Course 2DCis: 2D-Computer Graphics with C#
Chapter C2: The Draw Project

Projekt draw1 mit leerem Fenster

Anleitung für **Visual C# 2008 Express**:
`0)` Main Menu nach dem Start von VS 2008: `Tools → Options →`
check lower left checkbox: `Show all Settings → Projects and Solutions →`
`Visual Studio projects location: → C:\temp`

`1)` Main Menu nach dem Start von VS 2008: `File → New Project... →`
`Visual Studio installed templates: Windows Forms Application`
`Name: draw1 → 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 - draw1`-Window: Klicken Sie das Pluszeichen vor `draw1` 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`.

Das Miniprogramm wird jetzt automatisch übersetzt, gelinkt und gestartet. Beachten Sie das `Error List`-Fenster am unteren Rand von Visual Studio.
Unser Programm startet automatisch als eigenes Fenster, das aus drei Teilen besteht:
1. Hauptfenster = MainFrame mit blauer Titelzeile
2. drei funktionierende Buttons für `Minimize, Maximize, Close`
3. ein schmaler Rahmen, dessen 4 Seiten beweglich sind und dessen 4 Ecken beweglich sind.
Verkleinern/Vergrößern Sie den Rahmen. Diese gesamte Funktionalität ist in der vorprogrammierten Klasse `System.Windows.Forms.Form` enthalten und steht jetzt zu Verfügung als Basis für Ihr `draw1`-Programm.

Wichtig: Immer zuerst alle Instanzen von `draw1` beenden, bevor neuer Code eingegeben und übersetzt wird ! Im Zweifel Task Manager mit `Ctrl+Alt+Del` starten und kontrollieren, ob noch ein `draw1.exe`-Prozess läuft und diesen töten.

Minimales Malprogramm

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;

public class Form1 : Form
{ [STAThread] static void Main() { Application.Run( new Form1() ); }
  Graphics g;
  Point p0, p1;
  Brush redbrush   = new SolidBrush( Color.Red );
  Brush graybrush  = SystemBrushes.Control;
  Brush blackbrush = SystemBrushes.ControlText;
  Pen   blackpen   = SystemPens.ControlText;
  Pen   redpen     = new Pen( Color.Red );
  Pen   greenpen   = new Pen( Color.Green );

  public Form1()
  { Text = "Draw1: A Scribble Program";
    Width = 800;
    Height = 600;
    g = this.CreateGraphics();
  }
  protected override void OnMouseDown( MouseEventArgs e )
  { p0 = e.Location;
    Invalidate();
  }
  protected override void OnMouseMove( MouseEventArgs e )
  { if ( e.Button == MouseButtons.None ) return;
    p1 = e.Location;
    g.DrawLine( blackpen, p0, p1 );
    p0 = p1;
  }
  protected override void OnMouseUp( MouseEventArgs e )
  {
  }
  protected override void OnPaint( PaintEventArgs e )
  { g.DrawString( "Press the left mouse button and move!", Font, redbrush, 0, 0 );
  }
}
```

Klicken Sie `Debug` in der Menü-Zeile oberhalb des Hauptfensters
Es öffnet sich ein Untermenü. Klicken Sie auf `Start Without Debugging Ctrl F5`.
Das Programm wird wieder automatisch übersetzt, gelinkt und gestartet. Malen Sie alle möglichen Figuren, ändern Sie den `Pen` in `...OnMouseMove(...)` und den `Brush` in `...OnPaint(...)`.

Wichtiger Hinweis 1: Wenn Sie sich beim Programmieren vertippt haben, dann übersetzt der Compiler nichts, sondern meldet sich mit einer Message Box: `There were build errors. Continue ?`.
Sie quittieren mit `No`. Am unteren Rand von Visual Studio erscheint ein Fenster mit Warnungen und Fehlermeldungen. Scrollen Sie innerhalb dieser Fehlerliste nach oben zur ersten Fehlermeldung (Ignorieren Sie eventuelle Warnungen). Doppelklicken Sie die erste Fehlermeldung.
Der Cursor springt nun automatisch in Ihren Code auf die Zeile, wo der Fehler entdeckt wurde.
Suchen Sie dort den Fehler (manchmal liegt er gar nicht in dieser Zeile, sondern in vorherigen Zeilen, wenn dort ein Semikolon oder eine Klammer fehlt) und beseitigen Sie ihn.
Ignorieren Sie alle weiteren Fehler (das sind meistens Folgefehler des ersten Fehlers) und übersetzen Sie neu.
Diesen Vorgang wiederholen Sie so oft, bis keine Fehlermeldung mehr erscheint und Ihr übersetztes und gelinktes Programm automatisch startet.

Wichtiger Hinweis 2: Schalten Sie die Formatier- und Einrückautomatik des Code-Editors aus auf folgendem Weg:
1. Hauptmenu von Microsoft Visual Studio: Klick auf Menupunkt "`Tools`".
2. Es erscheint ein DropDown-Menu. Klick auf "`Options...`".
3. Es erscheint eine Dialog Box. Klick auf "`Text Editor`", dann auf "`C#`".
4. Es erscheint ein Unterbaum "`General, Tabs, Formatting`". Klick auf "`Tabs`".
5. Stellen Sie `Indenting` auf "`None`", `Tab size` und `Indent size` auf `1` und schalten Sie die Option "`Insert spaces`" ein.
6. Im Unterbaum "`General, Tabs, Formatting`". Klick auf "`Formatting`".
7. Schalten Sie "`Leave open braces on same line as construct`" ein, aber alle anderen Kästchen aus.
8. Verlassen Sie den Dialog mit Button "`OK`".

Anzeige der Koordinaten

Version2: Beenden Sie Ihr Programm `draw1`.
Schreiben Sie eine weitere Deklaration in den Kopf von `public class Form1 : Form` unterhalb der Zeile `Pen greenpen = new Pen(Color.Green);`:

```
String myline;
```

und drei weitere Zeilen in die Funktion `protected override void OnMouseMove(MouseEventArgs e)` direkt unterhalb von Zeile `p0 = p1;`, aber noch vor der geschweiften Klammer, die die Methode `OnMouseMove` abschließt:

```
    g.DrawString( myline, Font, graybrush , 0, Font.Height );
    myline = String.Format( "{0}, {1}", p1.X, p1.Y );
    g.DrawString( myline, Font, blackbrush, 0, Font.Height );
```

Klicken Sie `Debug` in der Menü-Zeile oberhalb des Hauptfenster und dann `Start Without Debugging Ctrl F5`.

Anzeige der Vertices

Vertices (deutsche Aussprache: Wertizehs, englisch: wörtiziis) ist der Plural des lateinischen Wortes Vertex = Eckpunkt = Computergraphik-Fachausdruck für Polygonecke.
Version3: Beenden Sie Ihr Programm `draw1`.
Schreiben Sie eine weitere Zeile in die Funktion `protected override void OnMouseMove(MouseEventArgs e)` unterhalb die schon vorhandenen Befehle:

```
    g.DrawRectangle( blackpen, p1.X-3, p1.Y-3, 7, 7 );
```

Klicken Sie `Debug` und dann `Start Without Debugging Ctrl F5`.
Zeichnen Sie schnell und langsam und beobachten Sie, wie die Dichte der Vertices von der Malgeschwindigkeit abhängt.
Langsame Computer erzeugen weniger Vertices als schnelle, weil das Betriebssystem die Windows-Message `WM_MouseMove` seltener erzeugt und an `class Form1` schickt, welche darum seltener ihre Methode `protected override void OnMouseMove(MouseEventArgs e)` aktiviert.
Wichtige Aussage: Das Programm kann keine konstante Anzahl von Vertices erzeugen, sondern diese Anzahl hängt von der Malgeschwindigkeit und der Hardware ab.
Bei langsamen Mausbewegungen und schnellen Rechnern werden offensichtlich viel zu viele Vertices erzeugt.
Es gibt witzige Effekte, wenn Sie die Quadrate größer machen z.B. mit `p1.X-19, p1.Y-19, 40, 40`.

Minimalabstand der Vertices

Version4: Beenden Sie Ihr Programm `draw1`.
Schreiben Sie drei weitere Zeilen in die Funktion `protected override void OnMouseMove(MouseEventArgs e)` direkt unterhalb des Statements: `p1 = e.Location;`

```
    Int32 dx = p1.X - p0.X;
    Int32 dy = p1.Y - p0.Y;
    if ( dx*dx + dy*dy < 100 ) return;
```

Klicken Sie `Debug` und dann `Start Without Debugging Ctrl F5`.
Wir benutzen den Satz des Pythagoras für rechtwinklige Dreiecke: Summe der Quadrate der Katheten = Quadrat der Hypotenuse. In unserem Fall ignorieren wir den neuen Vertex, wenn die Hypotenuse nicht mindestens 10 Pixel lang ist. Zeichnen Sie schnell und langsam und beobachten Sie die Dichte der Vertices.
Ersetzen Sie probeweise die Zahl `100` durch `0, 4, 16, 64, 400, 900, 1600` etc.
siehe: **../../Lectures/L02_2DVector/2DVector_d.htm**

Vertex-Array

Version5:
Das gezeichnete Polygon geht verloren, wenn `Form1` verkleinert wurde oder wenn `Form1` von einem anderen Fenster zeitweilig bedeckt war und es ist ein offenes Polygon = Polyline.
Diese Mängel werden nun behoben dadurch, dass 1) die Vertices laufend in einem Array fester Länge gespeichert und bei jeder Fenstererneuerung automatisch neu gezeichnet werden und 2) dass der letzte mit dem ersten Vertex automatisch verbunden wird.
Beenden Sie Ihr Programm `draw1`.
Schreiben Sie folgende Zeilen in den Kopf von `public class Form1 : Form`
unterhalb der Zeile `Graphics g;`:

```
  Int16 i, n;
  const Int16 nMax = 100;
  Point[] polygon = new Point[nMax];
```

Ändern Sie die Funktion `protected override void OnMouseDown(MouseEventArgs e)`,
dass sie so aussieht:

```
  protected override void OnMouseDown( MouseEventArgs e )
  { polygon[0] = p0 = e.Location;
    n = 1;
    Invalidate();
  }
```

Ändern Sie die Funktion `protected override void OnMouseMove(MouseEventArgs e)`,
dass sie so aussieht:

```
  protected override void OnMouseMove( MouseEventArgs e )
  { if ( e.Button == MouseButtons.None ) return;
    p1 = e.Location;
    Int32 dx = p1.X - p0.X;
    Int32 dy = p1.Y - p0.Y;
    if ( dx*dx + dy*dy < 100 ) return;
    if ( n >= nMax-1 ) return;
    g.DrawLine( blackpen, p0, p1 );
    polygon[n++] = p0 = p1;
    g.DrawString( myline, Font, graybrush , 0, Font.Height );
    myline = String.Format( "{0}, {1}", p1.X, p1.Y );
    g.DrawString( myline, Font, blackbrush, 0, Font.Height );
    g.DrawRectangle( blackpen, p1.X-3, p1.Y-3, 7, 7 );
  }
```

Ändern Sie die Funktion `protected override void OnMouseUp(MouseEventArgs e)`,
dass sie so aussieht:

```
protected override void OnMouseUp( MouseEventArgs e )
{ if ( n < 2 ) return;
  polygon[n++] = polygon[0];
  Invalidate();
}
```

Ändern Sie die Funktion `protected override void OnPaint( PaintEventArgs e )`,
dass sie so aussieht:

```
protected override void OnPaint( PaintEventArgs e )
{ g.DrawString( "Press the left mouse button and move!", Font, redbrush, 0, 0 );
  for ( int i=0; i < n-1; i++ ) g.DrawLine( blackpen, polygon[i], polygon[i+1] );
}
```

Klicken Sie `Debug` und dann `Start Without Debugging Ctrl F5`.
Erproben Sie das Programm durch 1.) ziehen am Fensterrand: `Form1` extrem verkleinern und zurückvergrößern und 2.) durch überdecken und abdecken von `Form1` mit irgend einer andern Windows-Applikation.
Verkürzen und verlängern Sie den Array durch Änderung von `nMax` auf `10` und `200` und
erproben Sie die Auswirkungen.
Hinweis: Die Verwendung eines Arrays fester Länge `Point[] polygon = new Point[nMax];`
ist nicht optimal, weil die Länge wilkürlich vorbestimmt ist; fast immer ist sie zu kurz oder zu lang.
Besser, eleganter aber auch langsamer ist ein dynamischer Array
`ArrayList polygon = new ArrayList();` wie im 3. Miniprogramm zum Training am Ende dieser Seite.
Mit dessen Methoden `Add`, `Insert` und `Clear` kann man den Array beliebig verlängern und kürzen.

Umfang, Fläche, Schwerpunkt, umschreibendes Rechteck

Version6: Beenden Sie Ihr Programm `draw1`.
see: **../../Lectures/L02_2DVector/2DVector_d.htm**
Schreiben Sie weitere Deklarationen in den Kopf von `public class Form1 : Form`
unterhalb der Zeile `Point p1 = new Point();`:

```
Point mid_of_p, mid_of_r;
Rectangle minmax;
Double perimeter, area;
```

Ändern Sie die Funktion `protected override void OnMouseUp(MouseEventArgs e)`,
dass sie so aussieht:

```
protected override void OnMouseUp( MouseEventArgs e )
{ if ( n < 2 ) return;
  p0 = polygon[n++] = polygon[0];
  perimeter = area = 0;
  mid_of_p.X = mid_of_p.Y = 0;
  Int32 xmin, xmax, ymin, ymax;
  xmin = xmax = p0.X;
  ymin = ymax = p0.Y;
  for ( i=1; i < n; i++ )
  { p1 = polygon[i];
    Double dx = p1.X - p0.X;
    Double dy = p1.Y - p0.Y;
    Double my = (p0.Y + p1.Y) / 2.0;
    perimeter  += Math.Sqrt( dx*dx + dy*dy );
    area       += dx * my;
    mid_of_p.X += p1.X;
    mid_of_p.Y += p1.Y;
    if ( p1.X < xmin ) xmin = p1.X;
    if ( p1.X > xmax ) xmax = p1.X;
    if ( p1.Y < ymin ) ymin = p1.Y;
    if ( p1.Y > ymax ) ymax = p1.Y;
    p0 = p1;
  }
```

```
    mid_of_r.X = ( xmax + xmin ) / 2;
    mid_of_r.Y = ( ymax + ymin ) / 2;
    mid_of_p.X /= n-1;
    mid_of_p.Y /= n-1;
    minmax.X = xmin-1; minmax.Width  = xmax - xmin + 2;
    minmax.Y = ymin-1; minmax.Height = ymax - ymin + 2;
    Invalidate();
  }
```

Ändern Sie die Funktion `protected override void OnPaint( PaintEventArgs e )`,
dass sie so aussieht:

```
  protected override void OnPaint( PaintEventArgs e )
  { g.DrawString( "Press the left mouse button and move!", Font, redbrush, 0, 0 );
    if ( n < 2 ) return;
    myline = String.Format( "Perimeter= {0}, Area= {1}", (Int32)perimeter, (Int32)area );
    g.DrawString( myline, Font, blackbrush, 0, 2*Font.Height );
    for ( i=0; i < n-1; i++ ) g.DrawLine( blackpen, polygon[i], polygon[i+1] );
    for ( i=0; i < n-3; i+=3 )
      g.DrawBezier( redpen, polygon[i], polygon[i+1], polygon[i+2], polygon[i+3] );
    g.DrawRectangle( greenpen, minmax );
    g.DrawLine( greenpen, mid_of_r.X - 4, mid_of_r.Y    , mid_of_r.X + 4, mid_of_r.Y     );
    g.DrawLine( greenpen, mid_of_r.X    , mid_of_r.Y - 4, mid_of_r.X    , mid_of_r.Y + 4 );
    g.FillEllipse( blackbrush, mid_of_p.X-5, mid_of_p.Y-5, 11, 11 );
  }
```

Klicken Sie `Debug` und dann `Start Without Debugging Ctrl F5`.
Erproben Sie das Programm und erarbeiten Sie sich den Zusammenhang zwischen
dem Code in `OnMouseUp / OnPaint` und dem was Sie im Ergebnis sehen.
Beachten Sie, wie das Vorzeichen von `Area` von der Umlaufrichtung abhängt.

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`. Dann geben Sie im Feld `Look for:` folgende Schlüsselwörter ein: `Rectangle, Point, Array, MouseEventArgs, PaintEventArgs, Math` etc. Sie bekommen dann eine Auswahl von Stichwörtern, mit dem gleichen Anfang.
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.
Beenden Sie Visual Studio, starten sie den Explorer, löschen Sie die gesamte Directory `C:\temp\draw1`
Starten Sie Visual Studio wieder und erzeugen dasselbe Programm so oft, bis Sie das Programm ohne Anleitung und schnell von Null an erstellen, verändern und bedienen können.
Benutzen Sie Drag&Drop beim Schreiben von Code = verschieben und kopieren mit Maus mit/ohne Strg-Taste.
Erfinden und erproben Sie neue Varianten des Programms (in Form von neuen Projekten `draw2, draw3` usw. nach obigem Muster.

Miniprogramme zum Training

1. MiniDraw ohne Speichern und Neuzeichnen nach verkleinern + vergrößern

```
using System;
using System.Drawing;
using System.Windows.Forms;
public class Form1 : Form
{ [STAThread] static void Main() { Application.Run( new Form1() ); }
  Graphics g;
  Point p0, p1;
  Pen redpen = new Pen( Color.Red, 5 );
  protected override void OnMouseDown( MouseEventArgs e )
  { g = this.CreateGraphics();
    p0 = e.Location;
  }
  protected override void OnMouseMove( MouseEventArgs e )
  { if ( e.Button == MouseButtons.None ) return;
    p1 = e.Location;
    g.DrawLine( redpen                , p0, p1 );
    g.DrawLine( SystemPens.ControlText, p0, p1 );
    p0 = p1;
  }
}
```

2. MiniDrawArray mit festem Array und mit `OnPaint`-Event Handler zum Neuzeichnen nach Zerstörung

```
using System;
using System.Drawing;
using System.Windows.Forms;
public class Form1 : Form
{ [STAThread] static void Main() { Application.Run( new Form1() ); }
  Graphics g;
  const Int16 N = 100;
  Point[] p = new Point[N];
  Int16 n;
  protected override void OnMouseDown( MouseEventArgs e )
  { g = this.CreateGraphics();
    p[0] = e.Location;
    n = 1;
    Invalidate();
  }
  protected override void OnMouseMove( MouseEventArgs e )
  { if ( e.Button == MouseButtons.None ) return;
    if ( n >= N ) return;
    p[n] = e.Location;
    g.DrawLine( SystemPens.ControlText, p[n-1], p[n] );
    n++;
  }
  protected override void OnPaint( PaintEventArgs e )
  { for ( Int16 i = 1; i < n; i++ )
      g.DrawLine( SystemPens.ControlText, p[i-1], p[i] );
  }
}
```

3. MiniDrawDynArray mit dynamischem Array

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Collections;
public class Form1 : Form
{ [STAThread] static void Main() { Application.Run( new Form1() ); }
  Graphics g;
  Point p0;
  ArrayList p = new ArrayList();
  protected override void OnMouseDown( MouseEventArgs e )
  { g = this.CreateGraphics();
    p0 = e.Location;
    p.Clear(); p.Add( p0 );
    Invalidate();
  }
  protected override void OnMouseMove( MouseEventArgs e )
  { if ( e.Button == MouseButtons.None ) return;
    p0 = e.Location;
    g.DrawLine( SystemPens.ControlText, (Point)p[p.Count-1], p0 );
    p.Add( p0 );
    g.FillRectangle( SystemBrushes.Control, 0, 0, 200, 15 );
    String s = "length = " + p.Count.ToString();
    g.DrawString( s, new Font( "Arial", 12 ), SystemBrushes.ControlText, 0, 0 );
  }
  protected override void OnPaint( PaintEventArgs e )
  { for ( Int16 i = 1; i < p.Count; i++ )
      g.DrawLine( SystemPens.ControlText, (Point)p[i-1], (Point)p[i] );
  }
}
```