

Das kleine Visual Basic 2005- Weihnachtsbuch 2006

oder ein kleines Visual Basic-Programm für die Feiertage und darüber hinaus.

Stand: 24/12/2006

***** noch nicht auf Rechtschreibung und Grammatik gelesen *****

Eigentlich hätte dies ein richtiges »Weihnachtsbuch« werden sollen. Kein Buch über Weihnachten, sondern ein Buch, das man besonders um Weihnachten gerne liest, weil es viele Beispiele und kleine Projekte enthält, für die man ein wenig Zeit braucht, und die sich besonders an einem jener Tage, an denen weder Schule, Uni, Arbeit oder andere Verpflichtungen des täglichen Lebens eine Ausrede sein sollten. Doch wie es mit den guten Vorsätzen so ist, man schiebt sie gerne vor sich her. Ende Januar sagte ich mir, dieses Mal soll es etwas werden mit dem Weihnachtsbuch und irgendwann im November stellte ich wieder einmal fest, dass es mit einem richtigen Buch schon wieder viel zu spät ist (das hätte, damit es vor Weihnachten im Buchhandel stehen kann, spätestens im September fertig sein müssen – denken Sie bitte das nächste Mal daran, wenn Sie sich ein Weihnachtsbuch vornehmen – die Marktgesetze des Buchhandels nehmen auf Schöngeister und Träumer keinen Rücksicht, was nicht heißt, dass diese Bevölkerungsgruppe keine Bücher schreiben soll – sie müssen nur ein wenig früher abgeben, falls es mit dem Weihnachtsgeschäft noch etwas werden soll). Eine Ausrede habe ich allerdings. Mein Jetzt lerne ich Visual Basic 2005-Buch wurde erst Anfang Dezember fertig und zwei Bücher gleichzeitig zu schreiben würde ein »Dual Core-Brain« voraussetzen, dass von Intel&Co meines Wissens noch nicht erfunden wurde.

Da ich aber auf keinen Fall wieder ein Jahr verstreichen lassen wollte (sooft ist schließlich auch nicht Weihnachten), kam ich auf eine andere Idee. Statt eines Buches gibt es nur ein »Büchlein« (selbst diese Mengenkategorisierung ist für die folgenden knapp 15 Seiten noch ein wenig anmaßend) in Gestalt einer PDF-Datei (neudeutsch »ebook«). Und da ich dieses Werk ein paar Stunden vor Heiligabend zusammentippe, wird lediglich ein kleines, aber sicherlich sehr interessantes Beispielprojekt vorgestellt, dass Sie sie mit dem kostenlosen Visual Basic 2005 Express Edition umsetzen können. Auch wenn das Beispielprojekt überschaubar ist, lernen Sie ein paar wichtige Programmieretechniken, so dass auch etwas erfahrenere Visual Basic-Programmierer etwas lernen können (für die echten Profis ist das alles vermutlich ein wenig zu trivial). Genug der Vorrede, der Text soll ja diesen Weihnachten (also 2006) fertig werden.

Das Beispielprojekt stellt sich vor

Bei dem Beispielprojekt handelt es sich um ein kleines Video-Poker mit dem Sie »gegen den Computer«, der die Rolle eines Groupiers spielt, Karten spielen können. Es heißt daher auch *VB Poker*. Das Spiel läuft so ab, dass Sie bei jedem neuen Spiel 5 Karten erhalten, entsprechend der erhaltenen Karten jene Karten »dazukaufen«, die Ihnen ein günstigeres Blatt verschaffen sollen, und je nach Kartenglück Ihren Einsatz entweder verlieren oder vergrößern. Auch wenn der Zufallszahlengenerator, der in Gestalt der *Random*-Klasse zur Verfügung steht, offenbar nicht 100% gleichmäßige Zufallszahlen erzeugt, ist die Kartenziehung doch einigermaßen zufällig, so dass es manchmal relativ spannend wird – da aber 52 Karten im Spiel sind bedarf es schon einer gewissen Portion Glück, um zu einem vernünftigen Blatt zu kommen. Bluffen, ein wesentliches Element des Spiels können Sie übrigens nicht. Es handelt sich vielmehr um eines jener »Video-Poker«, die Sie vielleicht schon einmal in einem Spielcasino gesehen haben.

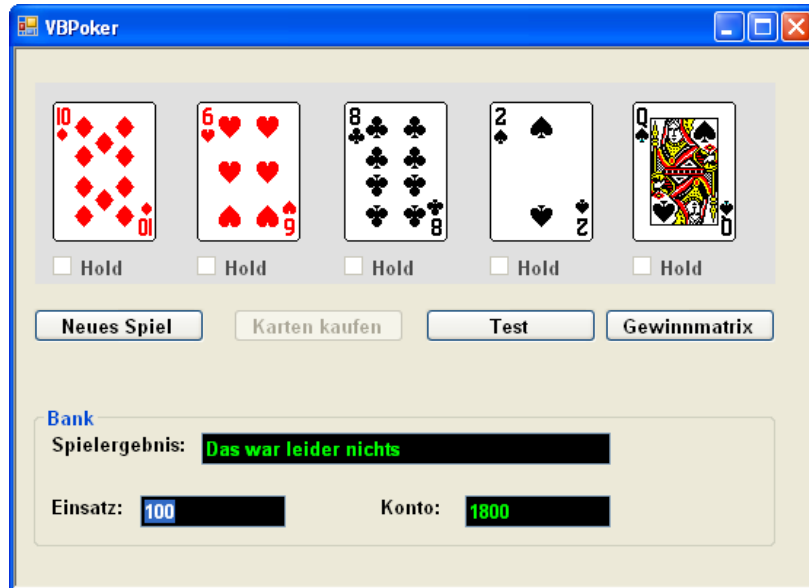


Abbildung 0.1: Das fertige Programm in Aktion

Pokern ist zur Zeit in Deutschland enorm populär¹, so dass das Spiel auch aus diesem Grund nicht uninteressant sein dürfte.

Falls Sie sich wundern, vorher die täuschend echten Kartenbilder stammen. Die wurden nicht aus einem anderen Kartenspiel herauskopiert, sondern stammen aus der (in Programmiererkreisen bekannten) Systemdatei *Cards.dll*, die ein Teil von Windows ist. Der Vorteil gegenüber statischen Bitmaps ist zum Beispiel, dass es eine Reihe von Deckblättern gibt und sogar einfache Animationen angezeigt werden (dieser Effekt wird in dem *VB Poker* aber nicht genutzt).

Die üblichen Formalitäten...

Der Autor kann leider keine Garantie übernehmen, dass das Programm tatsächlich wie versprochen funktioniert. Es wurde zwar getestet, aber nicht in dem Umfang, dass eine Hundertprozentige Fehlerfreiheit garantiert werden kann. Sollte jemanden einen Fehler finden, freut sich der Autor natürlich über einen entsprechenden Hinweis (bitte nur ernstgemeinte Zuschriften - E-Mail-Adresse am Ende des Textes).

¹ Der Autor ist allerdings kein Kartenspieler und sich daher auch nicht sicher, ob er alle Regeln richtig verstanden hat.

Auch ist der Quelltext nicht Hundertprozent optimal. Mir ging es in erster Linie darum, dass alles funktioniert. Aber mit etwas mehr Zeit zur Verfügung hätte ich den Quellcode etwas aufgeräumt und umstrukturiert.

Wo gibt es das fertige Projekt?

Da Sie das Beispiel vermutlich nicht selber Schritt für Schritt umsetzen möchten, sollten Sie sich als erstes das komplette Projekt herunterladen. Allerdings sollten Sie es auch nicht einfach nur starten und dann solange Karten spielen bis sich draußen im Garten oder Park die ersten Schneeglöckchen zeigen (was angesichts der nahenden »Klimakatastrophe« aber kein sehr zuverlässiger Indikator mehr ist), sondern sich erst einmal in den Quelltext vertiefen, um zu verstehen, wie das Programm funktioniert.

- Sie finden das fertige Projekt in Gestalt der Projektdatei und der zum Projekt gehörenden Quelltextdateien unter der Adresse <http://www.activetraining.de/buchdownloads/VBPoker.zip>.
- Falls Sie nur das Kartenspiel spielen möchten, benötigen Sie die Exe-Datei und die .NET 2.0-Laufzeit, die Sie vorher ausführen müssen, damit diese installiert wird. Die .NET-Laufzeit finden Sie unter dem Namen ».NET Framework Version 2.0 Redistributable Package (x86)« unter der allgemeinen Adresse www.microsoft.com/downloads (achten Sie dabei darauf, dass Sie die »deutsche Version« wählen).

Was lernen Sie an dem Programmbeispiel?

An dem kleinen Beispiel lernen Sie erstaunlich viel, wobei die genaue Menge natürlich immer von Ihrem (Er-) Kenntnisstand abhängig ist und »erstaunlich« eine relativ vage Mengenangabe ist. Hier eine stichwortartige Auflistung der wichtigsten »Lernerlebnisse«, die Ihnen das Beispielprojekt beschern wird:

- Der Aufruf von API-Funktionen (aus der *Cards.dll* des Betriebssystems)
- Das Freigeben von GDI-Grafikressourcen
- Der Umgang mit Steuerelementefeldern, die es offiziell bei Visual Basic 2005 gar nicht gibt.
- Das nachträgliche Einrichten von Ereignishandlern
- Der Umgang mit dem TabLayout-Control (gleich vorweg, eine echte Bereicherung ist es nicht gerade)
- Der Umgang mit Konstantenaufstellungen.

- Die Umsetzung einer »Entscheidungsmatrix« (konkret, feststellen, welches Blatt ein Spieler besitzt anhand einiger Entscheidungen, die gar nicht so leicht umzusetzen waren).
- Und einiges mehr.

Die Umsetzung im Detail

Gleich vorweg, eine Schritt für Schritt-Umsetzung wäre etwas zu aufwändig und Sie müssten vor allem relativ viel tippen, da einige Prozeduren zwangsläufig ein wenig umfangreicher sind. Die folgende Schrittfolge beschreibt die Umsetzung der Oberfläche und stellt anschließend die einzelnen »Codepunkte« vor. Ein solcher Codepunkt ist einfach nur eine Prozedur oder Funktion im Quelltext, die eine Nummer erhält, die in der Kommentarkopfzeile enthalten ist, so dass Sie sie schnell finden sollten. Es handelt sich also mehr um eine Umsetzung im Schnelldurchlauf.

Die Umsetzung beginnt

Schritt 1:

Grundlage ist ein Windows-Projekt, das Sie bei Visual Basic über *Datei/Neues Projekt* anlegen. Wählen Sie als Projekttyp »Windows-Anwendung«. Sie erhalten dadurch ein Projekt, zu dem bereits ein Formular gehört.

Geben Sie dem Projekt einen Namen und speichern Sie es gleich über *Datei/Alles speichern* ab. Sie speichern damit nicht nur die Projektdatei, sondern auch alle zum Projekt gehörenden Dateien.

Schritt 2:

Ordnen Sie auf dem Formular folgende Steuerelemente an:

- Ein Panel
- In dem Panel fünf PictureBoxen und darunter 5 CheckBoxen
- 3 Buttons (der Test-Button dient nur zu Testzwecken und hat keine offizielle Funktion).
- Eine GroupBox
- In der GroupBox zwei Textboxen und dazugehörige Labels.

Orientieren Sie sich bei der Umsetzung an Abbildung 0.2, wobei der Name des Steuerelements, sofern er vergeben wird, in Klammern gesetzt ist.

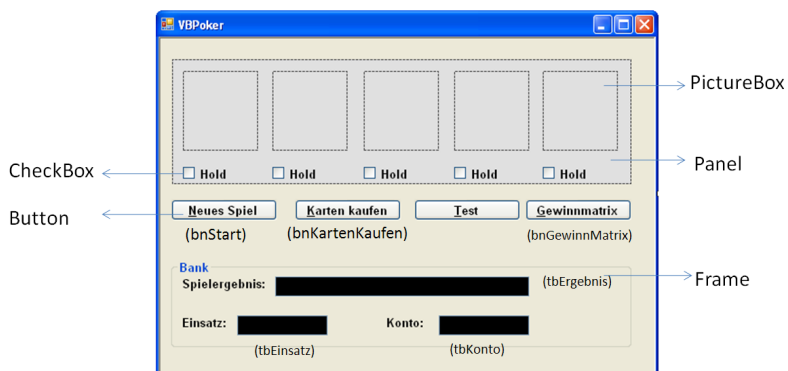


Abbildung 0.2: Das Hauptformular und seine Steuerelemente

Schritt 3:

Ordnen Sie auf dem Formular ein Panel (es stellt lediglich eine Fläche zur Verfügung und verhält sich ansonsten vollkommen passiv) an und in dem Panel fünf PictureBoxen und darunter jeweils eine Checkbox. Die Namen werden nicht geändert, da die Steuerelemente vom Programm über ein Feld angesprochen werden.

Schritt 4:

Ordnen Sie drei Buttons an.

Hinweis:

Der *Test*-Button hat keine echte Funktion. Er soll lediglich, die Bewertung der aktuellen »Kartenhand« anzeigen (um testen zu können, ob die Bewertung auch tatsächlich funktioniert).

Schritt 5:

Ordnen Sie unterhalb der Buttons eine Groupbox an. Auch sie spielt eine rein passive Rolle und dient lediglich zur optischen Abgrenzung der in ihr enthaltenen Steuerelemente.

Damit ist das Hauptformular auch schon fertig und es kann an die Programmierung gehen.

Schritt 6:

Im Codepunkt Nr. 1 werden eine Reihe von Variablen benötigt, die in der gesamten Formularklasse angesprochen werden sollen:

```
Private BildFeld() As PictureBox
Private HoldFeld() As CheckBox
Private KartenKaufenFeld(4) As Boolean
Private KartenGezogenFeld(4) As Byte
Private Einsatz As Short = 100
Private Konto As Short = 2000
Private SoundFilePfad As String
```

Es ist eine Eigenheit von Visual Basic, dass die Formularklasse automatisch mit *Public* ausgestattet wird, was aber nicht erforderlich ist, da sie praktisch nie von außerhalb des Programms angesprochen wird. In diesem Projekt wird sie daher mit *Friend* deklariert. Das hat zur Folge, dass alle Enumerationen ebenfalls explizit mit *Friend* deklariert werden müssen. *Friend* bedeutet generell, dass die Klasse oder Variable überall im Programm, aber nicht von anderen Programmen angesprochen werden kann.

Da ein paar Klassen im Namespace *System.IO* angesprochen werden sollte oberhalb des *Class*-Befehls ein *Imports*-Befehl stehen:

```
Imports System.IO
```

Schritt 7:

Im Codepunkt Nr. 2 wird die Ereignisprozedur *Form_Load* mit Inhalten gefüllt. Hier erfolgt der Aufruf der API-Funktion *cdtInit*, die die Karten-DLL initialisiert. Beachten Sie, dass die Argumente *Breite* und *Hoehe* per *ByRef* übergeben werden. Die API-Funktion trägt hier die Größe einer Kartenbitmap ein, so dass sie anschließend im Programm zur Verfügung stehen.

Ein wenig ungewöhnlich, aber sehr wichtig, ist der folgende Befehl:

```
BildFeld = New PictureBox() {PictureBox1, PictureBox2,
PictureBox3, PictureBox4, PictureBox5}
```

Er fasst die fünf PictureBoxen zu einem Feld zusammen, so dass sie im Stile eines Steuerelementefeldes angesprochen werden können.

Damit nach dem Anklicken einer der PictureBoxen etwas passiert (auf diese Weise wird der »Cheat-Modus« eingebaut, der bewirkt, dass nach dem Anklicken mit der linken Maustaste der Wert der Karte und mit dem Anklicken mit der rechten Maustaste die Farbe der Karte geändert wird), wird ein Eventhandler nachträglich eingerichtet:

```
For i As Short = 0 To BildFeld.Length - 1
    BildFeld(i).Width = Breite
    BildFeld(i).Height = Hoehe
    AddHandler BildFeld(i).MouseClicked, _
        AddressOf KartenKlickHandler
Next
```

Jede PictureBox erhält die von *cdInit* gelieferte Breite und Höhe, über *AddHandler* wird erreicht, dass das *MouseClicked*-Event zum Aufruf der Prozedur *KartenKlickHandler* führt.

Auch für die fünf CheckBoxen wird ein solcher Eventhandler eingerichtet. Zuvor werden sie ebenfalls zu einem Array zusammengefasst:

```
HoldFeld = New CheckBox() {CheckBox1, CheckBox2, CheckBox3,
CheckBox4, CheckBox5}
For i As Short = 0 To HoldFeld.Length - 1
    AddHandler HoldFeld(i).CheckedChanged, AddressOf
    CheckBoxHandler
    HoldFeld(i).Tag = i
Next
```

Ebenfalls ein wenig ungewöhnlich ist, dass vom *ProcessExit*-Event des aktuellen Prozesses (also des ausführenden Programms) Gebrauch gemacht wird. Diese Maßnahme soll erreichen, dass am Ende in jedem Fall die API-Funktion *cdtTerm* aufgerufen wird, welche die Kartenbibliothek wieder entlädt:

```
AddHandler AppDomain.CurrentDomain.ProcessExit, AddressOf
ProzessEndeHandler
```

Die *CurrentDomain* steht für die aktuelle Anwendungsdomäne und damit in der Welt von .NET für das aktuelle Programm.

Da das Anzeigen einer Karte von einem Klick-Geräusch begleitet werden soll, wird am Ende von *Form_Load* geprüft, ob die Datei *Start.wav* überhaupt existiert:

```
' Gibt es den Soundfile?  
SoundFilePfad =  
Environment.GetEnvironmentVariable("systemroot") &  
"\Media\Start.wav"  
If Not File.Exists(SoundFilePfad) Then  
    SoundFilePfad = ""  
End If
```

Abgespielt wird die Wav-Datei später über:

```
My.Computer.Audio.Play(SoundFilePfad)
```

Schritt 8:

Im Codepunkt Nr. 3 wird der in *Form_Load* eingerichtete *CheckBox*-Handler umgesetzt:

```
Sub CheckBoxHandler(ByVal Sender As Object, ByVal e As  
EventArgs)  
    Dim Cb As CheckBox = CType(Sender, CheckBox)  
    KartenKaufenFeld(Cb.Tag) = Cb.Checked  
End Sub
```

Schritt 9:

Codepunkt Nr. 4 ist die bereits erwähnte »Cheat-Funktion«, die aber nicht zum Mogeln da ist, sondern zum Testen des Programms, da sich Kartenkonstellationen so zusammenklicken lassen.

Schritt 10:

Codepunkt Nr. 5 ist der bereits erwähnte *ProcessExit*-Handler, der lediglich die API-Funktion *cdiTerm* aufruft.

Schritt 11:

Codepunkt Nr. 6 ist die *Click*-Prozedur des Buttons. Hier wird das Spiel gestartet.

Schritt 12:

Codepunkt Nr. 7 ist die Prozedur *NeuesSpiel()*, die einen Satz neuer Karten zieht. Dabei muss natürlich sichergestellt werden, dass keine Karte doppelt gezogen wird. Das wird über die sehr praktische *IndexOf*-Funktion der *Array*-Klasse verwirklicht:

```
Do  
    Farbe = R.Next(0, 4)
```

```

Wert = R.Next(0, 13)
Loop Until Array.IndexOf(KartenGezogenFeld,
GetKartenWert(Farbe, Wert)) = -1
KartenGezogenFeld(i) = Wert * 4 + Farbe

```

Da *IndexOf* einen beliebigen *Object*-Wert erwartet, muss der Typ gegebenenfalls über *CType* angepasst werden – würde *GetKartenWert* keinen *Byte*-, sondern einen *Short*-Typ zurückgeben, würde es nicht funktionieren und *IndexOf* würde stets -1 zurückgeben.

Schritt 13:

Codepunkt Nr. 8 ist eine kleine Funktion, die aus Kartenwert und Kartenfarbe jene Nummer »berechnet«, die beim Aufruf der API-Funktion *cdtDrawExt* übergeben werden muss. Dabei gilt die einfache Regel, dass *Kreuz Ass* den Wert 0, *Karo Ass* den Wert 1, *Herz Ass* den Wert 2, *Pik Ass* den Wert 3, *Kreuz Zwei* den Wert 4, *Karo Zwei* den Wert 5 usw. besitzt.

Kartennummer	Karte
0	Kreuz Ass
1	Karo Ass
2	Herz Ass
3	Pik Ass
4	Zwei Kreuz
5	Zwei Karo
6	Zwei Herz
7	Zwei Pik
8	Drei Kreuz
50	König Herz
51	König Pik
52	Deckblatt Nr. 1

Tabelle 0.1: Ein paar Beispiele für Kartennummern

Schritt 14:

Codepunkt Nr. 9 ist die Prozedur *KartenKaufen*, die für jede mit Hold belegte *PictureBox* eine neue Karte zieht und natürlich ebenfalls sicherstellen muss, dass diese nicht bereits gezogen wurde.

Schritt 15:

Codepunkt Nr. 10 ist ein gemeinsamer *KeyPress*-Handler für die beiden Textboxen, der erreicht, dass nur Ziffern und die Backspace-Taste erlaubt sind. Bei allen übrigen Tasten wird der Tastencode durch Setzen von *Handled* auf *True* verschluckt. Beachten Sie, dass es kein Problem ist, dass auf *Handles* mehrere Events folgen können:

```
Private Sub TbKeyHandler(ByVal sender As System.Object,  
ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles  
tbEinsatz.KeyPress, tbKonto.KeyPress
```

Schritt 16:

Codepunkt Nr. 11 zeigt das zweite Formular an, von dem noch die Rede sein wird.

Schritt 17:

Codepunkt Nr. 12 sorgt für das Nachkaufen von Karten.

Schritt 18:

Codepunkt Nr. 13 ist die umfangreiche Funktion *CardsBewerten()*, die feststellt, welches Blatt aktuell durch die fünf *PictureBox*en vorliegt. Es mag sein, dass es effektivere Möglichkeiten gibt, dies festzustellen. Die Funktion gibt einen Wert vom Typ *ErgebnisEnum* zurück.

Schritt 19:

Codepunkt Nr. 14 übernimmt die Auszahlung anhand des ermittelten Ergebnisses.

Schritt 20:

Codepunkt Nr. 15 ist eine weitere »Testfunktion«, die den Wert des aktuellen Blattes anzeigt.

Schritt 21:

Fügen Sie über *Projekt/Modul hinzufügen* ein weiteres Modul hinzu. Hier werden eine Reihe allgemeiner Prozeduren untergebracht, die theoretisch von allen anderen Formularen des Projekts aufgerufen werden können. Geben Sie dem Modul den Namen »basKarten«.

Schritt 22:

Die drei API-Funktionen in *Cards.dll* werden wie folgt deklariert:

```
<DllImport("Cards.dll")> _  
Friend Function cdtInit(<[In]()> ByRef Width As Integer,  
<[In]()> ByRef Height As Integer) As Boolean  
End Function  
  
<DllImport("Cards.dll")> _  
Friend Sub cdtTerm()
```

```

End Sub

<DllImport("Cards.dll")> _
Friend Function cdtDrawExt(ByVal hdc As IntPtr, ByVal x
As Integer, ByVal y As Integer, ByVal dx As Integer, ByVal
dy As Integer, ByVal ecsCard As Integer, ByVal ectDraw As
Integer, ByVal clr As Integer) As Integer

End Function

<DllImport("Cards.dll")> _
Friend Function cdtAnimate(ByVal hdc As IntPtr, ByVal
ecbCardBack As Integer, ByVal x As Integer, ByVal y As
Integer, ByVal iState As Integer) As Integer

End Function

```

Damit es funktioniert, muss ein *Imports*-Befehl vorangestellt werden:

```
Imports System.Runtime.InteropServices
```

Schritt 23:

Das Modul enthält die Funktion *KarteAnzeigen()* (Codepunkt Nr. 19), die eine Spielkarte anzeigt. Sie ist zweifach überladen, damit sie einmal lediglich mit einer *PictureBox* als Argument aufgerufen werden kann:

```

Sub KarteAnzeigen(ByVal Pb As PictureBox, ByVal Wert As
KartenWert, ByVal Farbe As KartenFarbe)
    If Wert < 0 And Wert > 12 Then
        Throw New ArgumentOutOfRangeException("Kartenwert muss
zwischen 0 und 12 liegen.")
    End If
    Dim Breite, Hoehe As Integer
    Breite = Pb.Width
    Hoehe = Pb.Height
    Dim KartenWert As Integer = Wert * 4 + Farbe
    Dim g As Graphics = Pb.CreateGraphics
    Dim hdc As IntPtr = g.GetHdc
    Try
        cdtDrawExt(hdc, 0, 0, Breite, Hoehe, KartenWert, 0,
Color.White.ToArgb)
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    Finally
        g.ReleaseHdc()
    End Try
End Sub

```

Schritt 24:

Fügen Sie über *Projekt/Windows Form hinzufügen* ein weiteres Formular hinzu, geben Sie ihm den Namen »fmErgebnismatrix«. Hier soll lediglich angezeigt, welche Kartenkombinationen welchen Gewinn ergeben.

Schritt 25:

Das Formular soll zu jeder Kartenkombination passende Karten anzeigen. Eine gute Gelegenheit, das mit VB 2005 neu hinzugekommene TabLayoutPanel-Control eine Chance zu geben, das einen Tabellenrahmen vorgibt, in dem Steuerelemente angeordnet werden. Der Vorteil ist (wie bei HTML-Tabellen), dass die Steuerelemente dadurch automatisch an festen Positionen erscheinen. Um es vorweg zu nehmen, das TabLayoutPanel-Control löst diese Aufgabe, verhält sich aber wie eine typische 1.0-Version und bietet einen minimalen Komfort. Der einzige Weg, Steuerelemente in den einzelnen Spalten anordnen zu können besteht anscheinend darin, sie aus der Toolbox dort abzulegen. Insgesamt umfasst die Tabelle 7 Spalten und 10 Zeilen. Das Einstellen einer einheitlichen Größe etwa der PictureBoxen ist dagegen dank des *Format*-Menüs relativ einfach.

					Paar	-
					Dreier	2x Einsatz
					Doppel	3x Einsatz
					Full House	10x Einsatz
					Vierer	12x Einsatz
					Straße klein	4x Einsatz
					Straße groß	6x Einsatz
					Flush	5x Einsatz
					Straight Flush	15x Einsatz
					Royal Flush	20x Einsatz

Karten anzeigen

Abbildung 0.3: Das zweite Formular besteht aus einem `TableLayout`-Steuerelement, in deren Zellen `PictureBox`en und `Labels` angeordnet wurden

Schritt 26:

Das Anzeigen der Kartenbildchen ist eine reine Formsache, die von der Prozedur `KartenAnzeigen()` durch den Aufruf der Prozedur `KarteAnzeigen()` (insgesamt 43 Mal) erledigt wird.

Nobody's perfect

Auch der Autor nicht, der zudem Visual Basic schon »relativ lange« kennt². Ein ungelöstes Problem ist der Umstand, dass nach dem Aufruf des zweiten Formulars die Kartenbilder erst in einer separaten *Click*-Routine eines Buttons gezeichnet werden. Wird die Prozedur *KartenAnzeigen()* bereits. Sollte daher Jemand eine Lösung haben und wissen wie es sich erreichen lässt, dass die Karten bereits mit dem Laden des Formulars erscheinen und nicht wieder gelöscht werden würde sich der Autor über einen Hinweis freuen.

² Um genau zu sagen seit dem mir eines schönen Tages eine Diskette mit der Aufschrift »Thunder« überreicht wurde.

Es gibt noch einiges zu tun

Pokern spielen können Sie mit dem Programm, doch das heißt noch nicht, dass es keinen Raum für Erweiterungen gibt. Hier ein paar Ideen, die Sie alle einmal umsetzen oder zu mindestens einmal andenken sollten:

- Der Kontostand sollte natürlich nicht negativ werden können.
- Eine zeitliche Begrenzung wäre ganz nett, so dass ein Spiel z.B. nach 30 Minuten wieder zu Ende ist. Das lässt sich mit Hilfe einer Timer-Komponente realisieren.
- Es sollte eine Bestenliste geben, in der sich jeder Spieler verewigen kann, der einen bestimmten Betrag, etwa 1000, gewonnen hat.
- Die letzte Erweiterung klingt spektakulär, ist aber alles andere als unlösbar. Das Spielen mit anderen Spielern im Netzwerk. Grundlage sind die »Socket-Klassen« im Namespace *System.net*, mit deren Hilfe sich relativ einfach (Text-) Nachrichten im Netzwerk verschicken und empfangen lassen. Das Spiel über das Internet zu spielen ist dann nur noch eine Frage der Adresse.

Werbung:

In meinem Buch *Visual Basic 2005 Kompendium* gibt es eine kleine Einführung in die Netzwerkprogrammierung, wenngleich es nicht um ein »Netzwerk-Poker« geht.

Damit wünsche Ihnen viel Spaß mit dem Visual Basic-Programm, ein wenig mit dem Spiel selber und hoffe, dass Sie durch das Umsetzen des Beispiels und dem Beschäftigen mit dem Quellcode einiges dazu lernen. Im nächsten Jahr wird es ein richtiges Weihnachtsbuch geben.

Der Autor (Peter Monadjemi)

Remseck, 24. Dezember 2006

Per E-Mail erreichen Sie mich unter pm@activetraining.de.