klammer, die `Form1` abschließt:

```
  protected static void OnTimer( Object myObject, EventArgs myEventArgs )
  { Single x, y, xmin, ymin, xmax, ymax, xmid, ymid;
    xmin = xmax = pf[0].X;
    ymin = ymax = pf[0].Y;
    for ( Int32 i=0; i < pf.Length; i++ )
    { x = pf[i].X;
      y = pf[i].Y;
      if ( x < xmin ) xmin = x;
      if ( x > xmax ) xmax = x;
      if ( y < ymin ) ymin = y;
      if ( y > ymax ) ymax = y;
    }
    xmid = (xmin+xmax) / 2;
    ymid = (ymin+ymax) / 2;
    if ( xmin < 0 || ymin < 0 || xmax > myWidth || ymax > myHeight )
    { g.Clear( SystemColors.Control );
      g.DrawString( "Press the left mouse button and move!",
                    arial10, redbrush, 320, 0 );
      zoom = 0.99f;
    }
    if ( xmax - xmin < 50 || ymax - ymin < 50 )
    { g.Clear( SystemColors.Control );
      zoom = 1.01f;
    }
    for ( Int32 i=0; i < pf.Length; i++ )
    { x = pf[i].X - xmid;
      y = pf[i].Y - ymid;
      x *= zoom;
      y *= zoom;
      Single xx = x*cosinus - y*sinus;
      Single yy = x*sinus + y*cosinus;
      pf[i].X = xx + xmid;
      pf[i].Y = yy + ymid;
    }
    g.DrawLines( blackpen, pf );
  }
```

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`.
Erproben Sie die Animation. Sie können nacheinander beliebig viele Figuren malen.

Flickerfrei

Wenn Sie kein Linienbündel, sondern nur einzelne bewegte Linien sehen wollen, dann schreiben Sie in die zweitletzte Zeile der `OnTimer`-Funktion **vor** `g.DrawLines( blackpen, pf );` folgende Zeile: `g.Clear( SystemColors.Control );`. Sie löschen damit die Client Area.
Jetzt tritt aber (vor allem bei langsamen Graphikkarten) ein Flickereffekt auf.
Der Effekt stört stärker, wenn die Linie dick ist.
Verdicken Sie dazu `static Pen blackpen = new Pen( Color.Black, 20 );`
Der Löschvorgang unterbricht die ruhige Bewegung, ein generelles Problem jeder Verschiebung von Graphikobjekten.
Abhilfe durch Double Buffer: Man löscht ein und zeichnet in ein unsichtbares Hintergrundbild und kopiert dieses schließlich en bloc in die Client Area.
Anleitung: Deklarieren Sie ein zweites `Graphics` Objekt im Kopf von `Form1` in der Zeile `static Graphics g;` und ein `Bitmap` Object

```
  static Graphics g, bitmap_g;
  static Bitmap bitmap;
```

Erstellen Sie je eine neue Instanz am Ende der Funktion `OnResize`:

```
    if ( bitmap != null ) bitmap.Dispose();
    bitmap = new Bitmap( myWidth, myHeight );
    if ( bitmap_g != null ) bitmap_g.Dispose();
    bitmap_g = Graphics.FromImage( bitmap );
```

Ersetzen Sie die letzten beiden Zeilen in der Funktion `OnTimer`: `g.Clear(SystemColors.Control);` und `g.DrawLines( blackpen, pf );` durch:

```
    bitmap_g.FillRectangle( SystemBrushes.Control, 0, 0, myWidth, myHeight );
    bitmap_g.DrawLines( blackpen, pf );
    g.DrawImage( bitmap, 0, 0 ); //en bloc transfer
```

Falls es noch übel flickert, kontrollieren Sie, ob beide Zeilen `g.Clear( SystemColors.Control );` und `g.DrawLines( blackpen, pf );` am Ende der `OnTimer`-Funktion entfernt sind.
Falls ja, tritt flickern nur noch am Anfang und Ende jeder Bewegung auf. Um auch das zu beseitigen, müssen Sie nur die beiden Befehle `g.Clear( SystemColors.Control );` in den beiden `if`-Klammern der `OnTimer` Funktion durch Kommentarzeichen `//` still legen.

Weitere Aufgaben

Klicken Sie auf `Help` in der Menüleiste von Visual Studio. Klicken Sie auf das Untermenü `Index`.
Gehen Sie in das Feld `Filtered by:` und wählen Sie dort `.NET Framework SDK`.
Dann geben Sie im Feld `Look for:` folgende Schlüsselworter ein:
`Timer class (System.Windows.Forms), Control.OnResize method (System.Windows.Forms), Math.Cos, Math.Sin, Math.PI, static keyword C#, Graphics.Clear`. Lesen Sie die Hilfetexte.
Die Hilfetexte überdecken Ihren Code. Wenn Sie mit dem lesen fertig sind, entfernen Sie die Hilfetexte mit deren X-Button in deren rechter oberer Fensterecke.

Malen Sie eine Figur in die rechte untere Fensterecke. Ziehen Sie zur Laufzeit am Fensterrand und beobachte Sie das adaptive Verhalten der Animation auf die wechselnde Fenstergröße. Verstehen Sie die dabei Wirkung der beiden Variablen `myWidth` und `myHeight`.
Bremsen Sie die Geschwindigkeit der Animation durch hochsetzen des `Timer.Interval` auf `10, 100, 1000`.
Erhöhen Sie die Geschwindigkeit der Animation durch Erhöhen der Schrittweite von 1 auf 2, 4, 6 Grad.
Vergrößern Sie die Abbruchbedingungen durch verkleinern (auch negativ !) von `xmin` und `xmax` und vergrößern von `ymin` und `ymax`.

Beenden Sie Visual Studio, starten sie den Explorer, löschen Sie die gesamte Directory `C:\temp\anim1`
Erfinden und erproben Sie neue Varianten des Programms in Form von neuen Projekten `anim2, anim3` usw. nach obigem Muster.

Miniprogramm zum Training

MiniAnim:

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Collections;
public class Form1 : Form
{ public static void Main() { Application.Run( new Form1() ); }
  static Graphics g;
  static Single cosinus    = (Single)Math.Cos( Math.PI/45.0 );
  static Single sinus      = (Single)Math.Sin( Math.PI/45.0 );
  static Pen    blackpen   = SystemPens.ControlText;
  static PointF p0 = new PointF( 400f, 200f );
  static PointF p1 = new PointF( 200f, 400f );
  static PointF p2 = new PointF( 600f, 400f );
  static PointF[] tri = { p0, p1, p2, p0 };//triangle
  static Rectangle cr;
  Timer myTimer = new Timer();
  public Form1()
  { Text = "MiniAnim";
    Width = 800;
    Height = 600;
    myTimer.Tick += new EventHandler( OnTimer );
    myTimer.Interval = 1;
    myTimer.Start();
  }
  protected override void OnResize( System.EventArgs e )
  { g = this.CreateGraphics();
    cr  = ClientRectangle;
    tri[0] = tri[3] = p0; tri[1] = p1; tri[2] = p2;//forget the past
  }
  protected static void OnTimer( Object myObject, EventArgs myEventArgs )
  { Single x, y, dx, dy;
    dx = p0.X - tri[0].X;
    dy = p0.Y - tri[0].Y;
    if ( (dx*dx + dy*dy) < 1f ) //near start ?
    { g.Clear( SystemColors.Control );
      g.DrawEllipse( new Pen( Color.Red, 5 ), cr.Width/2, cr.Height/2, 10, 10 );
      sinus *= -1f; //reverse
    }
    for ( Int32 i=0; i < 4; i++ )
    { x = tri[i].X - cr.Width/2;
      y = tri[i].Y - cr.Height/2;
      tri[i].X = x*cosinus - y*sinus + cr.Width/2;
      tri[i].Y = x*sinus + y*cosinus + cr.Height/2;
    }
    g.DrawLines( blackpen, tri );
  }
}
```