Course 2DC7, C2: Draw, Bauanleitung mit C++/MFC7.0

Projekt draw1 mit leerem Fenster

`Microsoft Visual Studio.NET starten`
`File – New – Project – Project Types: Visual C++ Projects, Templates: MFC Application`
Name: `draw1`
Location: `C:\temp`
Button `OK` unten Mitte klicken
Es meldet sich der `MFC Application Wizard - draw1` mit der Seite `Overview`, die uninteressant ist.
Links unter `Overview` auf `Application Type` klicken.
Schritt1: **`Single document`** einschalten
Schritt 2: **`Document/View architecture support`** Checkbox ausschalten
Schritt 3: `Finish` VS.NET legt nun unter `C:/temp` ein Directory `draw1` an und schreibt dorthin jede Menge Files, die insgesamt das Projekt `draw1` bilden.
Klicken Sie `Debug` in der Menü-Zeile oberhalb des Hauptfensters
Es öffnet sich ein Untermenü. Klicken Sie auf `Start Without Debugging Ctrl F5`
Es erscheint eine Message Box: `These project configuration(s) are out of date: draw1 - Debug Win 32. Would you like to build them ?`
Sie antworten mit `Yes`

Wichtig: Immer zuerst dieses leere Projekt ausführen `draw1`.
Nur so sind Sie sicher, dass es Sinn macht, eigenen Code einzutippen. Wenn Sie diesen Test nicht als erstes machen, riskieren Sie Projektcrash wenn Compiler/Linker-Optionen oder Pfade falsch eingestellt sind und Sie verlieren in diesem Fall leicht alle Ihre Eingaben.
Falls alles ok, dann `draw1.exe` erst beenden, bevor der erste Code eingegeben wird !

Minimales Malprogramm

Klicken Sie im Hauptmenu von VS.NET auf den Menupunkt `View` und dann auf den Unterpunkt `Class View Ctrl+Shift+C`.
Sie sehen im rechten Bereich des VS.NET-Hauptfensters das Unterfenster `Class View - draw1`.
An dessen unteren Rand die Registerkarte Klassen klicken.
Darin sehen Sie die Zeile `+ draw1` , klicken Sie das Pluszeichen.
Sie sehen u.a. `+ CChildView`, doppelklicken Sie diesen Klassennamen.
Sie editieren jetzt das File `ChildView.h`, der Cursor steht auf `class CChildView : public CWnd`.
Schreiben Sie dort direkt unter die sich öffnende geschweifte Klammer noch vor `//Construction` die Deklaration:
`private: CPoint old_vertex;`
Doppelklicken sie im `Class View - draw1` - Fenster das Pluszeichen vor CChildView.
Sie sehen nun u.a. die Member-Variable `old_vertex` im Klassenbaum von `CChildView`.
Klicken Sie dort mit der rechten Maustaste auf `CChildView` (nicht verwechseln mit dem gleichnamigen Konstruktor `CChildView(void)`).
Es öffnet sich ein Kontextmenü. Klicken Sie ganz unten im Kontextmenü auf `Properties`.
Es öffnet sich unter dem `Class View - draw1` - Fenster ein `Properties` - Fenster mit der Überschrift `CChildview VCCodesClass`.

Im Toolbar dieses Fensters klicken Sie rechts neben dem gelben Blitz auf das `Messages` - Symbol, das aussieht wie ein weißer Topf mit blauem Deckel.
Sie sehen nun die Liste der Windows-Messages von `CChildView`.
Klicken Sie auf `WM_LBUTTONDOWN` und dann auf das schwarze Abwärtsdreieck rechts in dieser Zeile und dann auf `Add OnLButtonDown`.
Klicken Sie auf `WM_MOUSEMOVE` und dann auf das schwarze Abwärtsdreieck rechts in dieser Zeile und dann auf `Add OnMouseMove`.
Klicken Sie auf `WM_LBUTTONUP` und dann auf das schwarze Abwärtsdreieck rechts in dieser Zeile und dann auf `Add OnLButtonUp`.
Sie sehen nun u.a. die Methoden `OnLButtonDown(..), OnMouseMove(..)` und `OnLButtonUp(..)` im Klassenbaum von `CChildView`, und Sie sehen die drei fast leeren Funktionsrümpfe in `CChildView.cpp`, die Visual Studio automatisch generiert hat.

Doppelklicken Sie im Klassenbaum auf `OnLButtonDown(...)`. (oder gehen Sie direkt in den Code in die Funktion `void CChildView::OnLButtonDown(...)`).
Der Cursor steht nun auf `void CChildView::OnLButtonDown(...)`. Anstelle von `// TODO` etc schreiben Sie die beiden Befehle:

```
old_vertex = point; Invalidate();
```

Doppelklicken Sie im Klassenbaum auf `OnMouseMove(...)`. (oder gehen Sie direkt in den Code in die Funktion `void CChildView::OnMouseMove(...)`).
Der Cursor steht nun auf `void CChildView::OnMouseMove(...)`. Anstelle von `// TODO` etc schreiben Sie die 5 Befehle:

```
if ( !nFlags ) return;
CClientDC dc(this);
dc.MoveTo( old_vertex ); dc.LineTo( point );
old_vertex = point;
```

Doppelklicken Sie auf `OnPaint(void)`.
Unterhalb der Zeile `CPaintDC dc(this);` schreiben Sie die neue Zeile:

```
dc.TextOut( 0, 0, "Press the left mouse button and move!" );
```

Klicken Sie `Debug` in der Menü-Zeile oberhalb des Hauptfensters
Es öffnet sich ein Untermenü. Klicken Sie auf `Start Without Debugging Ctrl F5`
Es erscheint eine Message Box: `These project configuration(s) are out of date: draw1 - Debug Win 32. Would you like to build them ?`
Sie antworten mit `Yes`
Die Malprogramm 1. Version ist damit fertig, ausführen, erproben, beenden.

Anzeige der Koordianten

Version 2: Beenden Sie `draw1.`
Ergänzen Sie in `OnMouseMove(...)` unter die Zeile `old_vertex = point;` folgende 3 Befehle:

```
CString blabla;
blabla.Format( "x=%d, y=%d      ", point.x, point.y );
dc.TextOut( 0, 20, blabla );
```

Die 2. Version ist damit fertig, ausführen, erproben, beenden.

Anzeigen der Vertices

Vertices (deutsche Aussprache: Wertizehs, englisch: wörtiziis) ist der Plural des lateinischen Wortes Vertex = Eckpunkt = Computergraphik-Fachausdruck für Polygonecke.
Version 3: Beenden Sie `draw1`.
Ergänzen Sie in `OnMouseMove(...)` unter `dc.TextOut( 0, 20, blabla );` noch eine Zeile:

```
dc.Rectangle( point.x-3, point.y-3, point.x+3, point.y+3 );
```

Die 3. Version ist damit fertig, ausführen.
Zeichnen Sie schnell und langsam und beobachten Sie, wie die Dichte der Vertices von der Malgeschwindigkeit abhängt.
Langsame Computern und Graphikkarten erzeugen ausserdem weniger Vertices als schnelle, weil das Betriebssystem die Windows-Message `WM_MouseMove` seltener erzeugt und `OnMouseMove(...)` 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.

Minimalabstand der Vertices

Version 4: Beenden Sie `draw1`.
Ergänzen Sie in `OnMouseMove(...)` unter `if ( !nFlags ) return;` folgende 3 Zeilen:

```
int dx = point.x - old_vertex.x;
int dy = point.y - old_vertex.y;
if ( dx*dx + dy*dy < 100 ) return;
```

Die 4. Version ist damit fertig, ausführen, schnelle und langsame Bewegungen erproben, beenden.

Vertex-Array, Umfang, Fläche etc.

Version 5: Beenden Sie `draw1`.
Doppelklick auf den Klassennamen `CChildView` im linken Arbeitsbereichfenster.
Sie editieren jetzt das File `CChildView.h`.
Vor die Klasse `CChildView` ganz an den Anfang des Files `CChildView.h` schreiben Sie die 2 folgenden Präprozessoranweisungen:

```
#define nMax 100
#include < math.h >
```

In die Klasse `CChildView : public CWnd` unter die Zeile: `private: CPoint old_vertex;`

```
CPoint polygon[nMax];
CPoint center_of_gravity;
CRect  minmax;
int    n;
double perimeter, area;
```

Die Membervariablen `old_vertex, area, minmax, n, polygon, center_of_gravity, perimeter` müssen jetzt im Klassenbaum zu sehen sein.

Neue Befehle in `OnLButtonDown(...)` vor `Invalidate()`:

```
polygon[0] = old_vertex = point;
n = 1;
```

Hinzufügen eines neuen Befehls, erhalten der alten `MoveTo`- und `LineTo`-Befehle und erweitern des letzten Befehls in `OnMouseMove(...)` unterhalb von Zeile
`if ( dx*dx + dy*dy < 100 ) return;`

```
if ( n > nMax - 2 ) return;
dc.MoveTo( old_vertex ); dc.LineTo( point );
old_vertex = polygon[n++] = point;
```

Neue Befehle in `OnLButtonUp(...)`:

```
if ( n < 2 ) return;
polygon[n++] = polygon[0];
perimeter = area = 0.;
center_of_gravity.x = center_of_gravity.y = 0;
minmax.left = minmax.right  = polygon[0].x;
minmax.top  = minmax.bottom = polygon[0].y;
for ( int i = 1; i < n; i++ )
{ int x = polygon[i].x;
   int y = polygon[i].y;
   double dx = double( x - polygon[i-1].x );
   double dy = double( y - polygon[i-1].y );
   double my = double( y + polygon[i-1].y ) * 0.5;
   perimeter        += _hypot( dx, dy );
   area             += dx * my;
   center_of_gravity.x += x;
   center_of_gravity.y += y;
   if ( x < minmax.left   ) minmax.left   = x;
   if ( x > minmax.right  ) minmax.right  = x;
   if ( y < minmax.top    ) minmax.top    = y;
   if ( y > minmax.bottom ) minmax.bottom = y;
}
center_of_gravity.x /= n-1;
center_of_gravity.y /= n-1;
Invalidate();
```

Neue Befehle in `OnPaint()` nach `dc.TextOut(0,0, "Press the left mouse button and move!");`

```
if ( n < 2 ) return;
CString blabla;
blabla.Format( "Perimeter=%d, Area=%d", int(perimeter), int(area) );
dc.TextOut( 0, 140, blabla );
dc.Rectangle( minmax );
dc.Polygon ( polygon, n );
dc.Rectangle( center_of_gravity.x-3, center_of_gravity.y-3, center_of_gravity.x+3,
center_of_gravity.y+3 );
```

Die 5. Version ist damit fertig, ausführen, erproben, beenden.

Bézier-Kurven

Version6: Beenden Sie `draw1`.
Erhöhen Sie in `OnMouseMove(...)` das Quadrat des Minimalabstandes von 100 auf 400, 900, 1600.
Ersetzen Sie in `OnPaint()` den Befehl `dc.Polygon ( polygon, n );` durch folgende Befehle:

```
dc.Polyline( polygon, n );
switch ( n%3 )
{   case 0: dc.PolyBezier( polygon, n-2 ); break;
    case 1: dc.PolyBezier( polygon, n   ); break;
    case 2: dc.PolyBezier( polygon, n-1 ); break;
}
```

Die 6. Version ist damit fertig, ausführen, erproben, beenden.

Weitere Aufgaben

Klicken Sie auf das Fragezeichen in der Menüleiste von Visual C++. Klicken Sie auf das Untermenü Index. Suchen Sie die Schlüsselwörter der Ihnen unbekannten Sprachelemente. Lesen Sie die Hilfetexte.
Beenden Sie VisualC++, starten sie den Explorer, löschen Sie die gesamte Directory `C:\temp\draw1`
Starten Sie VisualC++ wieder und erzeugen dasselbe Programm so oft, bis Sie das Programm ohne Anleitung und schnell von Null an erstellen, verändern und bedienen können.

Erfinden und erproben Sie neue Varianten des Programms.