Course 2DC7, C4: File, Bauanleitung mit C++/MFC7.0

Projekt file1 mit leerem Fenster

`Microsoft Visual Studio.NET starten`
`File – New – Project – Project Types: Visual C++ Projects, Templates: MFC Application`
Name: `file1`
Location: `C:\temp`
Button `OK` unten Mitte klicken
Es meldet sich der `MFC Application Wizard - file1` 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 **einschalten**
Schritt 3: `Finish`
Schritt 4: Schritt4 setzt den Microsoft Internet Explorer ab 5.0 als Browser voraus. Wenn das nicht der Fall ist, sehen Sie das Ergebnis von Schritt4 nicht. Öffnen Sie das Text-File `vml.txt` mit Hilfe des Browsers. Wenn Ihr Browser ein leeres Fenster zeigt, lassen Sie sich `vml.txt` vom Browser als Quelltext anzeigen (Menu `Ansicht, Quelltext anzeigen`). Speichern Sie `vml.txt` unter dem neuen Namen `vml.html` nach `C:\temp\file1` auf Ihre Harddisk.

Präprozessorbefehle und Deklarationen in CFile1Doc.h

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 - file1`.
An dessen unteren Rand die Registerkarte Klassen klicken.
Darin sehen Sie die Zeile `+ file1` , klicken Sie das Pluszeichen.
Sie sehen u.a. `+ CFile1Doc`, doppelklicken Sie diesen Klassennamen.
Sie editieren jetzt das File `file1Doc.h`, Schreiben Sie dort vor die Klasse `class CFile1Doc : public CDocument` am besten unter die Zeile `#pragma once` die Präprozessoranweisungen:

```
#include < vector > //für die dynamischen Arrays der STL
```

Schreiben Sie in die Klasse direkt unter die sich öffnende geschweifte Klammer die Deklarationen:

```
public:
   std::vector< CPoint > p;  // p ist ein dynamischer Array der Länge Null
```

Sie sehen nun diese Member-Variable `p` im Klassenbaum von `CChildView` im rechten `ClassView - file1` - Fenster, wenn Sie das Pluszeichen vor `Cfile1Doc` klicken.

Behandlungsroutinen für Windows-Nachrichten

Klicken Sie im Klassenbaum von `Cfile1View` mit der rechten Maustaste auf `CChildView` (nicht verwechseln mit dem gleichnamigen Konstruktor `Cfile1View(void)`).
Es öffnet sich ein Kontextmenü. Klicken Sie ganz unten im Kontextmenü auf `Properties`.
Es öffnet sich unter dem `Class View - file1` - Fenster ein `Properties` - Fenster mit der Überschrift `Cfile1View 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 `Cfile1View`.
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(..), OnLButtonUp(..)` im Klassenbaum von `Cfile1View`, und Sie sehen die fünf fast leeren Funktionsrümpfe in `Cfile1View.cpp`, die Visual Studio automatisch generiert hat.
Die ziemlich leeren Funktionshülsen stehen in `Cfile1View.cpp` unterhalb der Kommentarzeile `// Cfile1View message handlers`. Verschieben Sie die Funktionshülsen samt Inhalt und geschweiften Klammern per Drag und Drop so, dass diese eine logisch vernünftige Reihenfolge bilden (das ist wichtig für den Überblick).

`void Cfile1View::OnDraw(CDC* /*pDC*/)` Diese vorgefertigte Funktion von weiter oben hierher verschieben !
`void CChildView::OnLButtonDown(...)`
`void CChildView::OnMouseMove(...)`
`void CChildView::OnLButtonUp(...)`

Einfügen von Code in CFile1View

Nun füllen Sie die Funktionen mit Code, bis sie so aussehen:

```
void Cfile1View::OnDraw(CDC* pDC) // Pointer auf CDC übergeben
{  Cfile1Doc* pDoc = GetDocument();  // Pointer auf CFile1Doc
   ASSERT_VALID(pDoc);               // Schluss, wenn pDoc == NULL
   pDC->TextOut( 10, 10, "draw->store->close->open" );
   int n = pDoc->p.size(); // wie lange ist p ?
   if ( n < 2 ) return;    // zu kurz
   pDC->Polyline( &(pDoc->p.front()), n);  //&(pDoc->p.front()) = Adresse
des Anfangs von Array p
}
void Cfile1View::OnLButtonDown(UINT nFlags, CPoint point)
{  Cfile1Doc* pDoc = GetDocument();
   pDoc->p.clear();       // alles wegwerfen, Länge auf Null
   pDoc->p.push_back( point );  //ersten Punkt merken
   Invalidate();          // Fenster neu
}
void Cfile1View::OnMouseMove(UINT nFlags, CPoint point)
{  if ( !nFlags ) return;
   Cfile1Doc* pDoc = GetDocument();
   pDoc->p.push_back( point );
   Invalidate( false );  // Zeichnen, aber ohne löschen
}
void Cfile1View::OnLButtonUp(UINT nFlags, CPoint point)
{  Invalidate();         // Fenster neu
}
```

file1 ist damit fertig, ausführen, erproben, beenden.

Einfügen von Code in Cfile1Doc

In 8 Zeilen kan man das gesamte speichern auf /lesen von der Harddisk formulieren:
Öffnen Sie den Klassenbaum von Cfile1Doc.
Doppelklicken Sie auf Serialize(...). Der Cursor steht auf dem Funktionsrumpf void Cfile1Doc::Serialize(CArchive& ar). Füllen Sie die Funktion mit Code bis sie so aussieht:

```
void Cfile1Doc::Serialize(CArchive& ar)
{  if (ar.IsStoring())      // auf HD schreiben
   {  int n = p.size();     // wieviel Ecken hat das Polygon ?
      if ( n < 2 ) return;  // zu kurz
      ar << n;              // n kommt an den Anfang des Files
      ar.Write( &p.front(), n*sizeof( CPoint ) );  // das sind alle x,y
   }
   else                     // von HD lesen
   {  int n;
      ar >> n;       // zuerst n lesen
      p.resize( n ); // dynamischen Array auf die Länge n bringen
      ar.Read( &p.front(), n*sizeof( CPoint ) ); // alle x,y lesen
      UpdateAllViews( NULL ); //CFile1View::OnDraw(..) aufrufen
   }
}
```

file1 ist vorläufig fertig, ausführen, zeichnen, mit Hilfe des Dateimenus auf Harddisk speichern, von Harddisk laden, beenden, löschen, neu programmieren. Vorschlag: Malen sie zuerst eine 1 mit der Maus. Speichern Sie unter dem Namen "eins". Malen Sie dann eine 2 und speichern diese unter dem Namen "zwei". Dann eine 3 nach gleichem Muster. Jetzt öffnen sie nacheinander eins, zwei und drei.

Polygone im WEB: Vector Markup Language

Das WEB kannte bisher keinerlei Vektorgraphik. Mit den XML-basierten Sprachen SVG und VML ist das neuerdings möglich.
VML ist eine Spezifikation von Microsoft und Macromedia, jedoch kein W3C-Standard.

Der folgende Code schreibt in das in C:\temp\file1 vorhandene VML-File vml.html beliebig viele Polygone.
Um die Ergebnisse zu sehen, müssen Sie nach dem Malen vml.html öffnen mit dem InternetExporer (nur Versionen >= 5.0, kein Netscape).
Wollen Sie die Polygone wieder loswerden, müssen Sie diese zu Fuß mit einem Texteditor aus vml.html löschen.
Dabei dürfen Sie nur löschen was zwischen < body >...< /body > steht.
Die Grundstruktur von vml.html, nämlich < html > < header >...< /header > < body >...< /body > < /html > dürfen Sie beim Löschen nicht zerstören, andernfalls müssen Sie vml.txt neu von dieser Seite laden und unter vml.html nach C:\temp\file1 abspeichern.

Öffnen Sie den Klassenbaum von Cfile1View.
Doppelklicken Sie auf OnLButtonUp(...). Der Cursor steht auf dem Funktionsrumpf void Cfile1View::OnLButtonUp(UINT nFlags, CPoint point). Füllen Sie die Funktion mit Code bis sie wie unten aussieht.
Es gibt aber folgendes Problem:
(1) Der C-Code soll HTML-Code erzeugen.
(2) Dazu muss er HTML-Tags erzeugen.
(3) Der C-Code unten enthält deshalb Strings, die HTML-Tags enthalten.
(4) Ihr Browser, der in diesem Augenblick diesen C-Code anzeigt, würde diese Tags aber sofort und direkt interpretieren und damit den C-Code zerstören.

(5) Um das zu verhindern hat Prof. Miszalok bewusst Fehler in die Strings eingebaut, er hat nämlich nach jedem kleiner-Zeichen < und vor jedem größer-Zeichen > ein Leerzeichen eingebaut.
(6) Diese Leerzeichen müssen Sie herausnehmen, nachdem Sie den C-Code von dieser Seite in den Compiler kopiert haben.
(7) Wenn Sie die Leerzeichen lassen, übersetzt der Compiler alles richtig, aber das Programm erzeugt keine gültigen HTML-Tags, so dass der InternetExplorer `vml.html` falsch interpretieren wird.
(8) In der 13. Zeile `s = ....` müssen Sie also das Leerzeichen zwischen < und v beseitigen.
(9) In der drittletzten Zeile müssen Sie ebenso die Leerzeichen vor und nach `/v:polyline, /body, /html` entfernen.

```
void Cfile1View::OnLButtonUp(UINT nFlags, CPoint point)
{  Invalidate();
   Cfile1Doc* pDoc = GetDocument();
   int n = pDoc->p.size();         //wie lang?
   if ( n < 2 ) return;            //zu kurz
   CFile vml ( _T("vml.html"), CFile::modeReadWrite );  //öffnen
   int length = vml.GetLength();                 //wie lang?
   CString s;
   vml.Read ( s.GetBuffer(length), length ); //File to String
   int i = s.Find( "/body" );          //suche den Tag /body
   if ( i == -1 ) { MessageBeep(-1); return; } //nicht gefunden
   vml.Seek( i-1, CFile::begin );         //rueckspulen+vorspulen
   s = "< v:polyline strokecolor =\"red\" strokeweight = \"4px\" points =
\"";
   vml.Write( s, s.GetLength() );        //Anfang des VML-Polygon-Tags
   for ( i = 0; i < n-1; i++ )           //jeder Vertex in einen String
   {   s.Format( "%dpx, %dpx, ", pDoc->p[i].x, pDoc->p[i].y );
       vml.Write( s, s.GetLength() );    //Strings ans Fileende schreiben
   }
   //letzter Vertex, Ende VML-Polygon-Tag, 2 Leerzeilen, /body+/html-Tags
   s.Format( "%dpx, %dpx\"><  /v:polyline >\r\n\r\n< /body >< /html >",
             pDoc->p[n-1].x, pDoc->p[n-1].y );
   vml.Write( s, s.GetLength() );    //String ans Fileende, fertig
}
```

`file1` ist fertig, ausführen, zeichnen, mit Hilfe des Internet Explorers `C:\temp\file1\vml.html` starten.
Mit Hilfe von `Notepad C:\temp\file1\vml.html` wieder säubern, speichern.
Mit Hilfe von `file1.exe` neue Polygone nach `vml.html` zeichnen.

Hinweise:
(1) Dieser Teil des Codes ist nicht im fertigen Download: `file1.exe` von Prof. Miszalok enthalten.
(2) Wenn Sie diesen Code in Ihr eigenes `file1.exe` integrieren und in eine andere Directory verschieben, müssen Sie `vml.html` mitverschieben oder Sie müssen den richtigen Pfad in Zeile 5 der Funktion `void CFile1View::OnLButtonUp(UINT nFlags, CPoint point)` schreiben.

Weitere Aufgaben

Klicken Sie auf Help in der Menüleiste von VS.NET. Klicken Sie auf das Untermenü Index.
Suchen Sie die Schlüsselwörter der Ihnen unbekannten Sprachelemente. Lesen Sie die Hilfetexte.
Beenden Sie VS.NET, starten sie den Explorer, löschen Sie die gesamte Directory `C:\temp\file1`
Starten Sie VS.NET 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.