

# KV Betriebssysteme

## Windowsprogrammierung

KV Betriebssysteme  
Windowsprogrammierung  
Zeichenfunktionen

**Peter R. Dietmüller**  
Inst. f. Informationsverarbeitung und Mikroprozessortechnik (FIM)  
Johannes Kepler Univ. Linz, Altenbergerstr. 69, A-4040 Linz, Austria  
E-Mail: dietmueller@fim.uni-linz.ac.at

Peter R. Dietmüller KV Betriebssysteme 1

Display Device Context

**Typ:**

- **common:** Device Context für die Client Area
- **window:** Device Context für das gesamte Fenster
- **class, parent, private:** Spezielle Varianten

**Bei WM\_PAINT:**

- Anfordern mit `BeginPaint` (meist Typ `common`).
- Freigabe mit `EndPaint`.

**Direkte Ausgabe im Programm**

- Anfordern mit `GetDC` (meist Typ `common`) oder `GetWindowDC` (Typ `window`).
- Freigabe mit `ReleaseDC`.

Peter R. Dietmüller KV Betriebssysteme 2

Zeichenfunktionen

**MoveToEx**

- `BOOL MoveToEx(HDC hdc, int x, int y, LPPOINT lpPoint);`
- Setzt die aktuelle Zeichenposition auf einen bestimmten Punkt. Gibt die alte Zeichenposition in `lpPoint` zurück. Bei Erfolg ist der Rückgabewert ungleich 0.

**LineTo**

- `BOOL LineTo(HDC hdc, int x, int y);`
- Zieht eine Linie von der aktuellen Zeichenposition zu einem bestimmten Punkt und versetzt die Zeichenposition dorthin. Zum Zeichnen wird der aktuelle Stift verwendet. Bei Erfolg ist der Rückgabewert ungleich 0.

• **ACHTUNG: Der Enpunkt wird NICHT gezeichnet!**

Peter R. Dietmüller KV Betriebssysteme 3

Zeichenfunktionen

**Rectangle**

- `BOOL Rectangle(HDC hdc, int left, int top, int right, int bottom);`
- Ein Rechteck wird mit dem aktuellen Stift gezeichnet. Der innere Bereich wird mit dem aktuellen Pinsel gefüllt. Zu beachten ist, daß das Rechteck nur bis `right-1` und `bottom-1` gezeichnet wird. Bei Erfolg ist der Rückgabewert ungleich 0.

**RGB**

- `COLORREF RGB(BYTE red, BYTE green, BYTE blue);`
- Dieses Makro erzeugt aus einzelnen RGB-Werten einen 32 Bit Wert vom Typ `COLORREF`, der eine Farbe exakt definiert.

Peter R. Dietmüller KV Betriebssysteme 4

Zeichenfunktionen

**FillRect**

- `int FillRect(HDC hdc, CONST RECT *lprect, HBRUSH hbr);`
- Ein Rechteck wird mit dem angegebenen Pinsel gefüllt (ohne Rand!). Zu beachten ist, daß das Rechteck nur bis `right-1` und `bottom-1` gezeichnet wird. Bei Erfolg ist der Rückgabewert ungleich 0.

**RECT**

- `typedef struct {`  
    `LONG left;           LONG top;`  
    `LONG right;         LONG bottom`  
    `} RECT;`

Peter R. Dietmüller KV Betriebssysteme 5

Zeichenfunktionen

**TextOut**

- `BOOL TextOut(HDC hdc, int x, int y, LPCTSTR text, int textLen);`
- Der String `text` wird an der angegebenen Position gezeichnet. Es wird der aktuelle Zeichensatz, die aktuelle Hintergrund- und Textfarbe verwendet. Der String muß nicht nullterminiert sein, die Stringlänge wird in `textLen` übergeben. Bei Erfolg ist der Rückgabewert ungleich 0.
- `UINT SetTextAlign(HDC hdc, UINT mode)`
- Wie der Text mit der Position kombiniert wird:
  - `TA_BOTTOM, TA_CENTER, TA_LEFT, TA_UPDATECP...`

Peter R. Dietmüller KV Betriebssysteme 6

# KV Betriebssysteme

## Windowsprogrammierung

### Zeichenfunktionen

#### Ellipse

- **BOOL Ellipse** ( HDC hdc, int left, int top, int right, int bottom);
- Eine Ellipse bzw. Kreis wird mit dem aktuellen Stift gezeichnet. Der innere Bereich wird mit dem aktuellen Pinsel gefüllt. Mittelpunkt der Ellipse ist der Mittelpunkt des Rechtecks. Dieses spezifiziert auch den Umfang. Bei Erfolg ist der Rückgabewert ungleich 0.

Peter R. Diemüller

KV Betriebssysteme

7

### Zeichenfunktionen

#### PolyLine

- **BOOL PolyLine**(HDC hdc, CONST POINT \*lppt, int count);
- Zeichnet eine Anzahl von verbundenen Linien. Die aktuelle Position wird ignoriert (d. h. Start beim ersten Punkt von lppt). Die Anzahl der Punkte muß in count übergeben werden (=Linienanzahl-1). Bei Erfolg ist der Rückgabewert ungleich 0.

#### POINT

```
typedef struct {
    LONG x;
    LONG y;
} POINT;
```

Peter R. Diemüller

KV Betriebssysteme

8

### Zeichenfunktionen

#### Polygon

- **BOOL Polygon**(HDC hdc, CONST POINT \*lppt, int count);
- Zeichnet ein Polygon mit dem derzeitigen Stift und füllt es mit dem aktuellen Pinsel. Die aktuelle Position wird ignoriert (d. h. Start beim ersten Punkt von lppt). Die Anzahl der Punkte muß in count übergeben werden (=Linienanzahl-1). Das Polygon wird automatisch geschlossen (Verbindung letzter-erster Punkt). Bei Erfolg ist der Rückgabewert ungleich 0.

Peter R. Diemüller

KV Betriebssysteme

9

### Zeichenfunktionen

#### ExtFloodFill

- **BOOL ExtFloodFill**( HDC hdc, int xStart, int yStart, COLORREF crFill, UINT fillType);
- Füllt eine Fläche mit dem derzeitigen Füllmuster (brush), startend an der Position xStart/yStart. Die Fläche muß mit der Farbe crFill begrenzt/gefüllt sein. Bei Erfolg ist der Rückgabewert ungleich 0.
- fillType: **FLOODFILLBORDER** Farbe ist Randbegrenzung  
**FLOODFILLSURFACE** Farbe ist die zu füllende Fläche
- Achtung: xStart/yStart muß innerhalb der clipping region liegen, sonst wird ein Fehler zurückgegeben!

Peter R. Diemüller

KV Betriebssysteme

10

### Zeichenfunktionen

#### SetPixel

- **COLORREF SetPixel**( HDC hdc, int x, int y, COLORREF color);
- Der Rückgabewert ist die Farbe des gezeichneten Punkts. Diese kann von color abweichen, wenn für die gewünschte Farbe eine Näherung verwendet werden mußte. Der Rückgabewert ist -1, wenn der Punkt außerhalb des Clippingbereichs liegt.
- Eine Näherung für die Wunschfarbe muß verwendet werden, wenn aufgrund einer geringen Farbtiefe der Anzeige nicht jede RGB - Kombination zur Verfügung steht.
- Nicht jedes Ausgabegerät muß SetPixel unterstützen.

Peter R. Diemüller

KV Betriebssysteme

11

### Zeichenattribute in einem DC

#### Bitmap

- CreateBitmap, CreateBitmapIndirect, CreateCompatibleBitmap, ...

#### Pinsel (Brush)

- CreateHatchBrush, CreatePatternBrush, CreateSolidBrush, ...

#### Schriftart (Font)

- CreateFont, CreateFontIndirect

#### Stift (Pen)

- CreatePen, CreatePenIndirect

#### Region

- CreateEllipticRgn, CreatePolygonRgn, CreateRectRgn, ...

Peter R. Diemüller

KV Betriebssysteme

12

# KV Betriebssysteme

## Windowsprogrammierung

### Zeichenobjekte

#### Erzeugen

- Funktionen CreateXXX und CreateIndirectXXX
- Bei den CreateIndirectXXX Funktionen werden die Objekteigenschaften in einer eigenen Struktur definiert, sonst direkt als Parameter.

#### Freigeben

- Funktion DeleteObject
- Ein Objekt darf zu diesem Zeitpunkt nicht mehr in einem Gerätekontext selektiert sein. Dazu kann man den ursprünglichen Wert wieder selektieren.

### Zeichenobjekte

```
HPEN hPen, hPenOld;  
HDC hdc;
```

```
hPen = CreatePen(PS_SOLID, 6, RGB(0, 0, 255));  
hPenOld = SelectObject(hdc, hPen);
```

... (z.B. mit dem neuen Stift zeichnen)

```
SelectObject(hdc, hPenOld);  
DeleteObject(hPen);
```

### Freigeben eines Zeichenobjektes

#### DeleteObject

- BOOL DeleteObject(HGDIOBJ obj);
- Das bezeichnete Objekt und die damit verbundenen Ressourcen werden freigegeben. Bei Erfolg ist der Rückgabewert ungleich 0.
- Ein Objekt darf nicht in einem Gerätekontext selektiert sein, wenn es gelöscht werden soll.

### Erzeugen eines Stift

#### CreatePen

- HPEN CreatePen(int penStyle, int width, COLORREF color);
- Ein Stift mit der angegebenen Farbe und Dicke wird im angegebenen Stil generiert. Der Rückgabewert der Funktion ist ein Handle auf den Stift oder NULL, wenn der Aufruf fehlgeschlagen ist.
- Einige mögliche Werte für penStyle:
  - PS\_SOLID durchgehend gezeichnet
  - PS\_DOT Stift zeichnet punktiert, nur mit Stiftbreite 1
- Die Breite wird in **logischen Einheiten** spezifiziert
  - Breite=0: IMMER genau einen Pixel breit (bzw. Dünnsste mögliche Linie)

### Erzeugen eines Pinsel

#### CreateBrushIndirect

- HBRUSH CreateBrushIndirect(CONST LOGBRUSH\* lplb);
- Ein Pinsel mit den in der Struktur lplb definierten Eigenschaften wird generiert. Der Rückgabewert der Funktion ist ein Handle auf den Pinsel oder NULL, wenn der Aufruf fehlgeschlagen ist.
- typedef struct {  
    UINT lbStyle;  
    COLORREF lbColor; /\* Farbe des Pinsels\*/  
    int lbHatch;  
} LOGBRUSH;

### Erzeugen eines Pinsel

#### Für lbStyle existieren u. a. die folgenden Konstanten:

- BS\_SOLID Pinsel für einfache Farbflächen
- BS\_HATCHED Pinsel für Schraffuren

Ist lbStyle auf BS\_SOLID gesetzt, wird der Wert von lbHatch ignoriert. Für Pinsel mit dem BS\_HATCHED Stil existieren für lbHatch u. a. folgende Konstanten:

- HS\_BDIAGONAL 45 Grad steigend schraffiert
- HS\_FDIAGONAL 45 Grad fallend schraffiert
- HS\_HORIZONTAL horizontale Schraffur
- HS\_VERTICAL vertikale Schraffur
- HS\_CROSS horizontale und vertikale Schraffur

# KV Betriebssysteme

## Windowsprogrammierung

### Erzeugen einer Schriftart

#### CreateFontIndirect

- **HFONT CreateFontIndirect(LOGFONT\* lplf);**
- Ein Zeichensatz mit den in der Struktur lplf definierten Eigenschaften wird selektiert. Der Rückgabewert der Funktion ist ein Handle auf den Zeichensatz oder NULL, wenn der Aufruf fehlgeschlagen ist.

### Erzeugen einer Schriftart

```
typedef struct {
    LONG lfHeight;           /* Zeichenhöhe, auch negative Werte mögl. */
    LONG lfWidth;           /* Zeichenbreite */
    LONG lfEscapement;      /* Drehung */
    LONG lfOrientation;     /* Drehung */
    LONG lfWeight;         /* Strichstärke, 1 - 1000, 400 = normal */
    BYTE lfItalic;         /* Kursiv */
    BYTE lfUnderline;      /* Unterstrichen */
    BYTE lfStrikeOut;      /* Durchgestrichen */
    BYTE lfCharSet;        /* Zeichensatz (Griechisch, Japanisch, ...) */
    BYTE lfOutPrecision;    /* Genauigkeit am Ausgabegerät */
    BYTE lfClipPrecision;   /* Clipping-Genauigkeit */
    BYTE lfQuality;        /* Darstellungsqualität, z.B. Antialiasing */
    BYTE lfPitchAndFamily;  /* Familie der Schriftart, z.B. Roman */
    TCHAR lfFaceName[LF_FACESIZE]; /* Name der Schriftart (max. 31 chars + 0) */
} LOGFONT;
```

### Erzeugen einer Schriftart

#### lfHeight

- Siehe nächste Seite

#### lfFaceName

- Name der gewünschten Schriftart

#### Alle anderen Felder

- Alle anderen Felder der Struktur lplf können einfach auf 0 gesetzt werden, wodurch das System eine „vernünftige“ Wahl trifft.

### Erzeugen einer Schriftart

#### lfHeight

- **Gibt die Höhe der Schriftart (in logischen Einheiten) an:**
  - > 0: Höhe der Zeichen-Zelle
  - < 0: Zeichen-Höhe (=Höhe der Zeichen-Zelle - Internal-leading)
- **Die übliche Zeichensatzgröße in Punkten bezieht sich auf 72 dpi Auflösung und muß auf die tatsächliche Bildschirmauflösung umgerechnet werden.**
  - Mapping Modus MM\_TEXT:  
lfHeight=-MulDiv(points,GetDeviceCaps(hdc,LOGPIXELSY),72);
- **GetDeviceCaps(hdc, LOGPIXELSY) liefert die vertikale Auflösung des Bildschirms in dpi.**

### Erzeugen einer Schriftart

```
HFONT hfont, hfontOld;
LOGFONT lf;
```

```
memset(&lf, 0, sizeof(LOGFONT));
lf.lfHeight = -MulDiv(pointSize, GetDeviceCaps(hdc, LOGPIXELSY), 72);
strcpy(lf.lfFaceName, "Times New Roman");
```

```
hfont = CreateFontIndirect(&lf);
hfontOld = SelectObject(hdc, hfont);
...
TextOut(hdc, 10, 50, "Test", 4);
...
SelectObject(hdc, hfontOld);
DeleteObject(hfont);
```

### Erzeugen einer Schriftart

#### MulDiv

- **int MulDiv( int nNumber, int nNumerator, int nDenominator);**
- Die Funktion multipliziert zwei 32-Bit Werte und dividiert das 64-Bit Ergebnis durch einen dritten 32-Bit Wert. Das Ergebnis wird auf den nächsten ganzzahligen Wert auf- bzw. abgerundet.
- Bei Erfolg wird das Ergebnis der Multiplikation und Division zurückgegeben, sonst (bei Überlauf oder Divisor = 0) wird -1 zurückgegeben.

# KV Betriebssysteme

## Windowsprogrammierung

### Wahl von Zeichenattributen

#### SelectObject

- `HGDIOBJ SelectObject(HDC hdc, HGDIOBJ obj);`
- Zunächst besitzt jeder Gerätekontext einen Standardwert für den aktuellen Stift, Pinsel und Zeichensatz. Diesen Standard kann man jederzeit durch andere Zeichenobjekte ersetzen.
- Das durch `obj` übergebene Zeichenobjekt wird in den Gerätekontext selektiert und ersetzt damit das vorher selektierte Objekt gleichen Typs. Der Rückgabewert der Funktion ist ein Handle auf das ersetzte Objekt oder `NULL`, wenn der Aufruf fehlgeschlagen ist.
- Man sollte immer den vorherigen Zustand wiederherstellen und daher den Rückgabewert speichern.

Peter R. Dietmüller

KV Betriebssysteme

25

### Stock Objects

#### Stock Objects

- „lagernde“ Objekte
- stellt das System immer zur Verfügung
- müssen nicht mit `CreateXXX` angelegt werden

#### GetStockObject

- `HGDIOBJ GetStockObject(int objConst);`
- In `objConst` wird übergeben, welches Zeichenobjekt benötigt wird. Der Rückgabewert ist ein Handle auf das gewünschte Objekt oder `NULL` wenn der Aufruf fehlgeschlagen ist.

Peter R. Dietmüller

KV Betriebssysteme

26

### Stock Objects

#### Einige der möglichen Werte für `objConst` sind:

<code>BLACK_BRUSH</code>	schwarzer Pinsel
<code>GRAY_BRUSH</code>	grauer Pinsel
<code>WHITE_BRUSH</code>	weißer Pinsel
<code>HOLLOW_BRUSH</code>	durchsichtiger Pinsel
<code>BLACK_PEN</code>	schwarzer Stift
<code>WHITE_PEN</code>	weißer Stift
<code>ANSI_FIXED_FONT</code>	nicht proportionaler Systemzeichensatz
<code>ANSI_VAR_FONT</code>	proportionaler Systemzeichensatz
<code>SYSTEM_FONT</code>	Systemzeichensatz, z.B. für Menüs verwendet
<code>DEFAULT_PALETTE</code>	Statische Farben der System-Palette

Peter R. Dietmüller

KV Betriebssysteme

27

### KV Betriebssysteme Windowsprogrammierung Fokus-Rechteck

Peter R. Dietmüller

Inst. f. Informationsverarbeitung und Mikroprozessortechnik (FIM)  
Johannes Kepler Univ. Linz, Altenbergerstr. 69, A-4040 Linz, Austria  
E-Mail: dietmueler@fm.uni-linz.ac.at

Peter R. Dietmüller

KV Betriebssysteme

28

### Fokus-Rechteck

#### Bei `WM_LBUTTONDOWN`

- Mausevents „fangen“: `SetCapture()`
- Fokus-Rechteck zeichnen: `DrawFocusRect()`

#### Bei `WM_MOUSEMOVE`

- Altes Fokus-Rechteck löschen: `DrawFocusRect(<Alte Pos.>)`
- Position/Größe aktualisieren
- Neues Fokus-Rechteck zeichnen: `DrawFocusRect(<Neue Pos.>)`

#### Bei `WM_LBUTTONUP`

- Altes Fokus-Rechteck löschen: `DrawFocusRect()`
- Mausevents „freigeben“: `ReleaseCapture()`

Peter R. Dietmüller

KV Betriebssysteme

29

### Fokus-Rechteck

#### SetCapture

- `HWND SetCapture(HWND hWnd);`
- Die Funktion veranlaßt, daß alle Mausevents an das angegebene Fenster gesendet werden, auch wenn die Maus aus dem Fenster hinausbewegt wird (und eine Taste gedrückt ist). Es kann nur ein Fenster die Mausevents einfangen.
- Bei Erfolg gibt die Funktion das Fenster zurück, das vorher alle Mausevents einfing. Bei einem Fehler wird `NULL` zurückgegeben.
- Wenn das Fenster nicht mehr alle Mausevents benötigt, muß die Funktion `ReleaseCapture` aufgerufen werden.

Peter R. Dietmüller

KV Betriebssysteme

30

# KV Betriebssysteme

## Windowsprogrammierung

### Fokus-Rechteck

#### ReleaseCapture

- **BOOL ReleaseCapture(VOID);**
- Die Funktion gibt die Mausevents wieder frei, sodaß danach wieder alle Fenster Mausevents zugesandt bekommen.
- Bei Erfolg wird ein Wert ungleich Null zurückgegeben.

### KV Betriebssysteme Windowsprogrammierung Bitmaps

**Peter R. Dietmüller**

Inst. f. Informationsverarbeitung und Mikroprozessortechnik (FIM)  
Johannes Kepler Univ. Linz, Altenbergerstr. 69, A-4040 Linz, Austria  
E-Mail: dietmueller@fm.uni-linz.ac.at

### Arbeiten mit Bitmaps

#### Zeichenoperationen auf einer Bitmap im Speicher:

- passender Gerätekontext für die Bitmap, einen sog. „Speichergerätekontext“ (memory device context)
- Bitmap, die in diesen Speichergerätekontext selektiert wird

Zwischen Bitmap und Bildschirm können Ausschnitte in jede Richtung kopiert werden. Dazu müssen aber die internen Darstellungsformen beider Medien verträglich sein. Dazu verwendet man die Funktionen

- CreateCompatibleDC
- CreateCompatibleBitmap

### Arbeiten mit Bitmaps

#### CreateCompatibleDC

- **HDC CreateCompatibleDC(HDC hdc);**
- Erzeugt einen Speichergerätekontext, der zum in hdc übergebenen Gerätekontext kompatibel ist. Wenn hdc auf NULL gesetzt ist, wird ein zum Systembildschirm kompatibler Gerätekontext erzeugt. Der Rückgabewert ist ein Handle auf den neuen Speichergerätekontext oder NULL wenn der Aufruf fehlgeschlagen ist.
- Um auf einem Speichergerätekontext Zeichenoperationen durchführen zu können, muß eine Bitmap der gewünschten Größe erzeugt und in den Kontext selektiert werden.
- Ein mit dieser Funktion erzeugter Gerätekontext muß nach Verwendung mit DeleteDC freigegeben werden.
- Nur für Raster-Devices: Bildschirm, Drucker, ...

### Arbeiten mit Bitmaps

#### DeleteDC

- **BOOL DeleteDC(HDC hdc);**
- Ein mit Create...DC erzeugter Gerätekontext wird freigegeben. Zu diesem Zeitpunkt dürfen nur mehr Standard-Zeichenobjekte in den Gerätekontext selektiert sein. Der Rückgabewert ist ungleich 0 wenn der Aufruf erfolgreich war.

### Arbeiten mit Bitmaps

#### CreateCompatibleBitmap

- **HBITMAP CreateCompatibleBitmap( HDC hdc, int width, int height);**
- Es wird eine zum in hdc übergebenen Gerätekontext kompatible Bitmap erzeugt. Die Breite der Bitmap wird durch width und die Höhe durch height definiert.
- Wird durch hdc ein Speichergerätekontext bezeichnet, hat die neue Bitmap die gleichen Eigenschaften wie die Bitmap, die gerade in hdc selektiert ist. Ein neu erzeugter Speichergerätekontext hat automatisch eine monochrome Bitmap (ein stock object) selektiert.
- Der Rückgabewert ist ein Handle auf die neue Bitmap oder NULL wenn der Aufruf fehlgeschlagen ist.

# KV Betriebssysteme

## Windowsprogrammierung

### Arbeiten mit Bitmaps

#### BitBlt

- **BOOL BitBlt**( HDC dest, int xDest, int yDest, int width, int height, HDC source, int xSrc, int ySrc, DWORD ropCode);
- Eine Bitmap wird von einem auf einen anderen Gerätekontext kopiert. Kopiert wird von source nach dest. Die Größe des kopierten Rechtecks wird durch width und height bestimmt. Innerhalb des Zielgerätekontexts wird die linke obere Ecke der Bitmap mit xDest und yDest positioniert. Die linke obere Ecke innerhalb der Quelle wird durch xSrc und ySrc bestimmt. Durch ropCode wird die Art der Rasteroperation bestimmt.
- Einige vordefinierten Konstanten für ropCode sind:
  - SRCCOPY exakte Kopie
  - SRCINVERT kombiniere Ziel und Quelle mit XOR

### Arbeiten mit Bitmaps

#### GetObject

- int GetObject( HGDIOBJ hgdiojb, int cbBuffer, LPVOID lpvObject);
- Informationen über ein Graphik-Objekt werden festgestellt. Bei Bitmaps (hgdiojb ist vom Type HBITMAP) muß als lpvObject ein Zeiger auf eine (reservierte) BITMAP-Struktur übergeben werden.
- Hiermit kann etwa die Größe einer Bitmap festgestellt werden.

#### BITMAP

- typedef struct {  
LONG bmType;  
LONG bmWidth; LONG bmHeight;  
LONG bmWidthBytes; WORD bmPlanes;  
WORD bmBitsPixel; LPVOID bmBits;  
} BITMAP;

### Arbeiten mit Bitmaps

Bei komplexen graphischen Fensterinhalten ist es oft nicht möglich / zielführend, die Ausgabe des gesamten Fensterinhalts mit Zeichenprimitiva wie Punkt, Linie, Buchstabe, ... zu realisieren.

Mögliche Alternative: Alle während des Programmablaufs am Bildschirm durchgeführten Zeichenoperationen werden parallel im Hintergrund auf einer Bitmap im Speicher durchgeführt. Ist ein Update des gesamten Fensters notwendig, wird diese Bitmap auf den Bildschirm kopiert - schnell und speicheraufwendig!

Es gibt auch eine etwas einfachere Alternative: Alle Zeichenoperationen werden auf der Bitmap im Speicher durchgeführt und der Update des Bildschirms erfolgt immer über die Bitmap. Dadurch ergibt sich eine klarere Programmstruktur ohne Codeverdopplung (siehe Variante #2 der Bildschirmausgabe).

### Arbeiten mit Bitmaps

```
#define MAXWIDTH ...
#define MAXHEIGHT ...
HBITMAP hbmpOld, hbmp; HDC hdc, hdcmem; RECT r;
...
/* -- Initialisierung der Bitmap -- */
hdcmem = CreateCompatibleDC(NULL);
hbmp = CreateCompatibleBitmap(hdc, MAXWIDTH, MAXHEIGHT);
hbmpOld = SelectObject(hdcmem, hbmp);
r.left = 0; r.top = 0; r.right = MAXWIDTH; r.bottom = MAXHEIGHT;
FillRect(hdcmem, &r, GetStockObject(WHITE_BRUSH));
...
/* -- Kopieren auf den Bildschirm -- */
BitBlt(hdc, 0, 0, MAXWIDTH, MAXHEIGHT, hdcmem, 0, 0, SRCCOPY);
...
/* -- Aufräumen -- */
SelectObject(hdcmem, hbmpOld); DeleteDC(hdcmem);
```