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

Project anim1 with an empty window

Guidance for **Visual Studio 2008**:

1) Main Menu after start of VS 2005: `File -> New Project... ->`
`Visual Studio installed templates: Windows Forms Application`
`Name: anim1 -> Location: C:\temp -> Create directory for solution:` switch off `-> OK`
`Form1.cs[Design]` appears.

2) Two superfluous files must be deleted: `Form1.Designer.cs` and `Program.cs`.
You reach these files via the `Solution Explorer - anim1`-window: Click the plus-sign in front of branch `anim1` and the plus-sign in front of branch `Form1.cs`.
Right-click the branch `Program.cs`. A context menu opens. Click `Delete`. A message box appears:
`'Program.cs' will be deleted permanently.` Quit with `OK`.
Right-click the branch `Form1.Designer.cs` and delete this file too.

3) Right-click the gray window `Form1`. A small context menu opens. Click `View Code`.
You see now the preprogrammed code of `Form1.cs`. Erase this code completely.

4) Write the following three lines into the empty `Form1.cs`:

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

5) Click `Debug` in the main menu of VS 2005.
A submenu opens. Click `Start Without Debugging Ctrl F5`.

Important: Always finish all instances of `anim1` before writing new code and starting it !

Scribble program with dynamic array

Write the following lines into the empty window of `Form1.cs`:

```
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;
  }
}
```

Click `Debug` -> `Start Without Debugging Ctrl F5`. Try out the program.

Timer

Version2: Finish `anim1`.
Write the following line into the head of `Form1` below the existing line `Point p1 = new Point();`:

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

Write the following two lines into the Constructor `public Form1()` below the existing line `SetStyle(ControlStyles.ResizeRedraw,true);`:

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

Write as **first** line of the function `protected override void OnMouseDown( MouseEventArgs e )`:

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

Write as **last** line of the function `protected override void OnMouseUp( MouseEventArgs e )`:

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

Write a new function `protected static void OnTimer(...)` below the function `protected override void OnResize( SystemEventArgs e )`, but in front of the last brace, that closes `Form1`:

```
  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 );
  }
```

Click `Debug` -> `Start Without Debugging Ctrl F5`. Try out the animation by drawing several polygons.

Flicker free

If You want to replace the bundle of polygons by just a single polygon at a time,
then write a new line as second last line of the `OnTimer`-function
in front of `g.DrawLines( blackpen, pf );`:
`g.Clear( SystemColors.Control );`. //erases the client area.
This normally produces (with slow graphic boards) an obvious flicker .
The effect grows when You make the lines thicker .
For demonstration thicken Your pen: `static Pen blackpen = new Pen( Color.Black, 20 );`
Explanation: The `Clear`-command interrupts the smooth movement. This is a central problem of any displacement of graphic objects.
Remedy: Double Buffering: Clear and draw an invisible background image and
copy it finally en bloc into the client area.
Guidance: Declare a second `Graphics` object in the head of `Form1` in the line `static Graphics g;` and a `Bitmap` Object.

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

Create a new instance of these objects before leaving the function `OnResize`:

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

Replace the two last lines of function `OnTimer`: `g.Clear(SystemColors.Control);` and `g.DrawLines( blackpen, pf );` by:

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

If the program still flickers, make sure that the lines `g.Clear( SystemColors.Control );` and `g.DrawLines( blackpen, pf );` at the end of function `OnTimer`-Funktion have been removed.
At the start and end of any movement flickering still occurs. In order to suppress it completely,
cripple the lines `g.Clear( SystemColors.Control );`
inside both `if`-clauses of the `OnTimer` function by comment slashes `//`.

Exercises

Click `Help` in the main menu of Visual Studio. Click the sub-menu `Index`.
Go to `Filtered by:` and choose: `.NET Framework SDK`. Then enter into `Look for:` one of the following key words: `Timer class (System.Windows.Forms), Control.OnResize method (System.Windows.Forms), Math.Cos, Math.Sin, Math.PI, static keyword C#, Graphics.Clear` etc. You obtain a selection of key words beginning with those characters. Read the help texts. The help window covers Your code. When you finished reading, get rid of it with the X-button in the upper right window corner.

Draw something into the right lower quadrant of the window. Drag the right lower edge of the window and observe the adaptive behaviour of the animation when its window changes. Try to understand the sense of the variables `myWidth` and `myHeight`.
Retard the velocity of the animation by setting the `Timer.Interval` to `10, 100, 1000`.
Speedup the animation by increasing the angular steps from 1 to 2, 4, 6 degrees.
Extend the interrupt conditions by lowering (even negative values !) `xmin` and `xmax` and increasing `ymin` and `ymax`.
Invent and try out new variants of the program in form of new projects `anim2, anim3`.

Mini program for 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 );
  }
}
```