

# Scaling of data into dc

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-05/msg00664.html>

---

- *From:* Matthias Pospiech <[matthias79@xxxxxx](mailto:matthias79@xxxxxx)>
  - *Date:* Fri, 11 May 2007 08:36:53 +0200
- 

Looking for help with stretching of data into a dc...

I have data I want to draw into a dc. This data has mostly higher or lower sizes than the dc to draw to. Therefore I need to stretch or compress the data while drawing into the dc.

My approach was to combine StretchBlt with the class CMemDC. That means that I do in CMemDC the following:

```
m_pDC->StretchBlt(
m_rect_dest.left, // x-coord of destination upper-left corner
m_rect_dest.top, // y-coord of destination upper-left corner
m_rect_dest.Width(), // width of destination rectangle
m_rect_dest.Height(), // height of destination rectangle
this, // handle to source DC
0, // x-coord of source upper-left corner
0, // y-coord of source upper-left corner
m_rect_source.Width(), // width of source rectangle
m_rect_source.Height(), // height of source rectangle
SRCCOPY);
```

That however results in wrong sizes if the data is larger than the dc. For example if the data has the size 100x100 and the dc 50x50 then it plots with sizes 25x25 instead of 50x50.

The opposite the enlargement however does work.

Now I am totally confused what is going wrong since all my Rect sizes are correct.

An example with all the source code can be found at:  
<http://www.pospiech.eu/download/bitmapopenandsave.zip>

The main code is

```
-----
ifndef _MEMDC_H_
#define _MEMDC_H_

class CMemDC : public CDC {
private:
```

## Scaling of data into dc

```
CBitmap m_bitmap; // Offscreen bitmap
CBitmap* m_oldBitmap; // bitmap originally found in CMemDC
CDC* m_pDC; // Saves CDC passed in constructor
CRect m_rect_dest; // Rectangle of drawing area.
CRect m_rect_source; // Rectangle of source area.
BOOL m_bMemDC; // TRUE if CDC really is a Memory DC.
public:

CMemDC(CDC* pDC, const CRect* pSourceRect = NULL, const CRect* pDestRect = NULL) : CDC()
{
    ASSERT(pDC != NULL);

    // Some initialization
    m_pDC = pDC;
    m_oldBitmap = NULL;
    m_bMemDC = !pDC->IsPrinting();

    // Get the rectangle to draw
    if (pDestRect == NULL) {
        pDC->GetClipBox(&m_rect_dest);
    } else {
        m_rect_dest = *pDestRect;
    }
    // Get the rectangle to draw
    if (pSourceRect == NULL) {
        pDC->GetClipBox(&m_rect_source);
    } else {
        m_rect_source = *pSourceRect;
    }

    if (m_bMemDC) {
        // Create a Memory DC
        CreateCompatibleDC(pDC);
        pDC->LPtoDP(&m_rect_dest);

        m_bitmap.CreateCompatibleBitmap(pDC, m_rect_dest.Width(), m_rect_dest.Height());
        m_oldBitmap = SelectObject(&m_bitmap);

        SetMapMode(pDC->GetMapMode());

        SetWindowExt(pDC->GetWindowExt());
        SetViewportExt(pDC->GetViewportExt());

        pDC->DPtoLP(&m_rect_dest);
        SetWindowOrg(m_rect_dest.left, m_rect_dest.top);
    } else {
        // Make a copy of the relevant parts of the current DC for printing
        m_bPrinting = pDC->m_bPrinting;
        m_hDC = pDC->m_hDC;
        m_hAttribDC = pDC->m_hAttribDC;
    }
}
```

## Scaling of data into dc

```
// Fill background
FillSolidRect(m_rect_dest, pDC->GetBkColor());
}

~CMemDC()
{
if (m_bMemDC) {

m_pDC->StretchBlt(
m_rect_dest.left, // x-coord of destination upper-left corner
m_rect_dest.top, // y-coord of destination upper-left corner
m_rect_dest.Width(), // width of destination rectangle
m_rect_dest.Height(), // height of destination rectangle
this, // handle to source DC
0, // x-coord of source upper-left corner
0, // y-coord of source upper-left corner
m_rect_source.Width(), // width of source rectangle
m_rect_source.Height(), // height of source rectangle
SRCCOPY);

//Swap back the original bitmap.
SelectObject(m_oldBitmap);
} else {
// All we need to do is replace the DC with an illegal value,
// this keeps us from accidently deleting the handles associated with
// the CDC that was passed to the constructor.
m_hDC = m_hAttribDC = NULL;
}
}

// Allow usage as a pointer
CMemDC* operator->()
{
return this;
}

// Allow usage as a pointer
operator CMemDC*()
{
return this;
}
};

#endif

-----

and

-----

void CGraphCtrl::OnPaint()
{
```

## Scaling of data into dc

```
CPaintDC dc(this); // device context for painting

CRect rc_dest;
rc_dest=GetSize();

CRect rc_source;
rc_source=GetPlotDataSize();

CMemDC pDC(&dc,&rc_source, &rc_dest); // Double Buffering
PlotToDC(& pDC);

}
```

---