Course IPC7, C1: Bitmap, Bauanleitung mit C++/MFC7.0

Projekt bitmap1 mit leerem Fenster

`Microsoft Visual Studio.NET starten`
`File – New – Project – Project Types: Visual C++ Projects, Templates: MFC Application`
Name: `bitmap1`
Location: `C:\temp`
Button `OK` unten Mitte klicken
Es meldet sich der `MFC Application Wizard - bitmap1` 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`

Präprozessorbefehle und Deklarationen in CBitmap1Doc.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 - bitmap1`.
An dessen unteren Rand die Registerkarte Klassen klicken.
Darin sehen Sie die Zeile `+ bitmap1` , klicken Sie das Pluszeichen.
Sie sehen u.a. `+ Cbitmap1Doc`, doppelklicken Sie diesen Klassennamen.
Sie editieren jetzt das File `bitmap1Doc.h`, Schreiben Sie dort vor die Klasse `class Cbitmap1Doc : 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:
  BITMAPFILEHEADER    FH;
  BYTE                IBytes[1200];//bytes for BitmapInfoHeader+Palette
  BITMAPINFOHEADER*   pIH;         //pointer on BitmapInfoHeader
  BITMAPINFO*         pI;          //pointer on BitmapInfo
  std::vector< BYTE > Pixel;       //dyn. array for pixel
  int                 nClicks;  //zum Zählen der Mausklicks
```

Doppelklicken Sie auf den Konstruktor der Klasse: `Cbitmap1Doc(void)`. Sie editieren jetzt das File `bitmap1Doc.cpp`. Sie müssen dort vier Variable initialisieren:

```
Cbitmap1Doc::Cbitmap1Doc(void)
{  memset( IBytes, 0, sizeof(IBytes) );
   pIH = (BITMAPINFOHEADER*) IBytes;
   pI  = (BITMAPINFO*)       IBytes;
   nClicks = 0;
}
```

Einfügen von Code in CBitmap1Doc

Doppelklicken Sie auf `Serialize(CArchive& ar)`. Ersetzen Sie die fast leere Funktionshülse durch folgenden Code:

```
void Cbitmap1Doc::Serialize(CArchive& ar)
{ if (ar.IsStoring()) {}
  else
  { ar.Read( &FH, sizeof(BITMAPFILEHEADER) );
    if ( FH.bfType    != 'MB') { forget_it(); return; }
    if ( FH.bfSize     <= 54 ) { forget_it(); return; }
    if ( FH.bfOffBits <  54 ) { forget_it(); return; }
    int nBytesInfo  = FH.bfOffBits - sizeof(BITMAPFILEHEADER);
    int nBytesPixel = FH.bfSize    - FH.bfOffBits;
    ar.Read( IBytes, nBytesInfo ); //BitmapInfoHeader+Palette
    if ( !(pIH->biBitCount == 8 ||
           pIH->biBitCount ==24) ) { forget_it(); return; }
    Pixel.resize( nBytesPixel );
    ar.Read( &Pixel.front(), nBytesPixel );
  }
}
```

Klicken Sie im `Class View - bimap1` - Fenster mit der rechten Maustaste auf die Klasse `Cbitmap1Doc`. Es öffnet sich ein Kontextmenü. Stellen Sie die Maus auf den 4 Menüpunkt `Add` und klicken Sie dann auf `Add Function...` . Es erscheint der "`Add Member Function Wizard - bitmap1`". Tragen Sie in seine Dialogfelder ein:
`Return type : void`
`Function name: forget_it`
`Access : private`
`.cpp file : bitmap1doc.cpp`
Alle weiteren Felder und Checkboxen bleiben leer. Sie verlassen den Wizard mit dem Button `Finish`.

Der Wizard erzeugt Ihnen einen Funktionshülle, die Sie folgendermaßen füllen:

```
void Cbitmap1Doc::forget_it(void)
{ memset( IBytes, 0, sizeof(IBytes) ); //1200 Bytes löschen
  for ( int i=0; i < 10; i++ ) MessageBeep(-1); //10 Piepser
}
```

Ausführen, lesen muss funktionieren, aber man sieht noch nichts. Falls man eine Datei öffnet, die kein oder ein falsches Bitmap enthält, muss es piepsen.

Behandlungsroutinen für Windows-Nachrichten Cbitmap1View

Klicken Sie im Klassenbaum mit der rechten Maustaste auf `Cbitmap1View`.
Es öffnet sich ein Kontextmenü. Klicken Sie ganz unten im Kontextmenü auf `Properties`.
Es öffnet sich unter dem `Class View - bitmap1` - Fenster ein `Properties` - Fenster mit der Überschrift `Cbitmap1View 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 `Cbitmap1View`.
Klicken Sie auf `WM_LBUTTONDOWN` und dann auf das schwarze Abwärtsdreieck rechts in dieser Zeile und dann auf `Add OnLButtonDown`.

Programmieren Sie zunächst die eben generierte Funktion bis sie so aussieht:

```
void Cbitmap1View::OnLButtonDown(UINT nFlags, CPoint point)
{  Cbitmap1Doc* pDoc = GetDocument();
   pDoc->nClicks++;
   Invalidate();
}
```

Nun programmieren Sie die weiter oben in `bitmap1View.cpp` bereits vorhandene Funktion: `void Cbitmap1View::OnDraw(CDC* pDC)` bis sie so aussieht:

```
void Cbitmap1View::OnDraw(CDC* pDC)
{  Cbitmap1Doc* pDoc = GetDocument();
   if ( !pDoc->pIH->biSize ) { pDC->TextOut(0,0,"Open a *.BMP file !"); return; }
   CString s[11];
   CRect R;
   GetClientRect( R ); //welche Größe hat die ClientArea?
   int FHeader, IHeader, dx, dy, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p,
q, r;
   int xSize = pDoc->pIH->biWidth;
   int ySize = pDoc->pIH->biHeight;

   switch ( pDoc->nClicks % 4 )
   {case 0: // Der Inhalt der beiden Header
      FHeader = sizeof(BITMAPFILEHEADER);
      IHeader = sizeof(BITMAPINFOHEADER);
      a = pDoc->FH.bfSize;
      b = pDoc->FH.bfOffBits;
      c = pDoc->pIH->biSize;
      d = xSize;
      e = ySize;
      f = pDoc->pIH->biPlanes;
      g = pDoc->pIH->biBitCount;
      h = pDoc->pIH->biCompression;
      i = pDoc->pIH->biSizeImage;
      j = pDoc->pIH->biXPelsPerMeter;
      k = pDoc->pIH->biYPelsPerMeter;
      l = pDoc->pIH->biClrUsed;
      m = pDoc->pIH->biClrImportant;
      n = b - FHeader - IHeader;
      o = a - b;
      p = d * e;
      q = FHeader + IHeader + n + o;
      r = o-d*e*g/8;
      s[0].Format( "a) %d Bytes BITMAPFILEHEADER", FHeader );
      s[2].Format( "b) %d Bytes BITMAPINFOHEADER", IHeader );
      s[1].Format( "   ->Type=BM, Size=%d, OffBits=%d", a, b);
      s[3].Format( "   ->Size=%d, Width=%d, Height=%d, Planes=%d", c, d, e, f);
      s[4].Format( "   ->BitCount=%d, Compression=%d, SizeImage=%d", g, h, i );
      s[5].Format( "   ->XPelsPerMeter=%d, YPelsPerMeter=%d", j, k );
      s[6].Format( "   ->ClrUsed=%d, ClrImportant=%d", l, m );
      s[7].Format( "c) %d Bytes in der Palette = %d RGBQUADs", n, n/4 );
      s[8].Format( "d) %d Bytes für %d Pixel", o, p );
      s[9].Format( "a) + b) + c) + d) = %d Bytes = %0.3f KiloBytes",q,
(double)q/1024. );
      s[10].Format( "Click on left mouse button !" );
      for ( i = 0; i < 11; i++ ) pDC->TextOut(0,i*15,s[i]);
      break;
   }
}
```

`bitmap1` ist vorläufig fertig, ausführen, *.bmp-File öffnen, nicht in die ClientArea klicken, beenden.

Weiterer Code in CBitmap1View::OnDraw(CDC* pDC)

Erweitern Sie die Funktionalität von `Cbitmap1View::OnDraw(CDC* pDC)` durch weitere `case`-Blöcke, die bei jedem weitern Klick der linken Maustaste ein Bild zeigen. Sie müssen den Code direkt unter das letzte `break;`, aber noch vor die beiden letzten geschweiften Klammern einsetzen.

```
    case 1:  // Bild in Originalgroesse
      StretchDIBits( pDC->GetSafeHdc(),
                     0, 0, xSize, ySize,  //Originalgroesse
                     0, 0, xSize, ySize,
                     &(pDoc->Pixel.front()), pDoc->pI,
                     DIB_RGB_COLORS, SRCCOPY );
      pDC->TextOut(0,ySize,"Click on left mouse button !");
      break;
    case 2:   // Bild fensterfüllend
      StretchDIBits( pDC->GetSafeHdc(),
                     0, 0, R.Width(), R.Height(),
                     0, 0, xSize, ySize,
                     &(pDoc->Pixel.front()), pDoc->pI,
                     DIB_RGB_COLORS, SRCCOPY );
      pDC->TextOut(0,R.Height()-20,"Change window size ! Click on left mouse
button !");
      break;
    case 3:  // Animation
      dx = R.Width()  / 20;
      dy = R.Height() / 20;
      for ( i = 0; i < 20; i++  )
         StretchDIBits( pDC->GetSafeHdc(),
                        0, 0, R.Width() - i*dx, R.Height() - i*dy,
                        0, 0, xSize, ySize,
                         &(pDoc->Pixel.front()), pDoc->pI,
                        DIB_RGB_COLORS, SRCCOPY );
      pDC->TextOut(0,R.Height()-20,"Change window size ! Click on left mouse
button !");
      break;
```

Sie öffnen ein Bild, klicken 4 mal mit der linken Maustaste in die ClientArea und sehen das Bild jedesmal anders. Der Vorgang ist zyklisch beliebig oft wiederholbar.
FirstBitmap ist fertig, ausführen, Bitmaps von Harddisk laden, beenden, löschen, neu programmieren.

Beim Verkleinern der Bilder unter ihre Originalgröße (z.B. durch ziehen am Fensterrahmen) können Farbfehler im Bild auftreten. Das hängt von Ihrer Graphikkarte, Ihrem Graphkkartentreiber und der gewählten Bildschirmauflösung ab. Wenn Sie diese Fehler zuverlässig vermeiden wollen, dann müssen Sie in `void Cbitmap1View::OnDraw(CDC* pDC)` vor die `switch`-Zeile noch folgende Zeile einfügen:
`SetStretchBltMode( pDC->GetSafeHdc(), COLORONCOLOR );`
Diese Zeile verhindert, dass beim Weglassen von Zeilen und Spalten neuen Farben aus den alten Farben interpoliert werden.
Das macht nämlich `StretchDIBits` standardmäßig um die Bildqualität so gut es geht zu erhalten. Das ist zwar gut gemeint, führt aber im HighColorMode oft zu Farbfehlern.

Beispielbilder

Im Prinzip sollte das Programm alle Arten von Bitmaps lesen und anzeigen. Falls Sie eine alte Graphikkarte mit 8 Bit benutzen und/oder falls Sie Ihren Desktop auf 256 Farben eingestellt haben, kann es sein, dass die Farben ganz schlecht aussehen.
Falls Sie keine *.bmp - Dateien auf Ihrer Harddisk finden, benutzen Sie folgende Beispielbilder:

Download: Butterfly.bmp 217 kB 24Bit-TrueColor-Bild
Download: Madonna.bmp 18 kB 8Bit-Grauwert-Bild
Download: Lena256.bmp 66 kB 8Bit-Grauwert-Bild
Download: Lena512.bmp 258 kB 8Bit-Grauwert-Bild
Download: Angiography.bmp 66 kB 8Bit-Grauwert-Bild