Views und Mehr

Ziel, Inhalt

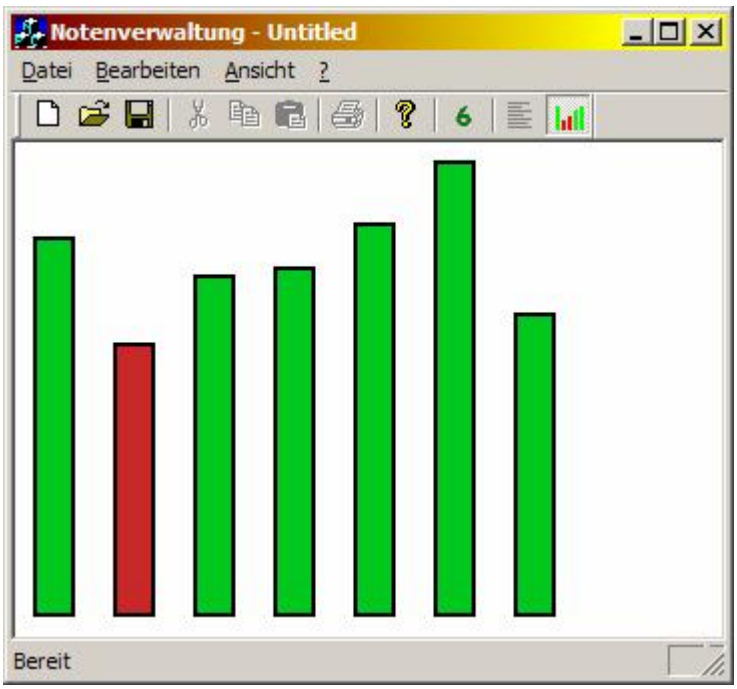
- Wir ergänzen heute bei unserer BarView die Scrollbars und erreichen mit einfachem Code auch gleich Skalierbarkeit.

Views und Mehr

Eine View mit Scrollbars

Darstellung mit fixer Grösse

Beim Beispiel mit den Noten haben wir eine View erzeugt (CBarView), die direkt von CView ableitet:

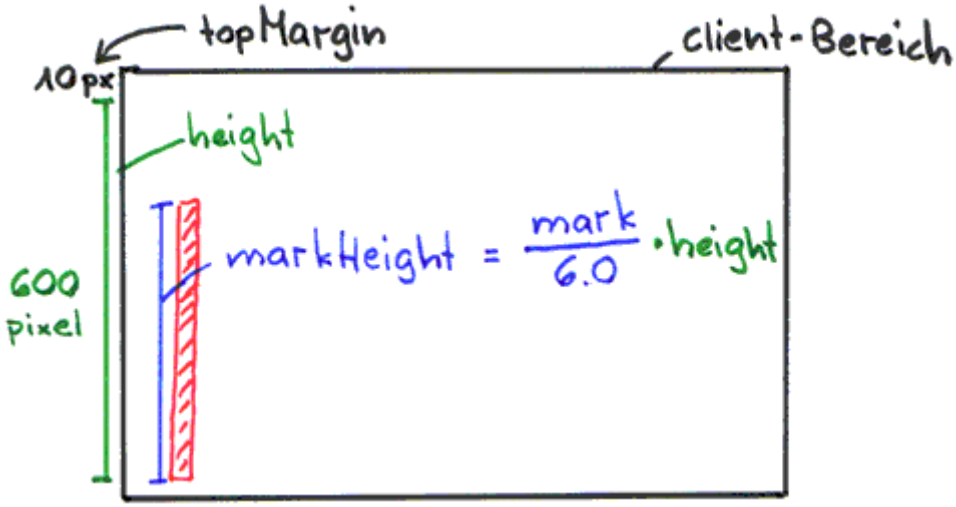


Noten in der CBarView-Darstellung

Wollen wir eine Darstellung bei der die Balken immer gleich gross sind, also die Ansicht nicht mit der Grösse des Fensters skaliert wird, vereinfacht sich unserer Zeichnungscode:

```
////////////////////////////////////////////////////////////
// CBarView drawing
void CBarView::OnDraw(CDC* pDC)
{
    CNotenVerwaltungTest1Doc* doc = GetDocument();

    const int height = 400; // Höhe des Balken für eine 6
    const int topMargin = 10; // oberer Rand

    CBrush redBrush; // Brush für roten Balkenhintergrund
    redBrush.CreateSolidBrush(RGB(200,40,40));
    CBrush greenBrush; // Brush für grünen Balkenhintergrund
    greenBrush.CreateSolidBrush(RGB(0, 200, 30));
    // Grüne Brush selektieren und Zeiger
    // auf alten Brush merken
    CBrush* oldBrush = pDC->SelectObject(&greenBrush);

    // Schwarzer Pen für Rahmen
    // um die Balken
    CPen pen;
    pen.CreatePen(PS_SOLID, 2, RGB(0,0,0));
    // alter Pen ebenso wie alten Brush
    // in temporärer Variable speichern
    CPen* oldPen = pDC->SelectObject(&pen);

    const NotenArray& array = doc->getNotenArray();
    int arraySize = array.GetSize();
    for(int i = 0; i < arraySize; ++i)
    {
        CNote* note = array[i];
        double notenWert = note->getMark();

        // so wird eine Note unter 4 rot
        if(notenWert < 4.0)
        {
            pDC->SelectObject(redBrush);
        }
        else
        {
            pDC->SelectObject(greenBrush);
        }

        int markHeight = (int)(notenWert * height / 6.0);
        CRect notenRect(10+i*40,
                        topMargin + height - markHeight,
                        30+i*40,
                        topMargin + height);

        pDC->Rectangle(notenRect);
    }

    pDC->SelectObject(oldBrush);
    pDC->SelectObject(oldPen);
}
```

Beachte, dass hier die Zeichnung am oberen linken Ecken „festgemacht“ ist. Wenn man das Fenster vergrössert, bleibt die Darstellung am oberen linken Ecken „hängen“. Dieses Verhalten ist nötig für das, was wir jetzt gleich machen.



Hilfe zur Berechnung der Zeichnung

Von der CView zur CScrollView

Es ist nun ziemlich einfach Scrollbalken für unsere View zu erhalten. Die MFC sieht hierfür eine spezielle CView-Klasse vor, welche sich um alles kümmert. Wir können diese einsetzen, da wir von der Klasse CView abgeleitet haben. Wir ersetzen einfach CView mit CScrollView als Basisklasse.

```
class CBarView : public CScrollView
{
public:
	CBarView();
protected:
	DECLARE_DYNCREATE(CBarView)
```

In der .cpp Datei müssen wir an zwei Stellen eine kleine Änderung vornehmen:

```
IMPLEMENT_DYNCREATE(CBarView, CScrollView)
```

und bei der Message Map:

```
BEGIN_MESSAGE_MAP(CBarView, CScrollView)
	//{{AFX_MSG_MAP(CBarView)
```

Setzen der Scrollbars

Die CScrollView Klasse bietet zur Unterstützung eine Methode SetScrollSizes. Diese Methode nimmt als erstes Argument einen so genannten Mapping Mode. Mit den Mapping Modes sagen wir dem Gerätekontext mit was für Einheiten wir den Bildschirm behandeln wollen. Es ist möglich physikalische Längeneinheiten zu verwenden, wie zum Beispiel 0.1 Millimeter pro Einheit. Wenn wir also ein Rechteck machen wollen, dass 10 Millimeter hoch ist, setzen wir zuerst den Mapping Mode auf MM_LOMETRIC und zeichnen das Rechteck 100 Einheiten hoch.
Der normale Modus ist MM_TEXT, wo eine Einheit einem Pixel entspricht. Hier geben wir diesen Modus der SetScrollSizes Methode mit, die dadurch weiss wie das Rechteck, das als zweites Argument mitgegeben wird zu betrachten ist.
Als erstes verschieben wir aber einige Werte als Konstanten an den Anfang unserer .cpp Datei:

```
const int Height    = 400;  // Höhe des Balken für eine 6
const int TopMargin = 10;   // oberer Rand
const int BarWidth  = 40;   // Breite des Balken
const int BarMargin = 10;   // Abstand der Balken
```

Dadurch ändert sich unser Zeichnungscode in Draw so:

```
for(int i = 0; i < arraySize; ++i)
{
    CNote* note = array[i];
    double notenWert = note->getMark();

    // so wird eine Note unter 4 rot
    if(notenWert < 4.0)
    {
        pDC->SelectObject(&redBrush);
    }
    else
    {
        pDC->SelectObject(&greenBrush);
    }

    int markHeight = (int)(notenWert * Height / 6.0);
    CRect notenRect(BarMargin + (BarMargin + BarWidth) * i,
                    TopMargin + Height - markHeight,
                    (BarMargin + BarWidth) * (i+1),
                    TopMargin + Height);

    pDC->Rectangle(notenRect);
}
```

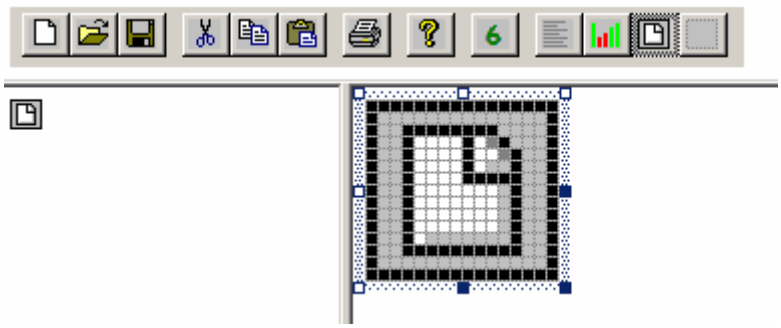
Die Methode SetScrollSizes muss jeweils aufgerufen werden, wenn sich das Dokument und damit unsere Darstellung ändert. Wenn sich das Dokument ändert werden unsere Update-Methoden aufgerufen:

```
void CBarView::OnInitialUpdate()
{
    CView::OnInitialUpdate();
    setScrollBars();
}

void CBarView::OnUpdate(CView* pSender,
                        LPARAM lHint, CObject* pHint)
{
    setScrollBars();
    Invalidate();
}

void CBarView::setScrollBars()
{
    CNotenVerwaltungTest1Doc* doc = GetDocument();

    const NotenArray& array = doc->getNotenArray();

    int arraySize = array.GetSize();
    int docSize = (BarMargin + BarWidth) *
                  arraySize + BarMargin;
    CSize size(docSize, 2*TopMargin + Height);
    SetScrollSizes(MM_TEXT, size);
}
```

Wir haben hier eine Methode setScrollBar, die einfach bei Bedarf aufgerufen werden kann.

Skalierte Ansicht

Mit der CScrollView ist es auch möglich eine skalierte Ansicht zu verwenden. Das führt dazu, dass unsere Noten immer das Fenster ausfüllen. Als erstes brauchen wir eine Möglichkeit dies ein und auszuschalten.
Erzeuge einen Menupunkt als Untermenu zur Balkenansicht und versuche auch einen Toolbar-Button zu erzeugen:



Ein Toolbar-Button für die skalierte Ansicht

Mit dem Klassenassistenten fügst du nun einen Nachrichtenbehandler für diese neue Control ID ein und einen Behandler für das CCmdUI:

```
void CBarView::OnBalkenansichtSkaliert()
{
    _scaledView = !_scaledView;
    setScrollBars();
    Invalidate();
}

void CBarView::OnUpdateBalkenansichtSkaliert(CCmdUI *pCmdUI)
{
    pCmdUI->SetCheck(_scaledView ? 1 : 0);
}
```

Das Datenelement _scaledView ist vom Datentyp bool und dienst als Flag ob eine skalierte Ansicht erwünscht ist oder nicht.
Es fehlt nur noch eine kleine Ergänzung in der setScrollBars Methode:

```
void CBarView::setScrollBars()
{
    CNotenVerwaltungTest1Doc* doc = GetDocument();

    const NotenArray& array = doc->getNotenArray();

    int arraySize = array.GetSize();
    int docSize = (BarMargin + BarWidth) *
                  arraySize + BarMargin;
    CSize size(docSize, 2*TopMargin + Height);
    if(_scaledView)
    {
        SetScaleToFitSize(size);
    }
    else
    {
        SetScrollSizes(MM_TEXT, size);
    }
}
```

Text ausgeben

Es ist einfach ein wenig Text zu ergänzen:

```
for(int i = 0; i < arraySize; ++i)
{
    CNote* note = array[i];
    double notenWert = note->getMark();

    // so wird eine Note unter 4 rot
    if(notenWert < 4.0)
    {
        pDC->SelectObject(redBrush);
    }
    else
    {
        pDC->SelectObject(greenBrush);
    }

    int markHeight = (int)(notenWert * Height / 6.0);
    CRect notenRect(BarMargin + (BarMargin + BarWidth) * i,
                    TopMargin + Height - markHeight,
                    (BarMargin + BarWidth) * (i+1),
                    TopMargin + Height);

    pDC->Rectangle(notenRect);

    CString notenString = note->asString();
    // Text transparent darstellen
    pDC->SetBkMode(TRANSPARENT);
    // Textfarbe auf weiss setzen
    pDC->SetTextColor(RGB(255,255,255));
    // TextOut nimmt eine x und eine y Koordinate
    pDC->TextOut(5+BarMargin + (BarMargin + BarWidth) * i,
                 TopMargin + Height - markHeight + 10,
                 notenString);
}
```

Drucken

Vorbereitungen für das Drucken

Auch das Drucken ist schon halb fertig mit der MFC. Als erstes müssen wir in unserer CBarView einige COMMANDS behandeln, indem wir sie in unsere Message Map eintragen:

```
BEGIN_MESSAGE_MAP(CBarView, CScrollView)
      //{{AFX_MSG_MAP(CBarView)
            // NOTE - the ClassWizard will add and remove
            // mapping macros here.
      //}}AFX_MSG_MAP
    ON_COMMAND(ID_BALKENANSICHT_SKALIERT,
               OnBalkenansichtSkaliert)
    ON_UPDATE_COMMAND_UI(ID_BALKENANSICHT_SKALIERT,
                         OnUpdateBalkenansichtSkaliert)
    ON_COMMAND(ID_FILE_PRINT, CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT,
               CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW,
               CScrollView::OnFilePrintPreview)
END_MESSAGE_MAP()
```

Druckmethoden überschreiben

Zusätzlich müssen wir zwei virtuelle Methoden von CView überschreiben (am besten mit Hilfe des Klassenassistenten):

```
void CBarView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo)
{
    if(pDC->IsPrinting())
    {
        // aus jeder Einheit einen
        // Zehntel Millimeter machen!
        pDC->SetMapMode(MM_LOMETRIC);
    }

    CScrollView::OnPrepareDC(pDC, pInfo);
}

BOOL CBarView::OnPreparePrinting(CPrintInfo* pInfo)
{

    DoPreparePrinting(pInfo);

    return CScrollView::OnPreparePrinting(pInfo);
}
```