4.1 Geeigneter Ort der Ausgabe

Sobald in Windows eine Ausgabe getätigt wird, wobei es unerheblich ist, ob sie auf dem Drucker oder auf dem Bildschirm erfolgt, wird vor allem mit dem GDI-Teil (Graphical Device Interface) von Windows zusammengearbeitet. Hier sind z.B. die API-Ausgabefunktionen definiert, die wir bei der Programmierung mit QuickC, TurboPascal und Borland C++ verwenden. GDI ist auch dafür verantwortlich, daß die Verbindung zu dem entsprechenden Gerätetreiber hergestellt werden kann.

4.1.1 Die Meldung WM_PAINT

Bei der Programmierung unter DOS war es dem Programmierer überlassen, zu welchem Zeitpunkt und in welcher Funktion er die Bildschirm-Ausgabe implementierte. Bei Windows gibt es dafür gewisse Richtlinien, die man beachten sollte. Die Ausgabe sollte im Normalfall nur bei der Meldung WM_PAINT oder einem dadurch angestoßenen Ereignis bzw. einer dadurch aktivierten Methode erfolgen.
Der Grund für diese Bestimmungen liegt darin, daß unter Windows gleichzeitig mehr als eine Applikation gestartet sein kann, die alle innerhalb ihrer Fenster Text oder Grafik ausgeben. Diese Fenster liegen häufig übereinander, so daß Teile der Ausgabe verdeckt sind. Sobald ein Fenster in den Vordergrund gebracht wird, muß aber die getätigte Ausgabe vollständig sichtbar sein, da der Anwender damit arbeiten möchte. Aus diesem Grund wurde in Windows die Meldung WM_PAINT eingeführt, die z.B. immer dann entsteht, wenn Windows feststellt, daß der Inhalt eines Teils oder des gesamten Arbeitsbereiches (Client Area) des Fensters nicht mehr aktuell ist. Diese Meldung wird in die entsprechende Application-Message-Queue gestellt, aus der sie dann wieder mit der Funktion GetMessage geholt und mit Hilfe von DispatchMessage an die Fensterroutine weitergeleitet wird.
Wenn das Fenster nur teilweise durch ein anderes überlagert war, wird bei der Meldung WM_PAINT die Information mitgegeben, welcher Bereich der Client Area neu aufbereitet werden muß. Dieser Teil wird als Clipping Region oder invalider Bereich bezeichnet. Windows erlaubt nur das Zeichnen in diese Clipping Region, aber nicht außerhalb, um sowohl aktuelle Bereiche der Client Area als auch andere Fenster vor einem unerlaubten Überschreiben zu schützen.

CS_HREDRAW, CS_VREDRAW

Eine WM_PAINT-Meldung kann aufgrund mehrerer Möglichkeiten entstehen. Das Fenster wird durch ein Anklicken auf den Bildschirm nach vorne geholt, wie bereits erwähnt wurde, oder der Anwender verändert die Größe des Fensters. Die beiden Stilparameter CS_HREDRAW für horizontal und CS_VREDRAW für vertikal sind in der WNDCLASS-Struktur gesetzt. In diesem Fall wird außer einer WM_SIZE-Meldung auch immer eine WM_PAINT-Meldung generiert, wobei die gesamte Client Area als ungültig markiert wird, damit der Inhalt insgesamt neu aufbereitet werden kann.
In einem QuickC-Programm, dessen Aussehen mit QuickCase:W erstellt wurde, werden diese zwei Konstanten automatisch dem style-Feld übergeben, wenn in der Dialogbox, die über den Menüpunkt Style aufgerufen wird, das erste Kontrollfeld (Redraw when Sized) markiert wurde. Bei einer in TurboPascal oder Borland C++ geschriebenen Applikation werden sie durch die Methode GetWindowClass der Klasse TWindow gesetzt. Falls die beiden Konstanten CS_HREDRAW und CS_VREDRAW nicht angegeben sind, wird nur dann die Meldung WM_PAINT generiert, wenn das Fenster vergrößert wird. Zudem wird nur der neu hinzugekommene Teil als ungültig markiert.
Wenn der Anwender das Fenster des ersten QuickC-Programms vergrößert, und das Flag CS_HREDRAW nicht gesetzt ist, wird bei der entstehenden WM_PAINT-Meldung nur der vergrößerte Fensterteil mitgeliefert. Nur in diesem ungültigen Bereich kann etwas ausgegeben werden. Bei gesetztem Stilparameter kann statt dessen die Ausgabe in die gesamte Client Area erfolgen, um z.B. bei einem mehrzeiligen Text neue Zeilenumbrüche erstellen zu können.
Bei bestimmten Aktionen versucht Windows selber, einen Teil des Bildschirms zu sichern, um ihn später wieder restaurieren zu können. Falls dies jedoch aus Mangel an Speicherplatz o.ä. nicht

möglich ist, sendet Windows WM_PAINT-Meldungen an die betroffenen Fenster. Diese Aktionen sind z.B. das Entfernen einer Dialogbox oder das Wegblättern eines Untermenüs. In bestimmten Situationen sichert Windows immer den überschriebenen Bildschirmbereich und stellt ihn anschließend wieder her. Der typische Fall dafür ist das Bewegen des Mauszeigers über die Client Area.

CS_SAVEBITS

Mit Hilfe der Konstanten CS_SAVEBITS, die genauso wie CS_HREDRAW als Stilparameter bei der Klassen-Registrierung gesetzt werden, ist es auch möglich, daß Windows selber den Bildschirmbereich als Bitmap sichert, der von einem anderen Fenster überlagert wird. Mit dieser Sicherung kann Windows dann den Bildschirm wiederherstellen, wenn das Fenster z.B. bewegt wird. Dabei wird das Bitmap an die ursprüngliche Stelle gezeichnet, und die zuvor verdeckten Fenster erhalten keine WM_PAINT-Meldung, da ja Windows die Aufbereitung übernimmt. Trotz dieser vermeintlichen Vorteile sollte die Konstante CS_SAVEBITS nur bei kleinen Fenstern gesetzt werden, die sich kurzzeitig auf dem Bildschirm befinden, da Windows für diese Sicherungs-Bitmaps sehr viel Speicherplatz benötigt. Auch wird durch CS_SAVEBITS die Zeit der Aufbereitung und Ausgabe des Fensters um einiges erhöht. Wir werden uns diese Möglichkeit anschließend bei VisualBasic ein wenig genauer ansehen, da sie dort über die Eigenschaft AutoRedraw realisiert wird.
Neben der Generierung einer WM_PAINT-Meldung durch Windows können auch wir als Programmierer fordern, daß diese Meldung erzeugt wird. Dafür gibt es zum einen die Funktion UpdateWindow, die eine WM_PAINT-Meldung als eine non-queued Meldung generiert, die direkt an die Window-Funktion gesendet wird, zum anderen existieren die beiden Funktionen InvalidateRect und InvalidateRgn, die viel häufiger als die zuerstgenannte Funktion benutzt werden. Der Grund dafür liegt in der Langsamkeit der Ausgabe. Eine Ausnahme stellt die Ausgabe eines neuen Abschnittes dar, wenn mit Scrollbars gearbeitet wird. In der Queue wird die WM_PAINT-Meldung immer als letzte Meldung an die anderen angehängt.
In vielen Programmen wird eine Ausgabe über einen Menüpunkt oder über eine bestimmte Taste angestoßen. Programmtechnisch gesehen wird jedoch der Text oder die Grafik nicht direkt bei einer WM_COMMAND-Meldung oder bei einer WM_CHAR-Meldung in das Fenster geschrieben, sondern hier wird nur die Funktion InvalidateRect aktiviert, die eine WM_PAINT-Meldung in die Application-Message-Queue stellt.
Die Funktion InvalidateRgn beinhaltet als zweiten Parameter keinen Zeiger auf eine Rechtecksstruktur, sondern ein Handle auf eine Region, die aus mehreren Teilbereichen zusammengesetzt sein kann.

Methode Refresh

Das Äquivalent zu der Funktion InvalidateRect ist in VisualBasic die Methode Refresh. Durch ihren Aufruf wird ein Paint-Ereignis erzeugt. Dies kann nicht nur bei einem Form, sondern auch bei Controls angewendet werden.
Auf die WM_PAINT-Meldung wird in QuickC in der Window-Funktion im switch-Zweig für die Meldungen reagiert. Dabei müssen die erste Funktion BeginPaint und die letzte Funktion EndPaint lauten, da sie für das Besorgen und die Freigabe eines Device Contextes zuständig sind.
Wenn Sie in Borland C++ oder TurboPascal mit der Klassen-Bibliothek ObjectWindows arbeiten, existieren in der Klasse TWindow die zwei Methoden WMPAINT und Paint, wobei die zweitgenannte Methode von der ersten aktiviert wird.
Normalerweise wird die Methode Paint verwendet, weil sie den Vorteil bietet, das Handle auf den Device Context als ersten Parameter zu übergeben. Somit muß nicht mehr das Funktionspaar BeginPaint und EndPaint geschrieben werden, da es schon in der WMPaint-Methode aufgerufen wird. Diese beiden Methoden sind fast die einzigen Unterschiede zwischen einer Ausgabe in QuickC und einer Ausgabe in TurboPascal bzw. Borland C++.

4.1.2 Das Ereignis Paint

Bei der Programmierung mit VisualBasic wird auf Ereignisse reagiert, die eigentlich nichts anderes als die Windows-Meldungen sind und nur etwas anders heißen. So wird das Ereignis-Gegenstück zur WM_PAINT-Meldung mit dem Namen Paint bezeichnet. Wenn die Meldung an ein Fenster gerichtet ist, kann sie verarbeitet werden, indem die Ereignis-Prozedur Form_Paint behandelt und z.B. ein Kreis gezeichnet wird.

```
Sub Form_Paint
 Circle (2500, 2000), 1500
End Sub
```

Eigenschaft AutoRedraw

Die Entstehung eines Paint-Ereignisses hängt von der Einstellung der Form-Eigenschaft AutoRedraw ab. Sie bestimmt, ob es die Aufgabe von Windows oder des Programmierers ist, die zuvor verdeckten Bereiche der Ausgabe in dem nun aktiven Fenster zu erneuern, wenn es in den Vordergrund geholt wird. Wenn die Eigenschaft AutoRedraw den Wert falsch (FALSE) besitzt, entsteht das Ereignis Paint, und der Programmierer muß sich bei diesem Ereignis um die Ausgabe kümmern, wie es auch im Normalfall in den anderen drei Sprachen der Fall ist.
Warum es so wichtig ist, die Texte und Grafiken bei dem Paint-Ereignis bzw. bei der WM_PAINT-Meldung auszugeben, kann leicht mit Hilfe eines kleinen VisualBasic-Beispiels demonstriert werden. In einem Fenster soll immer eine Raute und ein Rechteck angezeigt werden. Falls die Zeichnung z.B. beim Anklicken des Fensters ausgegeben wird, und ein Fensterteil von einem anderen Fenster überdeckt wird, wird nach dessen Entfernen die Zeichnung nicht erneuert. Der Anwender müßte erst wieder das Click-Ereignis auslösen. Wurde jedoch ordnungsgemäß die Ausgabe bei dem Paint-Ereignis getätigt, so wird die Raute und das Rechteck neu gezeichnet, sobald das darüberliegende Fenster verschwunden ist.
Die Ereignis-Prozedur Form_Paint enthält folgenden Code, um die beiden Grafiken darzustellen. Wir werden uns ein wenig später noch genauer mit der Funktion Line beschäftigen.

```
Sub Form_Paint ()
HalbeBreite = ScaleWidth / 2
HalbeHoehe = ScaleHeight / 2
Line (HalbeBreite / 2, HalbeHoehe / 2)-
(3 * ScaleWidth / 4, 3 * ScaleHeight / 4), , B
Line (0, HalbeHoehe)-(HalbeBreite, 0)
Line -(ScaleWidth, HalbeHoehe)
Line -(HalbeBreite, ScaleHeight)
Line -(ScaleLeft, HalbeHoehe)
End Sub
```

Ein Paint-Ereignis tritt auch genauso wie eine WM_PAINT-Meldung bei der Größenänderung eines Fensters ein. Deswegen dürfen in dieser Prozedur keine Eigenschaften wie Width und ForeColor beschrieben werden, die ein erneutes Paint-Ereignis bewirken, da man ansonsten eine Endlos-Schleife programmiert.

AutoRedraw = wahr

Wenn Ihr Fenster keine Paint-Ereignisse erhalten soll, können Sie die Eigenschaft AutoRedraw auf wahr (TRUE) einstellen. In diesem Fall legt Windows dasselbe anzuzeigende Bild als Bitmap in den Speicher ab, um bei Bedarf den Fensterinhalt restaurieren zu können. Auf dieses Bitmap kann man programmtechnisch über die Eigenschaft Image zugreifen, die ein Handle darauf enthält. Das Handle kann nur gelesen, aber nicht überschrieben werden. Es kann z.B. der Eigenschaft Picture eines Fensters zugewiesen werden, um das Bitmap als Bild in ein anderes Fenster auszugeben.

```
Form2.Picture = Form1.Image
```

Wenn wir unser kleines Beispiel von vorhin noch einmal betrachten, so erscheint bei der auf wahr gesetzten Eigenschaft AutoRedraw keine Zeichnung, wenn die Ausgabe bei dem Paint-Ereignis erfolgt, jedoch wird sie automatisch immer wieder vollständig ausgegeben, wenn sie einmal durch ein Click-Ereignis in das Fenster geschrieben wurde.
Der Nachteil dieser Möglichkeit liegt in dem großen Verbrauch von Speicherplatz und der langsameren Ausgabe.